Gurobi 快速入门手册

版本 4.6.1，Copyright © 2011，Gurobi Optimization，Inc.

Gurobi 优化器由刃之砺信息科技（上海）有限公司在中国地区代理，
联系电话 021-61508060，网址 www.edgestone-it.com/gurobi.htm

页面：Gurobi Optimizer Quick Start Guide

Gurobi 优化器快速入门指南

版本 4.6，Copyright © 2011，Gurobi Optimization，Inc.

本文将简要地介绍 Gurobi™ 优化器。首先要介绍的是软件安装指南和如何获取并安装 Gurobi 许可证信息，此外本文还会包括对 Gurobi 交互式命令解释器的简要说明。我们建议您从这三节开始阅读。

取决于您将要使用的 编程语言，您还可以阅读 Gurobi 的 C 接口、C++ 接口、Java® 接口、Microsoft®.NET 接口、或 Python® 接口等专门章节。我们所提供的 Python 接口对于那些想要使用更高层次的建模结构来生成模型的用户来说将是十分重要的。

本文之后还会介绍文件概述，在该节中会对 Gurobi 发布中所包含的文件信息进行简要的概述。

关于 Gurobi 接口的更多信息请参见 Gurobi 参考手册。Gurobi 发布中所提供的示例可参见 Gurobi 示例导览。

页面：Installation

安装

使用 Gurobi 优化器的第一步就是要在您的电脑上安装相关的文件。您应该已经从 Gurobi 网站（www.gurobi.com）获取了 Gurobi 的发布包。该发布包的名称取决于 Gurobi 优化器的版本，以及要运行它的系统平台。例如，文件 *Gurobi-4.6.0-win32.msi* 可以在一台运行 Microsoft Windows® 操作系统的机器上安装 Gurobi 4.6.0 的 32-位版本。另一方面，文件 *gurobi4.6.0_linux64.tar.gz* 可以在一台运行支持版本的 Linux® 操作系统的机器上安装 Gurobi 4.6.0 的 64-位版本。

安装 Gurobi 优化器所需的步骤在 Windows、Linux、和 Mac OS 系统上不尽相同。下面将对每个平台进行详细的说明。

安装—Windows

要在 Windows 机器上安装 Gurobi 优化器，只须要简单地双击合适的 Gurobi 安装程序（例如，Gurobi 4.6.0 的 32-位版本的安装程序为 *Gurobi-4.6.0-win32.msi*）。默认地，安装程序会将 Gurobi 4.6.0 的文件存放到 *c:\gurobi460\win32*（64-位的 Windows 安装程序会将文件存放到 *c:\gurobi460\win64*）路径下。在安装程序中可以选择要安装的目标路径。我们用 *<installdir>* 来引用安装路径。

默认的 Gurobi 安装只允许当前的 Windows 用户使用它。如果想要让所有的用户都可以使用 Gurobi，您可以使用 Windows 安装程序的命令行接口来实现：打开一个 *cmd* 提示窗口，使用 *cd* 命令定位到包含 Gurobi 安装程序映像的路径，并输入以下命令：

```
  msiexec /i Gurobi-4.6.0-win32.msi allusers=1
```

如果您想要在 Gurobi 优化器内使用压缩文件，我们建议您同时安装 gzip。可以从 www.gzip.org 下载该软件。

安装一旦完成之后，您应该可以在桌面上看到 Gurobi 的快捷方式。可以用它来运行 Gurobi 交互式命令解释器。现在您还不应该尝试去运行 Gurobi。这样做会产生冗长的错误信息指示您还未安装许可证。

您现在可以继续阅读如何获取并安装 Gurobi 许可证节。

如果您想要了解 Gorubi 发布中所包含的文件信息，请参阅文件概述节。

安装——Linux

在 Linux 系统中安装 Gurobi 优化器的第一步是选择一个目标路径。对于共享安装我们建议使用 */opt*，但其他的路径也能够工作。

一旦选择了目标路径，下一步就是将 Gurobi 发布复制到该目标路径中并提取其内容。可以使用以下命令来完成内容提取：

```
tar xvfz gurobi4.6.0_linux64.tar.gz
```

该命令会创建一个子路径 *gurobi460/linux64*，其中包含了完整的 Gurobi 发布。默认情况下，*<installdir>* 为 */opt/gurobi460/linux64*。当然，须要对文件和路径名称进行调整以反映真正要安装的发布（例如，提取 32-位的 Linux 发布（*gurobi4.6.0_linux32.tar.gz*）则会创建 */opt/gurobi460/linux32*）。

Gurobi 优化器会使用一些可执行文件。为了在需要时能够找到这些文件，您须要修改一些环境变量：

- *GUROBI_HOME* 应该指向 *<installdir>*。
- 应该扩展 *PATH* 以包含 *<installdir>/bin*。
- 应该扩展 *LD_LIBRARY_PATH* 以包含 *<installdir>/lib*。

Bash shell 的用户须要在 *.bashrc* 文件中添加以下行…

```
export GUROBI_HOME="/opt/gurobi460/linux64"
export PATH="${PATH}:${GUROBI_HOME}/bin"
export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:${GUROBI_HOME}/lib"
```

csh shell 的用户则须要在 *.cshrc* 文件中添加以下行…

```
setenv GUROBI_HOME /opt/gurobi460/linux64
setenv PATH ${PATH}:${GUROBI_HOME}/bin
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${GUROBI_HOME}/lib
```

如果还没有设置 *LD_LIBRARY_PATH*，请改为使用以下行进行设置：

```
export LD_LIBRARY_PATH="${GUROBI_HOME}/lib"
```

或

```
setenv LD_LIBRARY_PATH ${GUROBI_HOME}/lib
```

同样地，须要对这些路径进行调整以反映真正要安装的发布。

您现在可以继续阅读如何获取并安装 Gurobi 许可证节。

如果您想要了解 Gorubi 发布中所包含的文件信息，请参阅文件概述节。

安装——Mac OS

要在 Mac 上安装 Gurobi 优化器，只须要简单地双击合适的 Gurobi 安装程序（例如，Gurobi 4.6.0 的安装程序为 *gurobi4.6.0.pkg*），并遵循其提示。默认地，安装程序会将 Gurobi 4.6.0 的文件存放到 */Library/gurobi460/mac64*。

您现在可以继续阅读如何获取并安装 Gurobi 许可证节。

如果您想要了解 Gorubi 发布中所包含的文件信息，请参阅文件概述节。

页面：How to Obtain and Install a Gurobi License

如何获取并安装 **Gurobi** 许可证

通过以下两个步骤可以获取一个新的 Gurobi 许可证密钥：首先在我们的网站上创建一个新的许可证，然后生成一个许可证密钥将我们网站上的那个许可证与您的机器进行关联。下面让我们依次对这两个步骤进行讨论：

新许可证的创建

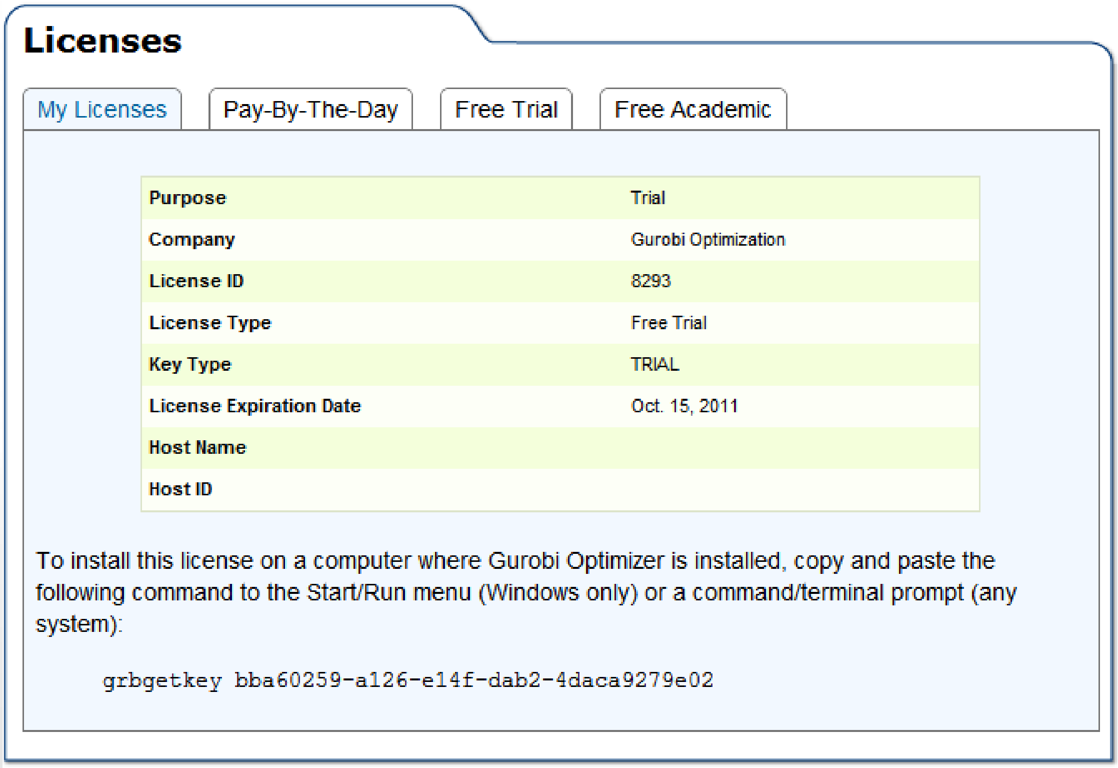
创建许可证的过程取决于要创建的许可证类型：

- **试用或教学许可证**：可以直接在我们的网站上创建试用或教学许可证：访问 *www.gurobi.com* Gurobi 帐户中“*Licenses*”区域的“*Free Trial*”或“*Free Academic*”页面、接受许可证协议、然后单击“*Submit Request*”。您立即就可以在“*My Licenses*”页面中看到您所创建的许可证。您可以根据需要创建任意多个试用或教学许可证。

- **当日付款许可证**：当日付款许可证也可以直接在我们的网站上创建：访问 *www.gurobi.com* Gurobi 帐户中“*Licenses*”区域的“*Pay-By-The-Day*”页面，然后将您想要购买的许可证添加到购物车中。准备好之后，单击“*Checkout*”，就会跳转到 Google 支付页面，在该页面中可以通过信用卡来支付您所购买的许可证。几分钟之内（一旦支付过程完成以后）您就可以在“*My Licenses*”页面中看到新创建的许可证。

- **其他的收费许可证**：关于其他的收费许可证，请联系我们（*license@gurobi.com*）以便安排支付事宜。一旦收到付款我们就会立即与您取得联系，此时您就可以在“*My Licenses*”页面中看到新创建的许可证。

在“*My Licenses*”页面看到新的许可证之后，单击*许可证 ID* 会生成如下所示的页面：



Purpose	Trial
Company	Gurobi Optimization
License ID	8293
License Type	Free Trial
Key Type	TRIAL
License Expiration Date	Oct. 15, 2011
Host Name	
Host ID	

To install this license on a computer where Gurobi Optimizer is installed, copy and paste the following command to the Start/Run menu (Windows only) or a command/terminal prompt (any system):

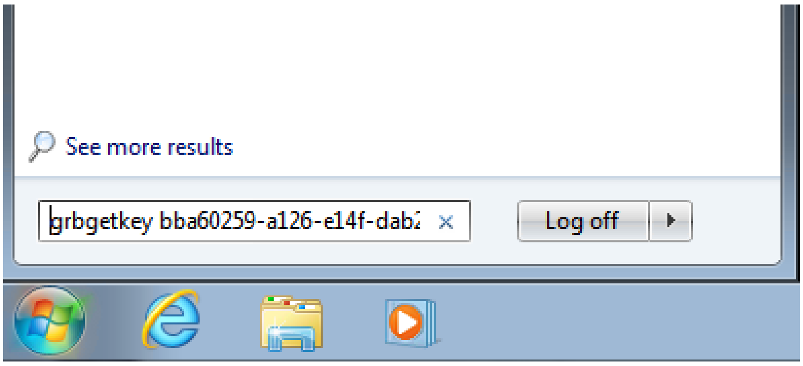
```
grbgetkey bba60259-a126-e14f-dab2-4daca9279e02
```

下一个步骤就是获取一个*许可证密钥*，将新创建的许可证与您的机器关联起来。

许可证密钥的获取

要获取 Gurobi 的许可证密钥，请先将一台须要运行 Gurobi 优化器的机器连接到网络、然后在这台机器上运行 *grbgetkey* 命令。如果您所使用的是免费的教学许可证，那么您的机器就必须连接到可以辨识的教学域中。如果您的电脑未连接到网络，并且您所请求的是收费许可证，请联系我们（*license@gurobi.com*）来得到关于如何获取许可证密钥的进一步说明。请注意，一旦获取了许可证密钥就不再需要网络连接了。

特定许可证所对应的须要运行的具体命令位于许可证页面的底部（例如，*grbgetkey bba60259-...*）。我们建议您使用复制-粘帖的方法从我们的网站获取完整的 *grbgetkey* 命令。在 Windows 系统中，可以直接将该命令粘帖在 Windows 的*搜索*框（在 Windows XP 中称为*运行*框）中。在 Windows 7 中如下所示：



grbgetkey bba60259-a126-e14f-dab2

在 Linux 机器上，将该命令粘帖在 shell 窗口中。在 Mac 系统中，则将该命令粘帖在 *Terminal* 窗口中。

grbgetkey 程序会将机器的识别信息传递回我们的网站，而网站则会回复许可证密钥。这样的信息交换一旦发生，*grbgetkey* 就会询问用于存储许可证密钥文件的路径名称。虽然可以将该文件存储在任何地方，但我们还是建议您接受默认的位置（简单地敲击*回车键*即可）。

许可证密钥文件的存储位置

运行 Gurobi 优化器时，它会在三个不同的默认位置寻找 *gurobi.lic* 许可证密钥文件。始终会查找的是主目录。此外，在 Windows 系统中，Gurobi 优化器 4.6.0 还会查找 *c:\gurobi460* 或 *c:\gurobi46* 路径。在 Linux 系统中查找的是 */opt/gurobi460* 以及 */opt/gurobi46*。而在 Mac 系统中则会查找 */Library/gurobi460* 和 */Library/gurobi46*。请注意这些默认路径都是绝对路径，所以举例来说，即使软件安装在 *d:\gurobi460*，Gurobi 仍然会在 *c:\gurobi460* 中寻找许可证密钥文件。

如果想要将您的许可证存储在一个非默认的位置，您就必须设置环境变量 *GRB_LICENSE_FILE* 使其指向许可证密钥文件。*重要注意事项：环境变量应该指向许可证密钥文件本身，而不是指向包含该文件的路径。*

在 Windows 系统中，可以在控制面板中创建并修改环境变量。对于 Windows Vista® 或 Windows 7，在控制面板的搜索框中输入“*Environment Variables*”，系统就会引导您进入正确的页面。对于 Windows XP®，在控制面板的“*系统*”窗口中选择“*高级*”页面，单击“*环境变量*”按钮来访问环境变量窗口。如果想要将许可证密钥文件放置在非默认的位置，就须要添加一个名为 *GRB_LICENSE_FILE* 的新*系统变量*，并将其设置为许可证文件所在的位置。

在 Linux 系统中，如果选择将许可证密钥文件放置在非默认的位置，就须要在 *.bashrc* 文件中添加一行，如下所示：

```
  export GRB_LICENSE_FILE=/usr/home/jones/gurobi.lic
```

对于 Linux *csh* shell 的用户，须要在 .cshrc 文件中添加以下行：

```
  setenv GRB_LICENSE_FILE /usr/home/jones/gurobi.lic
```

对变量进行设置使其指向许可证密钥文件的实际位置。

在 Mac 系统中，可以通过 *environment.plist* 来设置 *GRB_LICENSE_FILE* 环境变量，如 Apple’s Runtime configuration Guidelines 或 Apple Technical Q&A 1067 中所述。

最后的步骤

大多数的用户此时就已经可以使用许可证了。惟一的例外是那些使用免费的教学许可证、以及浮动或单用户许可证的情况。如果您所使用的是免费的教学许可证，*grbgetkey* 获得了许可证密钥之后就会接着进行教学许可证的验证步骤。该步骤会检查您的域名是否在我们已知的教学域列表中。如果这一步骤出现问题，关于如何继续的信息请参见验证疑难解答。

如果您所使用的是浮动或单用户许可证，您就须要建立 Gurobi 令牌服务器。

许可证的测试

一旦获得了机器的许可证密钥，就可以运行 Gurobi 交互式命令解释器了。在 Windows 系统中，可以直接双击桌面上的 Gurobi 图标（或在 *cmd* 窗口中输入 *gurobi.bat*）；在 Linux 或 Mac OS 系统中，则须要在 *Terminal* 窗口中输入 *gurobi.sh*。命令解释器会产生以下输出：

```
Gurobi Interactive Shell, Version 4.6.0
Copyright (c) 2011, Gurobi Optimization, Inc.
Type "help()" for help

gurobi>
```

现在可以使用 Gurobi 交互式接口了。

一些 Windows 用户报告说他们在运行安装程序之后无法启动 Gurobi 命令解释器。您可能须要先注销然后再重新登录，使得安装程序所修改的环境变量生效.

小节

- 教学验证疑难解答
- 令牌服务器的建立

页面：Academic validation troubleshooting

教学验证疑难解答

虽然大多数的教学域验证都能够顺利进行，但我们还是遇到过一些验证问题，这些问题需要您的一些操作来解决。

如果 *grbgetkey* 产生了如下所示的信息…

```
  ERROR: Unable to determine hostname for IP address 234.28.234.12.
```

…这意味着您的机器没有反向 DNS 信息。当您连接到互联网的 DHCP 服务器执行 NAT（网络地址转换）或 PAT（端口地址转换）而未提供其客户端的 DNS 信息时通常会发生这种情况。最简单的解决方法就是要求您的网络管理员为 DHCP 设备本身添加一个 DNS 项（更具体来说，一个 *PTR* 记录）。

如果 *grbgetkey* 产生如下所示的信息…

```
  ERROR: your hostname (mymachine.mydomain)
  was not recognized as belonging to an academic domain.
```

…这意味着您所在的域不在我们的教学域列表中。请确认您所连接的是学校的网络。如果所报告的主机名是一个有效的学校地址，请将您所收到的具体的错误信息发送到 *support@gurobi.com*，我们会添加您所在的域。

通过 VPN 连接到您的教学域通常不会导致验证问题，因为请求看上去是来自教学域内部的。然而，一些 VPN 会使用*分割隧道*，此时与公共网站的通信将通过您的 ISP 发送。分割隧道会将重大的安全漏洞引入专用网络，所以不是那么常用。如果您在连接到 VPN 时发现验证请求遭到拒绝，您可以询问网络管理员是否可以配置 VPN 将数据通过专用网络发送到 *gurobi.com*。

一些机器是通过代理服务器连接到网络的。不幸的是，这样的配置不符合我们的教学验证过程。

页面：Setting up a token server

令牌服务器的建立

重要注意事项：大多数的许可证，包括免费的试用与教学许可证，都不须要使用令牌服务器。如果您使用的不是浮动或单用户许可证，请跳过本节。

如果您所使用的是浮动或单用户许可证，使用 Gurobi 优化器之前也须要启动 Gurobi 令牌服务器。下一步取决于您所使用的平台。请按照需要选择参考激活 Windows 令牌服务器或激活 Linux 或 Mac OS 令牌服务器。请注意，AIX 系统目前尚不能作为令牌服务器使用。

激活 Windows 令牌服务器

如果您购买了单用户或浮动许可证（在许可证密钥文件中寻找 *TYPE=TOKEN*），您就须要建立一个 Gurobi 令牌服务器。对于单用户许可证，令牌服务器必须在得到许可的机器上运行。对于浮动许可证，您就须要选择一台机器充当令牌服务器。如果使用 Windows 机器作为您的令牌服务器，请遵循本节中的说明（请注意浮动许可证的客户端不须要为 Windows 机器）。本节的第一部分会说明如何建立令牌服务器。之后将说明如何建立客户端许可证。请注意，浮动许可证的客户端机器不须要运行令牌服务器。它们的许可证仅仅须要指示令牌服务器在何处运行。

在 Windows 系统中，选择“*开始*”菜单中“*Gurobi Optimization*”文件夹下的“*Gurobi Token Server*”菜单项以启动令牌服务。只有安装了 Gurobi 许可证密钥文件之后才可以这么做。

启动 Gurobi 令牌服务器之后，下一步将取决于您的杀毒软件和防火墙的设置。大多数杀毒软件会立即要求您确认是否允许程序 *grbd.exe* 接收网络通信。一旦确认允许，令牌服务器就会开始服务令牌。如果您没有收到这样的提示信息，您就须要将 *grbd.exe* 添加到防火墙的例外名单中。在 Windows Vista 或 Windows 7 中，选择控制面板“*安全*”区域下的“*允许程序通过 Windows 防火墙*”就可以做到这一点。在 Windows XP 中，则须要在控制面板下的“*Windows 防火墙*”中进行选择。您应该将 *grbd.exe* 添加到例外名单中。

令牌服务一旦启动之后，您就可以在任务管理器“*服务*”页面的列表中看到 *grbd* 服务。可以单击“*服务*”页面右下方的“*服务*”按钮，然后在弹出窗口中右键单击“*Gurobi License Manager*”项来启动或停止该服务。

您也可以在命令行中启动或停止 Gurobi License Manager 服务。运行 *grbd –h* 命令会列出所有可用的命令行选项。运行 grbd –s 命令会停止正在运行的许可证服务。而运行 grbd –v 命令则会以详细模式启动许可证服务。详细模式会在每次签入或签出令牌时产生一条日志信息。

请注意如果您是从 Gurobi 优化器较早的版本进行升级，在启动新的令牌服务器之前应该停止旧的服务。同一时间在一台机器上只能运行一个令牌服务器（它们都使用相同的网络端口）。

Gurobi License Manager 的所有输出都会进入到 Windows 事件日志中。在 Windows Vista 或 Windows 7 中可以通过事件查看器来访问它。在“*开始*”菜单下的搜索框中键入“*Event*”来运行查看器。

要建立客户端许可证，就须要在每台使用令牌服务器的机器上都安装 *gurobi.lic* 文件（使用之前介绍的许可证安装说明）。或者您也可以使用较为简单的客户端许可证密钥文件，它仅包含令牌服务器的机器名或 IP 地址：

```
TOKENSERVER=mymachine.mydomain.com
```

或

```
TOKENSERVER=192.168.1.100
```

请注意须要将其替换为您的令牌服务器的机器名或 IP 地址。

您现在可以尝试许可证的测试了。

激活 Linux 或 Mac OS 令牌服务器

如果您购买了单用户或浮动许可证（在许可证密钥文件中寻找 *TYPE=TOKEN*），您就须要建立一个 Gurobi 令牌服务器。对于单用户许可证，令牌服务器必须在得到许可的机器上运行。对于浮动许可证，您就须要选择一台机器充当令牌服务器。如果使用 Linux 或 Mac OS 机器作为您的令牌服务器，请遵循本节中的说明（请注意浮动许可证的客户端不须要为 Linux 或 Mac OS 机器）。本节的第一部分会说明如何建立令牌服务器。之后将说明如何建立客户端许可证。请注意，浮动许可证的客户端机器不须要运行令牌服务器。它们的许可证仅仅须要指示令牌服务器在何处运行。

在 Linux 或 Mac OS 系统中，您须要在许可证服务器计算机上运行 *grbd* 程序（不带参数）来启动令牌服务器后台程序。您只须要运行一次——令牌服务器会持续运行直到被停止（或直到机器关闭）。运行该程序之前请确认许可证密钥文件已经安装。请注意令牌服务器是作为用户进程运行的，所以启动它并不需要 root 权限。

请注意如果您想要在机器重启时重新启动许可证服务器，你就应该要求系统管理员在 */etc/rc.local* 中启动它。如果 Gurobi 的安装文件和许可证密钥文件都存放在默认位置，那么在 Linux 系统中添加以下行就可以了：

```
/opt/gurobi460/linux64/bin/grbd
```

要停止正在运行的令牌服务器，只须简单地杀死 *grbd* 进程即可。可以使用 *ps* 命令来找到相关的进程 ID，并使用 *kill* 命令来终止该进程。

如果您是从 Gurobi 优化器较早的版本进行升级，在启动新的令牌服务器之前应该停止旧的服务。同一时间在一台机器上只能运行一个令牌服务器（它们都使用相同的网络端口）。

令牌服务器的输出会进入到系统日志（*/var/log/syslog*）中。启动服务器时应该可以看到如下所示的信息：

```
Mar  7 12:37:21 mymachine grbd[7917]: Daemon started: Sat Mar  7
12:37:21 2009
```

默认地，令牌服务器只在启动时产生日志输出。要获取更为详细的日志信息，可以使用 –v 开关来启动令牌服务器。在这种情况下，每次签入或签出令牌时都会产生一条日志信息。

要建立客户端许可证，就须要在每台使用令牌服务器的机器上都安装 *gurobi.lic* 文件（使用之前介绍的许可证安装说明）。或者您也可以使用较为简单的客户端许可证密钥文件，它仅包含令牌服务器的机器名或 IP 地址：

```
TOKENSERVER=mymachine.mydomain.com
```

或

```
TOKENSERVER=192.168.1.100
```

请注意须要将其替换为您的令牌服务器的机器名或 IP 地址。

如果访问令牌服务器时出现问题，请检查服务器计算机上运行的防火墙软件（例如 *Bastille* 或 *ipfilter*）是否阻止了对于某些端口的访问。Gurobi 令牌服务器所使用的默认端口号为 41954，所以须要在服务器上打开对于该端口的访问。具体做法请参见防火墙软件的手册。如果默认端口存在冲突，则可以在服务器和客户端许可证密钥文件中添加 *PORT* 行选择不同的端口：

```
PORT=46325
```

可以选择任何可用的端口号。

页面：Interactive Shell

交互式命令解释器

Gurobi 交互式命令解释器允许对优化模型执行实际的交互式操作和实验。可以在交互式命令解释器中读取模型文件、对模型执行完整或局部的优化运行、修改参数、修改模型、并重新优化等等。Gurobi 命令解释器实际上是对 Python 命令解释器的一组扩展。Python 是一种丰富而又灵活的编程语言，任何在 Python 中可用的功能在 Gurobi 命令解释器中同样也是有效的。这里我们会对这些功能进行详细的介绍，但是我们仍然鼓励您进入帮助系统并亲身体验一下命令接口，从而对交互式命令解释器中可以实现的功能有一个更好的理解。

请注意，我们还提供了一组轻量的命令行帮助，使您可以快速地运行一些简单的测试。

AIX 用户的重要注意事项：由于在 AIX 之上的 Python 支持非常有限，我们的 AIX 端口并不包括交互式命令解释器或 Python 接口。您可以使用命令行、C、C++、或 Java 接口。

接下来我们会通过几个简单的示例来演示如何使用 Gurobi 命令解释器，从而简单地介绍其功能。关于 Gurobi 及其接口更为完整的文档请参见 Gurobi 参考手册。

模型的读取与优化

在 Windows 中有三种方法可以启动 Gurobi 交互式命令解释器：

- 双击桌面的 Gurobi 快捷方式。
- 在开始菜单中选择 Gurobi 交互式命令解释器。
- 打开一个 DOS 命令提示窗口并键入 *gurobi.bat*。

在 Linux 或者 Mac OS 中，只须要简单地在命令提示窗口中键入 *gurobi.sh* 即可。优化器一旦启动之后，就可以加载并优化模型。让我们以 *<installdir>/examples/data* 中的模型 *p0033.mps* 为例…

```
> gurobi.bat（在 Linux 或 Mac OS 中为 gurobi.sh)

Gurobi Interactive Shell, Version 4.6.0
Copyright (c) 2011, Gurobi Optimization, Inc.
Type "help()" for help

gurobi> m = read('c:/gurobi460/win64/examples/data/p0033.mps')
Read MPS format model from file
c:/gurobi460/win64/examples/data/p0033.mps
Reading time = 0.00 seconds
P0033: 16 Rows, 33 Columns, 98 NonZeros

gurobi> m.optimize()
Optimize a model with 16 Rows, 33 Columns and 98 NonZeros
```

```
Presolve removed 3 rows and 5 columns
Presolve time: 0.00s
Presolved: 13 Rows, 28 Columns, 91 Nonzeros
Objective GCD is 1
Found heuristic solution: objective 3412.0000000

Root relaxation: objective 2.838547e+03, 24 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0 2838.54674    0    6 3412.00000 2838.54674  16.8%     -
0s
H    0     0                    3347.0000000 2838.54674  15.2%     -
0s
     0     0 2928.65799    0    8 3347.00000 2928.65799  12.5%     -
0s
H    0     0                    3095.0000000 2928.65799  5.37%     -
0s
     0     0 2954.18420    0   15 3095.00000 2954.18420  4.55%     -
0s
     0     0 3008.37500    0   12 3095.00000 3008.37500  2.80%     -
0s
     0     0 3017.50000    0   10 3095.00000 3017.50000  2.50%     -
0s
     0     0 3023.52381    0    6 3095.00000 3023.52381  2.31%     -
0s
     0     0 3023.52381    0    6 3095.00000 3023.52381  2.31%     -
0s
     0     0 3023.52381    0    6 3095.00000 3023.52381  2.31%     -
0s
*    3     0               2    3089.0000000 3089.00000   0.0%   3.3
0s

Cutting planes:
  Cover: 5
  MIR: 4

Explored 4 nodes (74 simplex iterations) in 0.02 seconds
Thread count was 8 (of 8 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 3.0890000000e+03, best bound 3.0890000000e+03, gap 0.0%
```

在上例中，*read()* 命令从文件中读取模型并返回一个 *Model* 对象。在该示例中，所返回的对象存放到变量 *m* 中。在 Python 语言中没有必要声明变量，只须要简单地为变量赋值即可。

请注意，*read()* 接受通配符，所以也可以如下所示键入命令：

```
gurobi> m = read('c:/gurobi460/win64/*/*/p0033*')
```

另外请注意，Gurobi 的文件读写命令对于压缩文件也是同样有效的。如果在机器的默认路径中安装了 *gzip*、*bzip2*、或 *7zip*，那么您只须要在文件名中添加合适的后缀（*.gz*、*.bz2*、*.zip*、或 *.7z*）就可以读写对应的压缩文件了。

示例中的下一条语句 *m.optimize()* 调用了该 *Model* 对象的 *optimize* 方法（键入 *help(Model)* 或 *help(m)* 可以获取 *Model* 对象所有方法的列表）。Gurobi 优化引擎找到了一个最优解，对应的目标值为 3089。

解的查看

一旦求解了模型，就可以通过 *Model* 对象的 *printAttr()* 方法来查看最优解中模型变量的取值：

```
gurobi> m.printAttr('X')

  Variable          X
---------------------
      C157          1
      C163          1
      C164          1
      C166          1
      C170          1
      C174          1
      C177          1
      C178          1
      C180          1
      C181          1
      C182          1
      C183          1
      C184          1
      C185          1
      C186          1
```

该方法会打印指定的属性 *X* 中所有的非零值。属性在 Gurobi 优化器中发挥着重要的作用。我们马上就会对它们进行更为详细的介绍。

调用 *Model* 对象的 *getVars()* 方法（在我们的示例中为 *m.getVars()*）可以检索模型中所有变量的列表，通过这种方式可以更为精细地查看优化结果：

```
gurobi> v = m.getVars()
gurobi> print len(v)
33
```

其中第一条命令将模型 *m* 中所有的 *Var* 对象列表赋给变量 *v*。Python 的 *len()* 命令则返回该数组的长度（我们的示例模型 *p0033* 中有 33 个变量）。之后就可以查询列表中单个变量的不同属性。例如，使用以下命令来获取列表 *v* 中第一个变量的变量名和解决方案的值：

```
gurobi> print v[0].VarName, v[0].X
C157 1.0
```

可以键入 *help(Var)* 或 *help(v[0])* 来获取 *Var* 对象所有方法的列表。键入 *help(GRB.Attr)* 则会返回所有属性的列表。

简单的模型修改

我们现在来演示如何进行简单的模型修改。假设您只希望关注变量 *C157* 取 0.0 值时的解决方案，那么将第一个变量的上界值限制为 0.0 就能够找到一个满足额外约束的解。设置该变量的 *UB* 属性就可以做到…

```
gurobi> v = m.getVars()
gurobi> v[0].UB = 0
```

Gurobi 优化器会跟踪模型的状态，所以它知道当前所加载的解决方案对于修改以后的模型来说并不一定就是最优的。当再次调用 ***optimize()*** 方法时，Gurobi 优化器就会对修改的模型进行一次新的优化…

```
gurobi> m.optimize()
Optimize a model with 16 Rows, 33 Columns and 98 NonZeros
Presolve removed 3 rows and 6 columns
Presolve time: 0.00s
Presolved: 13 Rows, 27 Columns, 88 Nonzeros
Objective GCD is 1

Root relaxation: objective 2.839492e+03, 21 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0 2839.49184    0    5          - 2839.49184     -      -
0s
     0     0 2953.32500    0    2          - 2953.32500     -      -
0s
     0     0 2994.81212    0   11          - 2994.81212     -      -
0s
     0     0 3060.38462    0    2          - 3060.38462     -      -
0s
H    0     0                   3095.0000000 3060.38462  1.12%     -
0s
     0     0 infeasible    0      3095.00000 3094.00310  0.03%     -
0s

Cutting planes:
  Gomory: 1
  Cover: 5
  MIR: 5

Explored 0 nodes (46 simplex iterations) in 0.01 seconds
Thread count was 8 (of 8 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 3.0950000000e+03, best bound 3.0950000000e+03, gap 0.0%
```

当强制要求变量 *C157* 取零值时，模型可能取的最小目标值从原来的 3089 增加到 3095。一个简单的检查就可以确认新的上界约束是满足的：

```
gurobi> print v[0].X
```

```
0.0
```

对于较难计算的模型进行简单的实验

现在让我们来考虑一个较难计算的模型：*glass4.mps*。同样地，我们读取模型并开始优化：

```
gurobi> m = read('c:/gurobi460/win64/examples/data/glass4')
Read MPS format model from file
c:/gurobi460/win64/examples/data/glass4.mps
Reading time = 0.00 seconds
glass4: 396 Rows, 322 Columns, 1815 NonZeros
gurobi> m.optimize()
Optimize a model with 396 Rows, 322 Columns and 1815 NonZeros
Presolve removed 4 rows and 5 columns
Presolve time: 0.00s
Presolved: 392 Rows, 317 Columns, 1815 Nonzeros
Found heuristic solution: objective 3.691696e+09

Root relaxation: objective 8.000024e+08, 72 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0 8.0000e+08    0   72 3.6917e+09 8.0000e+08  78.3%     -
0s
     0     0 8.0000e+08    0   72 3.6917e+09 8.0000e+08  78.3%     -
0s
     0     0 8.0000e+08    0   72 3.6917e+09 8.0000e+08  78.3%     -
0s
     0     0 8.0000e+08    0   72 3.6917e+09 8.0000e+08  78.3%     -
0s
     0     2 8.0000e+08    0   72 3.6917e+09 8.0000e+08  78.3%     -
0s
H  769   732                    2.800024e+09 8.0000e+08  71.4%   5.2
0s
H  834   781                    2.666693e+09 8.0000e+08  70.0%   5.3
0s
H 1091   984                    2.475023e+09 8.0000e+08  67.7%   5.1
0s
H 1092   986                    2.400020e+09 8.0000e+08  66.7%   5.1
0s
H 1092   984                    2.380021e+09 8.0000e+08  66.4%   5.1
0s
H 1095   988                    2.350020e+09 8.0000e+08  66.0%   5.1
0s
* 1845  1543              94    2.316685e+09 8.0000e+08  65.5%   4.9
0s
* 2131  1627             126    2.150018e+09 8.0000e+08  62.8%   4.8
0s
H 2244  1580                    2.100019e+09 8.0000e+08  61.9%   4.8
0s
H 2248  1341                    1.900018e+09 8.0000e+08  57.9%   5.0
0s
```

```
H 3345  1816                    1.900018e+09 8.0000e+08  57.9%   4.1
0s
H 3346  1744                    1.900017e+09 8.0000e+08  57.9%   4.1
0s
H15979 10383                    1.900017e+09 8.0000e+08  57.9%   2.5
1s
H19540 13051                    1.900016e+09 8.0000e+08  57.9%   2.4
1s
*21124 13489             101    1.866683e+09 8.0000e+08  57.1%   2.4
1s
*23011 14690             100    1.850015e+09 8.0000e+08  56.8%   2.3
1s
*25630 15679             143    1.800016e+09 8.0000e+08  55.6%   2.3
1s
*28365 15421             113    1.700015e+09 8.0000e+08  52.9%   2.3
1s
H29910 16333                    1.700014e+09 8.0000e+08  52.9%   2.3
1s
*30582 16765             124    1.700014e+09 8.0000e+08  52.9%   2.3
1s
*33238 16251              92    1.677794e+09 8.0000e+08  52.3%   2.3
1s
*37319 18258              85    1.633349e+09 8.0000e+08  51.0%   2.2
1s
H40623 19584                    1.600015e+09 8.0000e+08  50.0%   2.3
2s
 81781 42951 1.1000e+09   49   51 1.6000e+09 8.0001e+08  50.0%   2.2
5s
 199990 100088 1.6000e+09   82   28 1.6000e+09 8.0001e+08  50.0%   2.3
10s
*242810 116891              97    1.600015e+09 8.2001e+08  48.8%   2.3
11s
*243703 116786              95    1.600014e+09 8.2001e+08  48.8%   2.3
11s

Interrupt request received

Explored 255558 nodes (588336 simplex iterations) in 12.36 seconds
Thread count was 8 (of 8 available processors)

Solve interrupted
Best objective 1.6000142000e+09, best bound 8.5000490000e+08, gap
46.8752%
```

在上例中十分明显就可以看到，这个模型与 *p0033* 相比计算起来要困难许多。实际上最优解是 *1,200,000,000*，但是优化器须要花费很长时间才能找到这个解。让模型运行了 10 秒之后，我们中止了运行（通过按下 CTRL-C 组合键生成“收到中断请求”消息）并考虑我们的选择。键入 *m.optimize()* 命令会使模型从中止的地方开始继续运行。

参数的修改

有时尝试不同的参数设置会比继续优化像 *glass4* 这样难解的模型更加有效。当下界值移动缓慢时（正如这个模型中那样），一个可能有用的参数就是 *MIPFocus*，该参数用于调节高

级 MIP 求解策略。现在让我们将这个参数的值设置为 1，从而使 MIP 搜索的关注点集中到寻找好的可行解上。有两种方法可以修改参数值。可以使用 *m.setParam()* 方法：

```
gurobi> m.setParam('MIPFocus', 1)
Changed value of parameter MIPFocus to 1
   Prev: 0   Min: 0   Max: 3   Default: 0
```

…或者也可以使用 *m.Params* 类…

```
gurobi> m.Params.MIPFocus = 1
Changed value of parameter MIPFocus to 1
   Prev: 0   Min: 0   Max: 3   Default: 0
```

参数一旦修改之后，我们调用 *m.reset()* 命令来重置模型的优化状态，然后调用 *m.optimize()* 重新开始进行新的优化：

```
gurobi> m.reset()
gurobi> m.optimize()
Optimize a model with 396 Rows, 322 Columns and 1815 NonZeros
Presolve removed 4 rows and 5 columns
Presolve time: 0.00s
Presolved: 392 Rows, 317 Columns, 1815 Nonzeros
Found heuristic solution: objective 3.691696e+09

Root relaxation: objective 8.000024e+08, 72 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0 8.0000e+08    0   72 3.6917e+09 8.0000e+08  78.3%     -
0s
     0     0 8.0000e+08    0   72 3.6917e+09 8.0000e+08  78.3%     -
0s
     0     0 8.0000e+08    0   72 3.6917e+09 8.0000e+08  78.3%     -
0s
     0     0 8.0000e+08    0   73 3.6917e+09 8.0000e+08  78.3%     -
0s
H    0     0                    3.075022e+09 8.0000e+08  74.0%     -
0s
H    0     0                    3.020023e+09 8.0000e+08  73.5%     -
0s
     0     0 8.0000e+08    0   76 3.0200e+09 8.0000e+08  73.5%     -
0s
     0     0 8.0000e+08    0   75 3.0200e+09 8.0000e+08  73.5%     -
0s
H    0     0                    2.550024e+09 8.0000e+08  68.6%     -
0s
H    0     2                    2.175020e+09 8.0000e+08  63.2%     -
0s
     0     2 8.0000e+08    0   75 2.1750e+09 8.0000e+08  63.2%     -
0s
H   95    98                    2.150020e+09 8.0000e+08  62.8%   4.6
0s
H   96    98                    2.120018e+09 8.0000e+08  62.3%   4.6
0s
```

```
H  101   103                    2.116687e+09 8.0000e+08  62.2%   4.5
0s
H  110   103                    2.100017e+09 8.0000e+08  61.9%   4.3
0s
H  352   325                    2.000018e+09 8.0000e+08  60.0%   4.2
0s
H  406   375                    1.991686e+09 8.0000e+08  59.8%   4.0
0s
H 1074   888                    1.981836e+09 8.0000e+08  59.6%   3.5
0s
H 1078   889                    1.966686e+09 8.0000e+08  59.3%   3.5
0s
H 1107   878                    1.900018e+09 8.0000e+08  57.9%   3.5
0s
H 1696  1125                    1.800017e+09 8.0000e+08  55.6%   3.4
0s
H 1845  1146                    1.800017e+09 8.0000e+08  55.6%   4.2
1s
H 1863  1087                    1.733350e+09 8.0000e+08  53.8%   4.3
1s
H 2353  1273                    1.733350e+09 8.0000e+08  53.8%   4.3
1s
H 2517  1299                    1.700016e+09 8.0000e+08  52.9%   4.3
1s
H 2598  1248                    1.666682e+09 8.0000e+08  52.0%   4.3
1s
H 2733  1252                    1.633349e+09 8.0000e+08  51.0%   4.2
1s
 14259  7927 1.5000e+09   85   28 1.6333e+09 8.0000e+08  51.0%   3.5
5s
 24846 14278 1.1000e+09   49   55 1.6333e+09 8.0001e+08  51.0%   3.5
10s
H25035 13985                    1.600016e+09 8.0001e+08  50.0%   3.5
10s
H25066 14020                    1.600016e+09 8.0001e+08  50.0%   3.5
10s
H25072 13532                    1.583350e+09 8.0001e+08  49.5%   3.5
10s
H26218 14083                    1.575016e+09 8.0001e+08  49.2%   3.5
10s
H26326 14118                    1.566682e+09 8.0001e+08  48.9%   3.5
10s
H26577 13650                    1.525016e+09 8.0001e+08  47.5%   3.5
10s

Interrupt request received

Cutting planes:
  Gomory: 6
  Implied bound: 26
  MIR: 60

Explored 30546 nodes (107810 simplex iterations) in 11.81 seconds
Thread count was 8 (of 8 available processors)

Solve interrupted
```

```
Best objective 1.5250155750e+09, best bound 8.0000520000e+08, gap
47.5412%
```

优化结果与我们的期望相一致。将关注点切换到寻找可行解之后不久我们就找到了一个更优的解（目标值从 *1.6e9* 变为 *1.525e9*）。

setParam() 被设计为非常灵活的方法，并且其容错性也很强。它可以接受通配符参数，并且忽略字符的大小写。因此，以下所有命令都是等同的：

```
gurobi> m.setParam('NODELIMIT', 100)
gurobi> m.setParam('NodeLimit', 100)
gurobi> m.setParam('Node*', 100)
gurobi> m.setParam('N???Limit, 100)
```

可以使用通配符来获取一组相匹配的参数：

```
gurobi> m.setParam('*Cuts', 2)
Matching parameters: ['Cuts', 'CliqueCuts', 'CoverCuts',
'FlowCoverCuts', 'FlowP
athCuts', 'GUBCoverCuts', 'ImpliedCuts', 'MIPSepCuts', 'MIRCuts',
'ModKCuts', 'NetworkCuts', 'SubMIPCuts', 'ZeroHalfCuts']
```

请注意，*Model.Params* 的容错性并没有 *setParam()* 那么好。特别是该方法不接受通配符；但是在两种方法中都不须要关注参数名称的大小写。所以，*m.Params.Heuristics* 和 *m.Params.heuristics* 是等同的。

可以通过 *paramHelp()* 命令来浏览所有可用参数的完整列表。而键入 *paramHelp('MIPGap')* 则可以获取某个参数（例如，*MIPGap*）更为详细的信息。

使用多个模型

Gurobi 命令解释器允许您同时使用多个模型。例如…

```
gurobi> a = read('c:/gurobi460/win64/examples/data/p0033')
Read MPS format model from file
c:/gurobi460/win64/examples/data/p0033.mps
Reading time = 0.00 seconds
P0033: 16 Rows, 33 Columns, 98 NonZeros.
gurobi> b = read('c:/gurobi460/win64/examples/data/stein9')
Read MPS format model from file
c:/gurobi460/win64/examples/data/stein9.mps
Reading time = 0.00 seconds
STEIN9: 13 Rows, 9 Columns, 45 NonZeros.
```

models() 命令会返回所有有效的模型列表。

```
gurobi> models()
Currently loaded models:
a : <gurobi.Model MIP instance P0033: 16 constrs, 33 vars>
b : <gurobi.Model MIP instance STEIN9: 13 constrs, 9 vars>
```

请注意，可以通过 *Model.setParam()* 方法或 *Model.Params* 类来配置模型中的某个参数，也可以调用全局的 *setParam()* 方法来修改 Gurobi 命令解释器中所有模型的参数。

帮助

交互式命令解释器中包含了非常丰富的帮助功能。只须要在提示窗口中简单地键入 *help()* 命令就可以访问这些信息。如之前所述，接口中所有的重要对象都存在详细的帮助。例如，在帮助功能中提到过，可以键入 *help(Model)*、*help(Var)*、或 *help(Constr)* 来获得不同对象的帮助文档。此外，也可以获得这些对象中任何一个可用方法的详细帮助。例如，*help(Model.setParam)* 会帮助提示如何设置模型参数。您也可以通过某个变量、或者该变量的某个方法来获取帮助。例如，如果变量 *m* 包含了一个模型对象，那么 *help(m)* 就等同于 *help(Model)*；而 *help(m.setParam)* 则等同于 *help(Model.setParam)*。

接口的自定义

Gurobi 交互式命令解释器存在于功能齐全的脚本语言中。这就使得您可以执行各式各样的接口自定义来满足自己特定的需求。创建自定义函数需要一些 Python 语言的知识，但是使用非常有限的语言特性就可以实现很多的自定义功能。

让我们来看一个简单的示例。假设模型文件存放在磁盘上的某个目录中。您可以创建一个自定义的 *read* 方法，而不须要在每次读取模型时都键入完整的文件路径：

```
gurobi> def myread(filename):
.......   return read('/home/john/models/'+filename)
```

请注意，第二行的行首缩进是必需的。

定义这一函数使您可以执行以下命令：

```
gurobi> m = myread('stein9')
Read MPS format model from file /home/john/models/stein9.mps
```

如果不希望在每次启动 Gurobi 命令解释器时都定义一遍该函数，则可以将其存储在文件中。函数文件的定义如下所示：

```
from gurobipy import *

def myread(filename):
  return read('/home/john/models/'+filename)
```

为了能够在 Gurobi 命令解释器中使用 *read* 方法，就必须在自定义函数中包含“*from gurobipy import **”这一行。自定义文件必须以“*.py*”为后缀名。如果将文件命名为 *custom.py*，之后就可以键入：

```
gurobi> from custom import *
```

来导入该函数。一个文件中可以包含您想要的任何数量的自定义函数（请参见 *<installdir>/examples/python* 目录中的 *custom.py* 示例）。如果想要进行全站式的自定义，您也可以修改包含在 *<installdir>/lib* 目录中的 *gurobi.py* 自定义文件。

通过回调实现的自定义

接下来我们简要地介绍一下另一种自定义方式，它可以通过 Gurobi 的回调来实现。通过回调可以跟踪优化过程的进度。举个例子来说，假设您想要让 MIP 优化器运行 10 秒才退出，但又不希望在找到可行解之前就终止优化。下面的回调方法可以满足这一需求：

```
from gurobipy import *

def mycallback(model, where):
  if where == GRB.Callback.MIP:
    time = model.cbGet(GRB.Callback.RUNTIME)
    best = model.cbGet(GRB.Callback.MIP_OBJBST)
    if time > 10 and best < GRB.INFINITY:
      model.terminate()
```

一旦导入了这个函数（“*from custom import **”），就可以调用 *m.optimize(mycallback)* 命令来达到想要的终止行为。或者，您也可以自定义一个优化方法，使它始终调用这一回调函数：

```
def myopt(model):
  model.optimize(mycallback)
```

这样您就可以调用 *myopt(m)* 命令了。

通过模型对象可以将任意数据传入回调函数。例如，如果在调用 *optimize()* 之前将 *m._mydata* 设置为 *1*，那么在回调函数中就可以查询 *m._mydata*。请注意用户数据字段的名称必须以下划线开始。

这个回调示例包含在 *<installdir>/examples/python/custom.py* 中。键入“*from custom import **”来导入自定义的回调和 *myopt()* 函数。

可以键入 *help(GRB.Callback)* 来获得更多关于回调的信息。您也可以参考 Gurobi 参考手册中 *Callback* 类的文档。

Gurobi 的 Python 用户接口

如果您是 Python 用户，并且想要在自己的 Python 环境中使用 Gurobi，那么您就可以直接在环境中安装 *gurobipy* 模块。在您的 *<installdir>* 下调用以下命令即可：

```
  python setup.py install
```

在 Linux 或 Mac OS 系统中，您须要作为超级用户来运行这条命令，除非您所使用的是您自己私有的 Python 安装。*gurobipy* 一旦成功安装，您就可以在您的 Python 命令解释器中键入“*import gurobipy*”或“*from gurobipy import **”并访问所有的 Gurobi 类和方法。

请注意，为了使安装成功进行，您的 Python 环境必须和 Gurobi 的 Python 模块想匹配。32-位的 Gurobi 库只应该安装到 32-位的 Python 命令解释器中（对于 64-位的版本也是一样）。另外，Python 的版本必须相互匹配。在此版本中，对于 Windows 和 Linux 系统，*gurobipy* 可以和 Python 2.5、2.6、或 2.7 一起使用；而在 Mac OS 上则只能和 Python 2.6 一起使用。

简单的命令行应用

如果您仅仅想要优化一个存储在文件中的模型，那么就有可能不须要使用交互式接口中的很多功能。我们还提供了一个简单的基于命令行的可执行文件——*gurobi_cl*，用于进行快速测试。这个程序允许修改 Gurobi 参数并将问题的解保存到文件中。

参数修改可以通过两种方法来指定。第一种方法是在命令行中包含一条或多条的 *ParamName=NewValue* 命令。*gurobi_cl* 的最后一个参数始终应该是文件名称。例如，使用单线程求解模型 *stein9.mps* 并将解存放到文件 *stein9.sol* 中，可以使用以下命令：

```
  gurobi_cl Threads=1 ResultFile=stein9.sol stein9.mps.gz
```

第二种方法是将参数修改存放到当前工作目录中的文件 *gurobi.env* 中。执行优化之前，*gurobi_cl* 会查找该文件并从其中读取所有的参数修改。每个参数设置保存为一行，以参数名称开始，后面跟上至少一个空格，再后面是所需的值。任何以“#”号开始的行都是注释行，其内容将被忽略。举个例子，下面的（Linux）命令：

```
  echo "Threads 1" > gurobi.env
  gurobi_cl stein9.mps.gz
```

会从文件 *gurobi.env* 中读取 *Threads* 参数的新值，然后使用单线程来优化模型 *stein9.mps*。请注意，如果一个参数既在命令行中修改又在 *gurobi.env* 文件中指定，则会使用命令行中所指定的值。

Gurobi 发布中有一个 *gurobi.env* 的样本文件（在 *bin* 目录中）。该样本文件包含了每一个参数以及与之相对应的默认值，但所有的设置都被注释掉了。

页面：Attributes

属性

如之前章节中所述，大多数与 Gurobi 模型相关联的信息都存储在一组*属性*之中。一些属性与模型中的变量相关联、一些与模型中的约束相关联、而另一些则与模型本身相关联。对模型进行优化之后，举例来说，解决方案就存储在 *X* 变量属性之中。用户不能直接修改由 Gurobi 优化器计算得到的属性（例如解属性）；而那些表示输入数据的属性（例如用于存储变量下界值的 *LB* 属性），则可以直接修改。

每一个 Gurobi 语言接口中都包含了用于查询和修改属性值的方法。要检索或修改某个特定的属性值，只须要简单地将该属性的名称传入对应的查询或修改方法。例如，在 C 接口中，使用以下代码来查询变量 1 当前解的值：

```
double x1;
error = GRBgetdblattrelement(model, GRB_DBL_ATTR_X, 1, &x1);
```

该方法会从包含了 double 精度数据的数组属性中返回单个元素的值。接口中所提供的方法可以用于查询并修改 *int*、*double*、*char*、或 *char ** 类型标量和数组属性的值。

在面向对象接口中，可以通过合适的对象来查询或修改属性的值。举例来说，如果变量 *v* 是一个 Gurobi 的变量对象（*GRBVar*），那么下面的方法调用就可以用于修改 *v* 的下界值：

```
C++:    v.set(GRB_DoubleAttr_LB, 0.0);
Java:   v.set(GRB.DoubleAttr.LB, 0.0);
C#:     v.Set(GRB.DoubleAttr.LB, 0.0);
Python: v.LB = 0.0
```

不同语言中用于查询和修改属性的具体语法略有不同，但是基本做法都是一致的：调用合适的查询或修改方法，并将须要访问的属性名称作为参数传入。

Gurobi 属性的完整列表请参见 Gurobi 参考手册的*属性*节。

页面：C Interface

C 接口

本节通过一个简单的 C 程序示例来演示如何使用 Gurobi 的 C 接口。该示例会生成一个简单的混合整数规划模型、对它进行优化、并输出最优的目标值。本节须要您对于 C 编程语言有一定的了解。如果您对于 C 语言编程还不是非常熟悉，可以先参考一些相关的书籍资料进行语言学习（例如，*The C Programming Language*， Kernighan 和 Ritchie 著）。

我们的示例须要优化下面的模型：

$$
\begin{array}{rl}
\textbf{maximize} & x + y + 2z \\
\textbf{subject to} & x + 2y + 3z \leq 4 \\
 & x + y \geq 1 \\
 & x, y, z \text{ binary}
\end{array}
$$

示例 mip1_c.c

我们这个示例的完整源代码如下所示（也可以参见 *<installdir>/examples/c/mip1_c.c*）…

```
#include <stdlib.h>
#include <stdio.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0;
  double    sol[3];
  int       ind[3];
  double    val[3];
  double    obj[3];
  char      vtype[3];
  int       optimstatus;
  double    objval;
  int       zero = 0;

  /* Create environment */

  error = GRBloadenv(&env, "mip1.log");
  if (error || env == NULL) {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create an empty model */
  error = GRBnewmodel(env, &model, "mip1", 0, NULL, NULL, NULL, NULL,
NULL);
  if (error) goto QUIT;


  /* Add variables */

  obj[0] = -1; obj[1] = -1; obj[2] = -2;
  vtype[0] = GRB_BINARY; vtype[1] = GRB_BINARY; vtype[2] = GRB_BINARY;
  error = GRBaddvars(model, 3, 0, NULL, NULL, NULL, obj, NULL, NULL,
vtype,
                     NULL);
  if (error) goto QUIT;

  /* Integrate new variables */

  error = GRBupdatemodel(model);
  if (error) goto QUIT;


  /* First constraint: x + 2 y + 3 z <= 4 */

  ind[0] = 0; ind[1] = 1; ind[2] = 2;
  val[0] = 1; val[1] = 2; val[2] = 3;

  error = GRBaddconstr(model, 3, ind, val, GRB_LESS_EQUAL, 4.0, NULL);
  if (error) goto QUIT;

  /* Second constraint: x + y >= 1 */

  ind[0] = 0; ind[1] = 1;
  val[0] = 1; val[1] = 1;

  error = GRBaddconstr(model, 2, ind, val, GRB_GREATER_EQUAL, 1.0,
NULL);
  if (error) goto QUIT;

  /* Optimize model */

  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Write model to 'mip1.lp' */

  error = GRBwrite(model, "mip1.lp");
  if (error) goto QUIT;

  /* Capture solution information */

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;

  error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
  if (error) goto QUIT;

  error = GRBgetdblattrarray(model, GRB_DBL_ATTR_X, 0, 3, sol);
  if (error) goto QUIT;
  printf("\nOptimization complete\n");
  if (optimstatus == GRB_OPTIMAL) {
    printf("Optimal objective: %.4e\n", objval);

    printf("  x=%.0f, y=%.0f, z=%.0f\n", sol[0], sol[1], sol[2]);
  } else if (optimstatus == GRB_INF_OR_UNBD) {
    printf("Model is infeasible or unbounded\n");
  } else {
    printf("Optimization was stopped early\n");
  }

QUIT:

  /* Error reporting */

  if (error) {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

示例细解

让我们一行一行来浏览一遍示例代码，从而理解它是如何达到指定模型所需的优化结果的。

示例代码的一开始指定了一些须要包含的头文件。Gurobi C 应用程序的一开始总是应该包含头文件 *gurobi_c.h*、以及其他的标准 C 头文件（*stdlib.h* 和 *stdio.h*）。

环境的创建

声明了必需的程序变量之后，示例代码继续创建了一个环境：

```
  error = GRBloadenv(&env, "mip1.log");
  if (error || env == NULL) {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }
```

程序之后在试图创建优化模型时总是需要一个环境，所以环境的创建始终应该是使用 Gurobi 优化器时的第一个步骤。*GRBloadenv()* 中的第二个参数指定了 Gurobi 日志文件的名称。如果该参数是一个空字符串或者 NULL，则不会写入任何日志文件。

请注意，环境的创建可能会失败，所以应该检查函数调用的返回值。

模型的创建

一旦创建了环境，下一步就可以创建模型了。一个 Gurobi 模型中保存了单个优化问题。它包含一组变量、一组约束、以及相关联的属性（变量边界、目标系数、变量整性类型、约束含义、约束右侧值，等等）。生成一个包含所有这些信息的模型可以从创建一个空的模型对象开始：

```
  /* Create an empty model */
  error = GRBnewmodel(env, &model, "mip1", 0, NULL, NULL, NULL, NULL,
NULL);
  if (error) goto QUIT;
```

GRBnewmodel() 的第一个参数是之前所创建的环境。第二个参数是一个指针，指向新创建的模型指针应该存放的位置。第三个参数是模型的名称。第四个参数是一开始就加入模型的变量个数。我们创建的是一个空的模型，所以初始变量的个数应该是 0。剩下的参数如果存在则用于描述这些初始变量（下界值、上界值、变量类型，等等）。

在模型中添加变量

一旦创建了 Gurobi 模型，我们就可以开始在其中添加变量和约束了。在我们的示例中，我们从添加变量开始：

```
  /* Add variables */
  obj[0] = -1; obj[1] = -1; obj[2] = -2;
  vtype[0] = GRB_BINARY; vtype[1] = GRB_BINARY; vtype[2] = GRB_BINARY;
  error = GRBaddvars(model, 3, 0, NULL, NULL, NULL, obj, NULL, NULL,
vtype,
                     NULL);
```

GRBaddvars() 的第一个参数是要在其中添加变量的模型。第二个参数是要添加的变量个数（在我们的示例中为 3）。

第三个参数到第六个参数用于描述与这些新变量相关联的约束矩阵系数。第三个参数给出了与新变量相关联的非零约束矩阵项的个数，而接下来的三个参数则提供了这些非零值的详细信息。在我们的示例中，我们会在添加约束时才加入这些非零值。因此，这里的非零值个数是 0，而后面的三个参数都是 NULL。

GRBaddvars() 的第七个参数是每个新变量的线性目标系数。默认地目标含义是最小化，所以我们在这里对最大化目标系数进行取反。我们也可以不去修改这些最大化系数，而将目标含义变为最大化（通过修改模型的 *ModelSense* 属性）。

接下来的两个参数分别用于指定变量的下界值和上界值。NULL 表示这些变量应该取其默认值（对于二元变量为 0.0 和 1.0）。

第十个参数指定了变量的类型。在这一示例中，变量都是二元的（*GRB_BINARY*）。

最后一个参数给出了变量的名称。在该示例中，我们让变量名称都取其默认值（*X0*、*X1*、和 *X2*）。

模型的更新——懒惰修改

Gurobi 优化器中对于模型的修改是以一种*懒惰*方式来执行的，这意味着修改的效果并非马上就可以看到。这一更新方式使得执行一系列的模型修改变得更为容易，因为模型在修改的过程中并不会真正改变。然而，懒惰的修改方式要求您在需要的时候手动地整合模型的修改。这可以通过以下函数来实现：

```
  /* Integrate new variables */
  error = GRBupdatemodel(model);
  if (error) goto QUIT;
```

在我们的示例中，调用 *GRBupdatemodel()* 之前模型中不包含任何变量；而在调用该函数之后则有三个。如果不首先调用该函数，程序之后在试图添加约束时就会发生错误，因为模型中并没有变量。

在模型中添加约束

一旦将新变量整合到模型之中，下一步就可以添加我们的两个约束了。可以通过 *GRBaddconstr()* 函数来添加约束。添加约束时必须指定一些信息，包括与约束相关联的非零值、约束含义、右侧值、以及约束名称。这些信息都是作为 *GRBaddconstr()* 函数的参数来指定的：

```
  /* First constraint: x + 2 y + 3 z <= 4 */

  ind[0] = 0; ind[1] = 1; ind[2] = 2;
  val[0] = 1; val[1] = 2; val[2] = 3;

  error = GRBaddconstr(model, 3, ind, val, GRB_LESS_EQUAL, 4.0, NULL);
  if (error) goto QUIT;
```

GRBaddconstr() 的第一个参数是要在其中添加约束的模型。第二个参数是与新约束相关联的非零系数的总数。接下来两个参数描述了新约束中的非零值。约束系数是通过一组索引-值对来指定的，每一个非零值都对应一个索引-值对。在我们的示例中，第一个要添加的约束是 x + 2y + 3z ≤ 4。我们已经选择将 x 作为我们约束矩阵中的第一个变量、y 作为第二个、z 则作为第三个（请注意这一选择是任意的）。对应于我们的变量顺序选择，第一个约束所需要的索引-值对为 (0, 1.0)、(1, 2.0)、和 (2, 3.0)。这些数据对存放在 *ind* 和 *val* 数组中。

GRBaddconstr() 的第五个参数用于指定新约束的含义。可能的取值为 *GRB_LESS_EQUAL*、*GRB_GREATER_EQUAL*、或 *GRB_EQUAL*。第六个参数给出了右侧值。而最后一个参数则给出了约束的名称（这里我们通过将参数指定为 *NULL* 让约束名称取其默认值）。

第二个约束以相似的方式添加：

```
 /* Second constraint: x + y >= 1 */
```

```
  ind[0] = 0; ind[1] = 1;
  val[0] = 1; val[1] = 1;

  error = GRBaddconstr(model, 2, ind, val, GRB_GREATER_EQUAL, 1.0,
NULL);
  if (error) goto QUIT;
```

请注意，*GRBaddconstrs()*函数允许您在单次调用中同时添加多个约束。然而，这一函数的参数相比起来复杂许多，但又没有提供任何其他显著的优势。所以我们建议您每次只添加一个约束。

模型的优化

现在模型已经生成了，下一步就是对其进行优化：

```
  error = GRBoptimize(model);
  if (error) goto QUIT;
```

该函数会对模型执行优化并填充一些内部模型属性，包括优化状态、解决方案，等等。该函数一旦返回，我们就可以查询这些属性的值。特别地，我们可以通过检索 *Status* 属性的值来查询优化过程的状态…

```
  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;
```

优化状态可能取很多不同的值。比如找到了模型的一个最优解，或者已经确定模型是不可行或无边界的、也有可能求解的过程被打断了。可能的状态值列表请参见 Gurobi 参考手册。在我们的示例中，我们知道模型是可行的，并且我们没有修改任何可能导致优化提前停止的参数（例如，时间限制），所以优化状态将会是 *GRB_OPTIMAL*。

另一个重要的模型属性就是目标函数对于求得的解所取的值。目标函数的值可以通过下面的函数调用来访问：

```
  error = GRBgetdoubleattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
  if (error) goto QUIT;
```

请注意，如果优化器没有找到该模型的解，此函数就会返回一个非零的错误结果。

我们一旦知道优化器已经求解了模型，我们就可以提取该模型的 *X* 属性。该属性包含了求得的解中每个变量的取值：

```
  error = GRBgetdoublearrayattr(model, GRB_DBL_ATTR_X, 0, 3, x);
  if (error) goto QUIT;
  printf("  x=%.0f, y=%.0f, z=%.0f", x[0], x[1], x[2]);
```

该方法用于检索数组类型属性的值。其中的第三个和第四个参数分别指定要检索的第一个数组元素的索引、以及要检索的数组元素的个数。在本示例中，我们检索从第 0 项到第 2 项的数据（也就是所有三个元素）。

错误报告

我们想要指出该示例中的另外一个方面。几乎所有的 Gurobi 方法都会返回一个错误代码。错误代码通常为零，表示没有遇到错误，但是万一错误发生时检查错误代码的值是十分重要的。

尽管您也许想要在可能发生错误的每一个地方都打印专门的错误代码，但 Gurobi 接口却提供了更为灵活的错误报告功能。*GRBgeterrormsg()* 函数会返回与某个环境相关联的最近发生的那次错误的文本描述：

```
if (error) {
  printf("ERROR: %s\n", GRBgeterrormsg(env));
  exit(1);
}
```

错误报告一旦完成以后，示例中惟一剩下的任务就是释放与我们的优化任务相关联的资源。在这个示例中，我们填充了一个模型并且创建了一个环境。我们调用 *GRBfreemodel(model)* 来释放模型，并调用 *GRBfreeenv(env)* 来释放环境。

示例的生成与运行

要生成并运行该示例，请参见 *<installdir>/examples/build* 中的文件。对于 Windows 平台，该目录中包含了 *C_examples_2008.sln* 和 *C_examples_2010.sln*（C 示例的 Visual Studio 2008 和 2010 解决方案文件）。双击解决方案文件会启动 Visual Studio。单击“mip1_c”项目，然后在“*Build*”菜单中选择“*Run*”，Visual Studio 会编译并运行该示例。对于 Linux 或 Mac OS 平台，*<installdir>/examples/build* 目录中包含了该示例的 Makefile。键入“*make mip1_c*”会生成并运行该示例。

C 的示例目录 *<installdir>/examples/c* 中包含了许多示例。我们鼓励您浏览并修改它们，从而对 Gurobi 的 C 接口更加熟悉。我们同时也鼓励您阅读 Gurobi 示例导览来获得更多的信息。

页面：C++ Interface

C++ 接口

本节通过一个简单的 C++ 程序示例来演示如何使用 Gurobi 的 C++ 接口。该示例会生成一个模型、对它进行优化、并输出最优的目标值。本节须要您对于 C++ 编程语言有一定的了解。如果您对于 C++ 语言编程还不是非常熟悉，可以先参考一些相关的书籍资料进行语言学习（例如，*The C++ Programming Language*，Stroustrup 著）。

我们的示例须要优化下面的模型：

$$\begin{array}{rl} \textbf{maximize} & x + y + 2z \\ \textbf{subject to} & x + 2y + 3z \le 4 \\ & x + y \ge 1 \\ & x, y, z \text{ binary} \end{array}$$

示例 mip1_c++.cpp

我们这个示例的完整源代码如下所示（也可以参见 *<installdir>/examples/c++/mip1_c++.cpp*）…

```
#include "gurobi_c++.h"
using namespace std;

int
main(int   argc,
     char *argv[])
{
  try {
    GRBEnv env = GRBEnv();

    GRBModel model = GRBModel(env);

    // Create variables

    GRBVar x = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "x");
    GRBVar y = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "y");
    GRBVar z = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "z");

    // Integrate new variables

    model.update();

    // Set objective: maximize x + y + 2 z

    model.setObjective(x + y + 2 * z, GRB_MAXIMIZE);

    // Add constraint: x + 2 y + 3 z <= 4

    model.addConstr(x + 2 * y + 3 * z <= 4, "c0");
    // Add constraint: x + y >= 1

    model.addConstr(x + y >= 1, "c1");

    // Optimize model

    model.optimize();

    cout << x.get(GRB_StringAttr_VarName) << " "
         << x.get(GRB_DoubleAttr_X) << endl;
    cout << y.get(GRB_StringAttr_VarName) << " "
         << y.get(GRB_DoubleAttr_X) << endl;
    cout << z.get(GRB_StringAttr_VarName) << " "
         << z.get(GRB_DoubleAttr_X) << endl;

    cout << "Obj: " << model.get(GRB_DoubleAttr_ObjVal) << endl;

  } catch(GRBException e) {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  } catch(...) {
    cout << "Exception during optimization" << endl;
  }

  return 0;
}
```

示例细解

让我们一行一行来浏览一遍示例代码，从而理解它是如何达到指定模型所需的优化结果的。

示例代码的一开始指定了须要包含的头文件 *gurobi_c++.h*。Gurobi C++ 应用程序的一开始总是应该包含这个头文件。

环境的创建

示例代码中的第一条可执行语句获取了一个 Gurobi 环境（通过 *GRBEnv()* 构造函数）：

```
  GRBEnv env = GRBEnv();
```

程序之后用于创建优化模型的方法调用总是需要一个环境，所以环境的创建通常是 Gurobi 应用程序中的第一个步骤。

模型的创建

一旦创建了环境，下一步就可以创建模型了。一个 Gurobi 模型中保存了单个优化问题。它包含一组变量、一组约束、以及相关联的属性（变量边界、目标系数、变量整性类型、约束含义、约束右侧值，等等）。生成一个包含所有这些信息的模型可以从创建一个空的模型对象开始：

```
  GRBModel model = GRBModel(env);
```

构造函数以之前所创建的环境作为它的参数。

在模型中添加变量

在我们的示例中，下一步须要在模型中添加变量。

```
  // Create variables
  GRBVar x = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "x");
  GRBVar y = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "y");
  GRBVar z = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "z");
```

变量可以通过模型对象的 *addVar()* 方法来添加。变量总是和某个特定的模型相关联的。

addVar() 调用的第一个和第二个参数分别是变量的下界值和上界值。第三个参数是线性目标系数（此处先设置为 0，我们稍后将添加目标函数）。第四个参数是变量的类型。在这一示例中，变量都是二元的。最后一个参数是变量的名称。

addVar() 方法有重载了的方法签名以接受不同的参数列表。更多信息请参见 Gurobi 参考手册。

模型的更新——懒惰修改

Gurobi 优化器中对于模型的修改是以一种*懒惰*方式来执行的，这意味着修改的效果并非马上就可以看到。这一更新方式使得执行一系列的模型修改变得更为容易，因为模型在修改的过程中并不会真正改变。然而，懒惰的修改方式要求您在需要的时候手动地整合模型的修改。这可以通过以下方法来实现：

```
  // Integrate new variables
  model.update();
```

设置目标函数

接下来我们在示例中添加优化目标：

```
  // Set objective: maximize x + y + 2 z
  model.setObjective(x + y + 2 * z, GRB_MAXIMIZE);
```

此处的目标函数是通过重载运算符生成的。C++ 的 API 对算术运算符进行了重载，您可以使用它们来生成包含 Gurobi 变量的线性与二次表达式。

第二个参数表示目标最大化。

在模型中添加约束

下一步就是在模型中添加约束。第一个约束是在这里添加的：

```
  // Add constraint: x + 2 y + 3 z <= 4
  model.addConstr(x + 2 * y + 3 * z <= 4, "c0");
```

和变量一样，约束也总是和某个特定的模型相关联的。可以调用模型对象的 *addConstr()* 或 *addConstrs()* 方法来创建约束。

我们同样使用了重载的算数运算符来生成线性表达式。C++ 的 API 中还重载了比较运算符，这使得线性约束的生成变得十分简单。

第二个约束也是通过相似的方法调用来添加的：

```
// Add constraint: x + y >= 1
model.addConstr(x + y >= 1, "c1");
```

模型的优化

现在模型已经生成了，下一步就是对其进行优化：

```
// Optimize model
model.optimize();
```

该方法会对模型执行优化并填充一些内部模型属性，包括优化状态、解决方案，等等。

结果报告——属性

优化一旦完成，我们就可以查询一些属性的值。特别地，我们可以查询 *VarName* 和 *X* 属性来获得每个变量的名称以及解决方案的值：

```
cout << x.get(GRB_StringAttr_VarName) << " "
     << x.get(GRB_DoubleAttr_X) << endl;
cout << y.get(GRB_StringAttr_VarName) << " "
     << y.get(GRB_DoubleAttr_X) << endl;
cout << z.get(GRB_StringAttr_VarName) << " "
     << z.get(GRB_DoubleAttr_X) << endl;
```

我们还可以查询模型的 *ObjVal* 属性来获得当前解所对应的目标值：

```
cout << "Obj: " << model.get(GRB_DoubleAttr_ObjVal) << endl;
```

所有模型、变量、以及约束属性的名称和类型都可以在 Gurobi 参考手册的*属性*节中找到。

错误处理

在 Gurobi 的 C++ 接口中，错误都是通过 C++ 的异常机制来处理的。在我们的示例中，所有 Gurobi 语句都包含在一个 *try* 代码块中，任何错误都可以通过相关联的 *catch* 代码块捕获。

示例的生成与运行

要生成并运行该示例，请参见 *<installdir>/examples/build* 中的文件。对于 Windows 平台，该目录中包含了 *C++_examples_2008.sln* 和 *C++_examples_2010.sln*（C++ 示例的 Visual Studio 2008 和 2010 解决方案文件）。双击解决方案文件会启动 Visual Studio。单击“mip1_c++”项目，然后在“*Build*”菜单中选择“*Run*”，Visual Studio 会编译并运行该示例。对于 Linux 或 Mac OS 平台，*<installdir>/examples/build* 目录中包含了该示例的 Makefile。键入“*make mip1_c++*”会生成并运行该示例。

C++ 的示例目录 *<installdir>/examples/c++* 中包含了许多示例。我们鼓励您浏览并修改它们，从而对 Gurobi 的 C++ 接口更加熟悉。我们同时也鼓励您阅读 Gurobi 示例导览来获得更多的信息。

页面：Java Interface

Java 接口

本节通过一个简单的 Java 程序示例来演示如何使用 Gurobi 的 Java 接口。该示例会生成一个模型、对它进行优化、并输出最优的目标值。本节须要您对于 Java 编程语言有一定的了解。如果您对于 Java 语言编程还不是非常熟悉，可以先参考一些相关的书籍资料和网站进行语言学习（例如，在线 Java 教程）。

我们的示例须要优化下面的模型：

$$\begin{array}{rl} \textbf{maximize} & x + y + 2z \\ \textbf{subject to} & x + 2y + 3z \le 4 \\ & x + y \ge 1 \\ & x, y, z \text{ binary} \end{array}$$

请注意，这个模型和我们在 C 接口节中建立并优化的是同一个模型。

示例 mip1.java

我们这个示例的完整源代码如下所示（也可以参见 *<installdir>/examples/java/Mip1.java*）…

```
import gurobi.*;

public class Mip1 {
  public static void main(String[] args) {
    try {
      GRBEnv    env   = new GRBEnv("mip1.log");
      GRBModel  model = new GRBModel(env);

      // Create variables

      GRBVar x = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, "x");
      GRBVar y = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, "y");
      GRBVar z = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, "z");

      // Integrate new variables

      model.update();

      // Set objective: maximize x + y + 2 z

      GRBLinExpr expr = new GRBLinExpr();
      expr.addTerm(1.0, x); expr.addTerm(1.0, y); expr.addTerm(2.0, z);
      model.setObjective(expr, GRB.MAXIMIZE);

      // Add constraint: x + 2 y + 3 z <= 4

      expr = new GRBLinExpr();
      expr.addTerm(1.0, x); expr.addTerm(2.0, y); expr.addTerm(3.0, z);
      model.addConstr(expr, GRB.LESS_EQUAL, 4.0, "c0");
      // Add constraint: x + y >= 1

      expr = new GRBLinExpr();
      expr.addTerm(1.0, x); expr.addTerm(1.0, y);
      model.addConstr(expr, GRB.GREATER_EQUAL, 1.0, "c1");

      // Optimize model

      model.optimize();

      System.out.println(x.get(GRB.StringAttr.VarName)
                         + " " +x.get(GRB.DoubleAttr.X));
      System.out.println(y.get(GRB.StringAttr.VarName)
                         + " " +y.get(GRB.DoubleAttr.X));
      System.out.println(z.get(GRB.StringAttr.VarName)
                         + " " +z.get(GRB.DoubleAttr.X));

      System.out.println("Obj: " + model.get(GRB.DoubleAttr.ObjVal));

      // Dispose of model and environment

      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
                         e.getMessage());
    }
  }
}
```

示例细解

让我们一行一行来浏览一遍示例代码，从而理解它是如何达到指定模型所需的优化结果的。

示例代码的一开始导入了 Gurobi 类（*import gurobi.**）。Gurobi Java 应用程序总是应该以这一行开始。

环境的创建

示例代码中的第一条可执行语句获取了一个 Gurobi 环境（通过 *GRBEnv()* 构造函数）：

```
   GRBEnv env = new GRBEnv("mip1.log");
```

程序之后用于创建优化模型的方法调用总是需要一个环境，所以环境的创建通常是 Gurobi 应用程序中的第一个步骤。构造函数的参数指定了日志文件的名称。

模型的创建

一旦创建了环境，下一步就可以创建模型了。一个 Gurobi 模型中保存了单个优化问题。它包含一组变量、一组约束、以及相关联的属性（变量边界、目标系数、变量整性类型、约

束含义、约束右侧值，等等）。生成一个包含所有这些信息的模型可以从创建一个空的模型对象开始：

```
  GRBModel model = new GRBModel(env);
```

构造函数以之前所创建的环境作为它的参数。

在模型中添加变量

在我们的示例中，下一步须要在模型中添加变量。

```
  // Create variables
  GRBVar x = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, "x");
  GRBVar y = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, "y");
  GRBVar z = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, "z");
```

变量可以通过模型对象的 ***addVar()*** 方法来添加。变量总是和某个特定的模型相关联的。

addVar() 调用的第一个和第二个参数分别是变量的下界值和上界值。第三个参数是线性目标系数（此处先设置为 0，我们稍后将添加目标函数）。第四个参数是变量的类型。在这一示例中，变量都是二元的。最后一个参数是变量的名称。

addVar() 方法有重载了的方法签名以接受不同的参数列表。更多信息请参见 Gurobi 参考手册。

模型的更新——懒惰修改

Gurobi 优化器中对于模型的修改是以一种*懒惰*方式来执行的，这意味着修改的效果并非马上就可以看到。这一更新方式使得执行一系列的模型修改变得更为容易，因为模型在修改的过程中并不会真正改变。然而，懒惰的修改方式要求您在需要的时候手动地整合模型的修改。这可以通过以下方法来实现：

```
  // Integrate new variables
  model.update();
```

设置目标函数

接下来我们在示例中添加优化目标：

```
  // Set objective: maximize x + y + 2 z

  GRBLinExpr expr = new GRBLinExpr();
  expr.addTerm(1.0, x); expr.addTerm(1.0, y); expr.addTerm(2.0, z);
  model.setObjective(expr, GRB.MAXIMIZE);
```

第二个参数表示目标最大化。

在模型中添加约束

下一步就是在模型中添加约束。第一个约束是在这里添加的：

```
  // Add constraint: x + 2 y + 3 z <= 4
  GRBLinExpr expr;
```

```
expr = new GRBLinExpr();
expr.addTerm(1.0, x); expr.addTerm(2.0, y); expr.addTerm(3.0, z);
model.addConstr(expr, GRB.LESS_EQUAL, 4.0, "c0");
```

和变量一样，约束也总是和某个特定的模型相关联的。可以调用模型对象的 *addConstr()* 或 *addConstrs()* 方法来创建约束。

addConstr() 方法的第一个参数是约束的左侧表达式。我们在生成左侧表达式时，首先创建了一个空的线性表达式对象，然后在其中添加三个线性项。第二个参数是约束的含义（*GRB_LESS_EQUAL*、*GRB_GREATER_EQUAL*、或 *GRB_EQUAL*）。第三个参数是右侧表达式（在我们的示例中为一个常量）。最后一个参数是约束的名称。*addConstr()* 有多个方法签名可以使用。更多信息请参见 Gurobi 参考手册。

第二个约束也是通过相似的方法调用来添加的：

```
// Add constraint: x + y >= 1

expr = new GRBLinExpr();
expr.addTerm(1.0, x); expr.addTerm(1.0, y);
model.addConstr(expr, GRB.GREATER_EQUAL, 1.0, "c1");
```

模型的优化

现在模型已经生成了，下一步就是对其进行优化：

```
// Optimize model
model.optimize();
```

该方法会对模型执行优化并填充一些内部模型属性，包括优化状态、解决方案，等等。

结果报告——属性

优化一旦完成，我们就可以查询一些属性的值。特别地，我们可以查询 *VarName* 和 *X* 属性来获得每个变量的名称以及解决方案的值：

```
System.out.println(x.get(GRB.StringAttr.VarName)
                   + " " +x.get(GRB.DoubleAttr.X));
System.out.println(y.get(GRB.StringAttr.VarName)
                   + " " +y.get(GRB.DoubleAttr.X));
System.out.println(z.get(GRB.StringAttr.VarName)
                   + " " +z.get(GRB.DoubleAttr.X));
```

我们还可以查询模型的 *ObjVal* 属性来获得当前解所对应的目标值：

```
System.out.println("Obj: " + model.get(GRB.DoubleAttr.ObjVal));
```

所有模型、变量、以及约束属性的名称和类型都可以在 Gurobi 参考手册的*属性*节中找到。

资源清理

示例的最后调用了 *dispose* 方法：

```
model.dispose();
env.dispose();
```

这些语句用于释放模型与环境中所占用的资源。如果不调用 *dispose* 方法，垃圾回收器最终还是会回收这些资源，但如果您的程序在执行优化之后并不会立即退出，那么最好还是能够显式地执行资源清理。

请注意，释放环境之前必须先释放与该环境相关的所有模型。

错误处理

在 Gurobi 的 Java 接口中，错误都是通过 Java 的异常机制来处理的。在我们的示例中，所有 Gurobi 语句都包含在一个 *try* 代码块中，任何错误都可以通过相关联的 *catch* 代码块捕获。

示例的生成与运行

要生成并运行该示例，请参见 *<installdir>/examples/build* 中的文件。对于 Windows 平台，该目录中包含了 *runjava.bat* 文件，它是用于编译并运行 java 示例的一个简单脚本。键入“*runjava Mip1*”来运行该示例。对于 Linux 或 Mac OS 平台，*<installdir>/examples/build* 目录中包含了该示例的 Makefile。键入“*make Mip1*”会生成并运行该示例。

Java 的示例目录 *<installdir>/examples/java* 中包含了许多示例。我们鼓励您浏览并修改它们，从而对 Gurobi 的 Java 接口更加熟悉。我们同时也鼓励您阅读 Gurobi 示例导览来获得更多的信息。

页面：.NET Interface (C#)

.NET 接口（C#）

本节通过一个简单的 .NET 程序示例来演示如何使用 Gurobi 的 .NET 接口。该示例会生成一个模型、对它进行优化、并输出最优的目标值。本节须要您对于 C# 编程语言有一定的了解。如果您对于 .NET 语言编程还不是非常熟悉，可以先参考一些相关的书籍资料和网站进行语言学习（例如，Microsoft C# 在线文档）。

我们的示例须要优化下面的模型：

$$\begin{array}{rl}\textbf{maximize} & x + y + 2z \\ \textbf{subject to} & x + 2y + 3z \le 4 \\ & x + y \ge 1 \\ & x, y, z \text{ binary}\end{array}$$

请注意，这个模型和我们在 C 接口节中建立并优化的是同一个模型。

示例 mip1_cs.cs

我们这个示例的完整源代码如下所示（也可以参见 *<installdir>/examples/c#/mip1_cs.cs*）…

```
using System;
using Gurobi;

class mip1_cs
{
  static void Main()
  {
    try {
      GRBEnv    env   = new GRBEnv("mip1.log");
      GRBModel  model = new GRBModel(env);

      // Create variables

      GRBVar x = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "x");
      GRBVar y = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "y");
      GRBVar z = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "z");

      // Integrate new variables

      model.Update();

      // Set objective: maximize x + y + 2 z

      model.SetObjective(x + y + 2 * z, GRB.MAXIMIZE);

      // Add constraint: x + 2 y + 3 z <= 4

      model.AddConstr(x + 2 * y + 3 * z <= 4.0, "c0");
      // Add constraint: x + y >= 1

      model.AddConstr(x + y >= 1.0, "c1");

      // Optimize model

      model.Optimize();

      Console.WriteLine(x.Get(GRB.StringAttr.VarName)
                         + " " + x.Get(GRB.DoubleAttr.X));
      Console.WriteLine(y.Get(GRB.StringAttr.VarName)
                         + " " + y.Get(GRB.DoubleAttr.X));
      Console.WriteLine(z.Get(GRB.StringAttr.VarName)
                         + " " + z.Get(GRB.DoubleAttr.X));

      Console.WriteLine("Obj: " + model.Get(GRB.DoubleAttr.ObjVal));

      // Dispose of model and env

      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
  }
}
```

示例细解

让我们一行一行来浏览一遍示例代码，从而理解它是如何达到指定模型所需的优化结果的。

示例代码的一开始导入了 Gurobi 命名空间（*using Gurobi*）。Gurobi .NET 应用程序总是应该以这一行开始。

环境的创建

示例代码中的第一条可执行语句获取了一个 Gurobi 环境（通过 *GRBEnv()* 构造函数）：

```
   GRBEnv env = new GRBEnv("mip1.log");
```

程序之后用于创建优化模型的方法调用总是需要一个环境，所以环境的创建通常是 Gurobi 应用程序中的第一个步骤。构造函数的参数指定了日志文件的名称。

模型的创建

一旦创建了环境，下一步就可以创建模型了。一个 Gurobi 模型中保存了单个优化问题。它包含一组变量、一组约束、以及相关联的属性（变量边界、目标系数、变量整性类型、约束含义、约束右侧值，等等）。生成一个包含所有这些信息的模型可以从创建一个空的模型对象开始:

```
  GRBModel model = new GRBModel(env);
```

构造函数以之前所创建的环境作为它的参数。

在模型中添加变量

在我们的示例中，下一步须要在模型中添加变量。

```
  // Create variables
  GRBVar x = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "x");
  GRBVar y = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "y");
  GRBVar z = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "z");
```

变量可以通过模型对象的 *AddVar()* 方法来添加。变量总是和某个特定的模型相关联的。

AddVar() 调用的第一个和第二个参数分别是变量的下界值和上界值。第三个参数是线性目标系数（此处先设置为 0，我们稍后将添加目标函数）。第四个参数是变量的类型。在这一示例中，变量都是二元的。最后一个参数是变量的名称。

AddVar() 方法有重载了的方法签名以接受不同的参数列表。更多信息请参见 Gurobi 参考手册。

模型的更新——懒惰修改

Gurobi 优化器中对于模型的修改是以一种*懒惰*方式来执行的，这意味着修改的效果并非马上就可以看到。这一更新方式使得执行一系列的模型修改变得更为容易，因为模型在修改的过程中并不会真正改变。然而，懒惰的修改方式要求您在需要的时候手动地整合模型的修改。这可以通过以下方法来实现：

```
  // Integrate new variables
  model.Update();
```

设置目标函数

接下来我们在示例中添加优化目标：

```
  // Set objective: maximize x + y + 2 z
  model.SetObjective(x + y + 2 * z, GRB.MAXIMIZE);
```

此处的目标函数是通过重载运算符生成的。C# 的 API 对算术运算符进行了重载，您可以使用它们来生成包含 Gurobi 变量的线性与二次表达式。

第二个参数表示目标最大化。

在模型中添加约束

下一步就是在模型中添加约束：

```
  // Add constraint: x + 2 y + 3 z <= 4
  model.AddConstr(x + 2 * y + 3 * z <= 4.0, "c0");

  // Add constraint: x + y >= 1
  model.AddConstr(x + y >= 1.0, "c1");
```

和变量一样，约束也总是和某个特定的模型相关联的。可以调用模型对象的 *AddConstr()* 或 *AddConstrs()* 方法来创建约束。

我们同样使用了重载的算数运算符来生成线性表达式。C# 的 API 中还重载了比较运算符，这使得线性约束的生成变得十分简单。

在 Gurobi 的 .NET 接口中，还可以通过一项一项生成线性表达式的方式来添加约束：

```
  GRBLinExpr expr = 0.0;
  expr.AddTerm(1.0, x);
  expr.AddTerm(2.0, x);
  expr.AddTerm(3.0, x);
  model.AddConstr(expr, GRB.LESS_EQUAL, 4.0, "c0");
```

上例中的 *AddConstr()* 方法签名包含四个参数，其中第一个参数是一个线性表达式，它描述了约束的左侧表达式、第二个参数是约束的含义、第三个参数也是一个线性表达式，用于描述约束的右侧表达式、第四个参数则指定了约束的名称。

模型的优化

现在模型已经生成了，下一步就是对其进行优化：

```
  // Optimize model
  model.Optimize();
```

该方法会对模型执行优化并填充一些内部模型属性，包括优化状态、解决方案，等等。

结果报告——属性

优化一旦完成，我们就可以查询一些属性的值。特别地，我们可以查询 *VarName* 和 *X* 属性来获得每个变量的名称以及解决方案的值：

```
  Console.WriteLine(x.Get(GRB.StringAttr.VarName) + " " +
x.Get(GRB.DoubleAttr.X));
  Console.WriteLine(y.Get(GRB.StringAttr.VarName) + " " +
y.Get(GRB.DoubleAttr.X));
  Console.WriteLine(z.Get(GRB.StringAttr.VarName) + " " +
z.Get(GRB.DoubleAttr.X));
```

我们还可以查询模型的 *ObjVal* 属性来获得当前解所对应的目标值：

```
  Console.WriteLine("Obj: " + model.Get(GRB.DoubleAttr.ObjVal));
```

所有模型、变量、以及约束属性的名称和类型都可以在 Gurobi 参考手册的*属性*节中找到。

资源清理

示例的最后调用了 *Dispose* 方法：

```
  model.Dispose();
  env.Dispose();
```

这些语句用于释放模型与环境中所占用的资源。如果不调用 *Dispose* 方法，垃圾回收器最终还是会回收这些资源，但如果您的程序在执行优化之后并不会立即退出，那么最好还是能够显式地执行资源清理。

请注意，释放环境之前必须先释放与该环境相关的所有模型。

错误处理

在 Gurobi 的 .NET 接口中，错误都是通过 .NET 的异常机制来处理的。在我们的示例中，所有 Gurobi 语句都包含在一个 *try* 代码块中，任何错误都可以通过相关联的 *catch* 代码块捕获。

示例的生成与运行

可以使用 *<installdir>/examples/build* 中的 *CS_examples_2008.sln* 或 *CS_examples_2010.sln* 文件分别在 Visual Studio 2008 或 2010 中生成并运行该示例。单击“mip1_cs”项目，然后在“*Build*”菜单中选择“*Run*”，Visual Studio 会编译并运行该示例。

C# 和 Visual Basic 的示例目录（*<installdir>/examples/c#* 和 *<installdir>/examples/vb*）中包含了许多示例。我们鼓励您浏览并修改它们，从而对 Gurobi 的 .NET 接口更加熟悉。我们同时也鼓励您阅读 Gurobi 示例导览来获得更多的信息。

页面：Python Interface

Python 接口

您在很多场合下都会使用 Gurobi 的 Python 接口。首先，它是交互式命令解释器的基础，通常可以使用交互式命令解释器来优化现有的模型。您也可以直接使用 Python 接口来编写独立的程序，就像使用其他的编程语言接口一样创建并求解模型。不仅如此，我们的 Python 接口中还包含了一些更高层次的建模结构，通过这些结构您可以以一种更加数学化的语法来生成模型，非常类似于传统的建模语言。在之前的章节中我们已经对交互式命令解释器进行了介绍。本章节中我们将一起去看两个示例。第一个示例中所呈现的 Python 程序与之前章节中介绍的 C、C++、Java、以及 C# 程序十分相似。而第二个示例则会演示 Python 接口中的一些高级建模功能。

本节须要您对于 Python 编程语言有一定的了解，并且已经阅读了之前的 Gurobi 交互式命令解释器节。如果您想要进一步学习 Python 语言，我们建议您访问 Python 在线教程。

请注意，您并不须要专门安装 Python，因为 Gurobi 的发布包中已经包含了运行 Python 程序所需的全部工具。您须要使用一组由我们提供的脚本，从而在我们所发布的 Python 环境中运行 Gurobi Python 程序。此外，如果您已经是 Python 的用户，我们为您提供了一个 *setup.py* 脚本，它会在您的 Python 环境中安装 *gurobipy* 模块。更多信息请参见示例的生成与运行。

AIX 用户的重要注意事项：由于在 AIX 之上的 Python 支持非常有限，我们的 AIX 端口并不包括交互式命令解释器或 Python 接口。您可以使用 C、C++、或 Java 接口。

Python 的 example 目录中包含了许多示例。我们建议您浏览并修改这些示例，它们能够帮助您更好地熟悉 Gurobi 的 Python 接口。您也可以阅读 Gurobi 示例导览来获取更多的信息。

小节

页面：Simple Python Example

简单的 Python 示例

本节通过一个简单的 Python 程序示例来演示如何使用 Gurobi 的 Python 接口。该示例会生成一个模型、对它进行优化、并输出最优的目标值。

我们的示例须要优化下面的模型：

$$
\begin{array}{rl}
\textbf{maximize} & x + y + 2z \\
\textbf{subject to} & x + 2y + 3z \leq 4 \\
 & x + y \geq 1 \\
 & x, y, z \text{ binary}
\end{array}
$$

请注意，这个模型和我们在 C 接口节中建立并优化的是同一个模型。

示例 mip1.py

我们这个示例的完整源代码如下所示（也可以参见 *<installdir>/examples/c#/mip1.py*）…

```
from gurobipy import *

try:

    # Create a new model
    m = Model("mip1")

    # Create variables
    x = m.addVar(vtype=GRB.BINARY, name="x")
    y = m.addVar(vtype=GRB.BINARY, name="y")
    z = m.addVar(vtype=GRB.BINARY, name="z")

    # Integrate new variables
    m.update()

    # Set objective
    m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)

    # Add constraint: x + 2 y + 3 z <= 4
    m.addConstr(x + 2 * y + 3 * z <= 4, "c0")

    # Add constraint: x + y >= 1
    m.addConstr(x + y >= 1, "c1")

    m.optimize()

    for v in m.getVars():
        print v.varName, v.x

    print 'Obj:', m.objVal

except GurobiError:
    print 'Error reported'
```

示例细解

让我们一行一行来浏览一遍示例代码，从而理解它是如何达到指定模型所需的优化结果的。

示例代码的一开始导入了 Gurobi 的函数和类：

```
from gurobipy import *
```

Gurobi Python 应用程序总是应该以这一行开始。

请注意，要成功地执行该命令，Python 应用程序必须清楚地知道如何才能找到 Gurobi 中的函数与类。回忆一下此时我们的两个选择。首先，我们可以使用发布包中所包含的那些 Python 文件。您可以通过键入 *gurobi.bat mip1.py*（Windows 系统）或 *gurobi.sh mip1.py*（Linux 与 Mac 系统）命令运行该示例。或者，您也可以将 Gurobi 中的函数与类安装到您自己的 Python 环境中。具体做法请参照 setup.py 的运行方法说明。

模型的创建

在我们的示例中，第一步是创建一个模型。一个 Gurobi 模型中保存了单个优化问题。它包含一组变量、一组约束、以及相关联的属性（变量边界、目标系数、变量整性类型、约束含义、约束右侧值，等等）。我们从一个空的模型对象开始该示例：

```
  m = Model("mip1")
```

该函数以所需模型的名称作为它的参数。

在模型中添加变量

在我们的示例中，下一步须要在模型中添加变量。

```
  # Create variables
  x = m.addVar(vtype=GRB.BINARY, name="x")
  y = m.addVar(vtype=GRB.BINARY, name="y")
  z = m.addVar(vtype=GRB.BINARY, name="z")
```

变量可以通过模型对象的 *addVar()* 方法来添加。变量总是和某个特定的模型相关联的。

在 Python 中，您可以通过输入参数的位置或名称来传递参数的值。在本例中，我们是通过参数名称来传递的。我们为每一个变量都指定其类型（二元变量）与名称。而所有其他的参数均取默认值。关于 *addVar()* 函数更多的信息请参见在线帮助（在 Gurobi 命令解释器中键入“*help(Model.addVar)*”）。

模型的更新——懒惰修改

Gurobi 优化器中对于模型的修改是以一种*懒惰*方式来执行的，这意味着修改的效果并非马上就可以看到。这一更新方式使得执行一系列的模型修改变得更为容易，因为模型在修改

的过程中并不会真正改变。然而，懒惰的修改方式要求您在需要的时候手动地整合模型的修改。这可以通过以下方法来实现：

```
  # Integrate new variables
  m.update()
```

设置目标函数

接下来我们在示例中添加优化目标：

```
  // Set objective: maximize x + y + 2 z
  model.setObjective(x + y + 2 * z, GRB.MAXIMIZE)
```

此处的目标函数是通过重载运算符生成的。Python 的 API 对算术运算符进行了重载，您可以使用它们来生成包含 Gurobi 变量的线性与二次表达式。

第二个参数表示目标最大化。

在模型中添加约束

下一步就是在模型中添加约束。第一个约束是在这里添加的：

```
  # Add constraint: x + 2 y + 3 z <= 4
  m.addConstr(x + 2 * y + 3 * z <= 4, "c0")
```

和变量一样，约束也总是和某个特定的模型相关联的。可以调用模型对象的 *addConstr()* 方法来创建约束。

我们同样使用了重载的算数运算符来生成线性表达式。Python 的 API 中还重载了比较运算符，这使得线性约束的生成变得十分简单。

第二个约束也是通过相似的方法来添加的：

```
  # Add constraint: x + y >= 1
  m.addConstr(x + y >= 1, "c1")
```

模型的优化

现在模型已经生成了，下一步就是对其进行优化：

```
  // Optimize model
  m.optimize()
```

该方法会对模型执行优化并填充一些内部模型属性，包括优化状态、解决方案，等等。

结果报告——属性

优化一旦完成，我们就可以查询一些属性的值。特别地，我们可以查询 *varName* 和 *x* 属性来获得每个变量的名称以及解决方案的值：

```
  for v in m.getVars():
    print v.varName, v.x
```

我们还可以查询模型的 *objVal* 属性来获得当前解所对应的目标值：

```
  print 'Obj:', m.objVal
```

所有模型、变量、以及约束属性的名称和类型都可以在 Python 在线文档中找到。在 Gurobi 命令解释器中键入“*help(GRB.attr)*”来获得更多信息。

错误处理

在 Gurobi 的 Python 接口中，错误都是通过 Python 的异常机制来处理的。在我们的示例中，所有 Gurobi 语句都包含在一个 *try* 代码块中，任何错误都可以通过相关联的 *except* 代码块捕获。

示例的运行

运行该示例时（在 Windows 中键入 *gurobi.bat mip1.py* 命令；在 Linux 或 Mac 中为 *gurobi.sh mip1.py*），您可以看到以下输出：

```
Optimize a model with 2 rows, 3 columns and 5 nonzeros
Presolve removed 2 rows and 3 columns
Presolve time: 0.00s

Explored 0 nodes (0 simplex iterations) in 0.00 seconds
Thread count was 1 (of 4 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 3.000000000000e+00, best bound 3.000000000000e+00, gap
0.0%
x 1.0
y 0.0
z 1.0
Obj: 3.0
```

页面：Python Dictionary Example

Python 的字典示例

为了提供一个循序渐进的学习过程，我们目前为止所介绍的示例仅仅使用了 Python 接口中最为基本的功能。我们现在将要介绍一个稍稍复杂一些的示例，用于演示 Python 接口中一些强大的建模功能。该示例包含了那些大型复杂优化模型中最为常见的元素。使用其他语言的 API 来实现一个这样的示例可能需要几百行代码（而在 Python 中我们只用了大约 70 行代码就做到了）。

在开始具体的示例介绍之前让我们先作一些简要的申明。我们将会涉及一些关于自定义类与函数的内容，因此您须要对 Python 语言的基本概念有一定的了解。我们希望您在阅读完本节内容之后能够深深感受到 Python 接口强大而又灵活的优势。在 Python 接口中，您可以通过非常简洁而又自然的建模结构来创建复杂的模型。我们对于 Python 接口的期望在于，我们希望它可以为您提供一些比起编程语言来说更像是数学建模语言的 API，能够帮助您实现模型的创建。

如果您想要更深入地了解我们这里所提到的 Python 语言结构，我们建议您可以访问在线 Python 教程。

挑战

任何一个优化模型中最为重要的部分总是它所包含的那些决策变量。在实现模型时经常会遇到的一个主要的挑战就是：如何才能方便地存储并访问这些变量。尽管有些模型变量可以通过编程语言中简单的数据结构十分自然地进行表示（例如，*x[i]* 与连续整数值 *i*），然而在大多数其他的模型中，变量的表示可能会复杂许多。例如，我们考虑一个这样的模型，它用于优化供应网络中多种不同商品的流量。您也许会定义一个变量 *x['Pens', 'Denver', "New York']* 来表示某一种制造产品（本例中为钢笔）从丹佛运输到纽约的数量。与此同时，您也许并不需要变量 *x['Pencils', "Denver', "Seattle']*，因为并不是所有的商品、输出城市、与目标城市的组合都是有效的网络路径。在普通的编程语言中，决策变量的稀疏集合表示通常是十分麻烦的。

除此之外，您通常还会在模型中添加一些约束。约束中须要包含决策变量的某些子集，这无疑进一步增加了建模的难度。以我们之前提到的网络流量模型为例，您也许想要设定一个上界值，用于限制流入某个目标城市的商品总量。您当然可以遍历每一个城市，并收集所有表示从这个城市进入目标城市的商品流量。然而，如果并非所有的输出城市-目标城市组合都是有效的，那么这种做法显然将会十分的低效。在一个庞大的网络问题中，这种低效的方法有可能会导致严重的效率问题。通常来说，须要使用较为复杂的数据结构才能有效地解决这一问题。

Gurobi 中所支持的 Python 接口可以帮助我们十分简单地解决之前所讨论的所有这些问题。我们稍后将会给出一个具体的示例来展示这一切将是如何做到的。然而在这之前，让

我们一起来熟悉一下 Python 中一些重要的数据结构：*列表*、*元组*、*字典*、*列表解析*、以及 *tuplelist* 类。其中前四种是标准的 Python 数据结构，它们在我们所提供的接口中将起到十分重要的作用；最后一种是我们在 Gurobi 的 Python 接口中添加的自定义类。

快速提醒：关于这些 Python 数据结构更为详细的介绍请参见在线 Python 文档。

列表与元组

大多数的 Python 程序中都会用到*列表*数据结构；Gurobi 的 Python 程序也不例外。另一种经常使用的数据结构称为元组。在 Gurobi 的 Python 程序中，使用元组能够非常高效而又方便地访问 Gurobi 模型中的决策变量。列表与元组之间的差别是十分细微的但却非常重要。我们之后将对其进行详细的介绍。

列表与元组结构都是 Python 对象的简单有序集。列表可以通过方括号中一组由逗号分隔的成员对象来创建并表示。元组的表示十分相似，只是所有的成员对象都须要放在圆括号之中。例如，*[1, 2, 3]* 表示的是列表；而 *(1, 2, 3)* 则是元组。相似地，*['Pens', 'Denver', 'New York']* 是列表；而 *（'Pens', 'Denver', 'New York'）* 是元组。

您可以通过方括号以及从零开始的索引值来检索列表或元组中的单个元素：

```
gurobi> l = [1, 2.0, 'abc']
gurobi> t = (1, 2.0, 'abc')
gurobi> print l[0]
1
gurobi> print t[1]
2.0
gurobi> print l[2]
abc
```

那么列表与元组之间有什么区别呢？元组是无法修改的，这意味着一旦创建了元组之后就不能再对其进行修改。而在列表中，我们则可以随意地添加、删除、或修改其中的成员。我们可以利用元组的这种特性将其作为*字典*的索引来使用。

字典

Python 中的*字典*可以通过任意*键*值对数据进行映射。任何无法修改的 Python 对象都可以当作键值使用：整数、浮点数、字符串、甚至于元组。

举个例子，以下这些语句创建了一个字典对象 *x*，并将数值 *1* 与键值 *('Pens', 'Denver', 'New York')* 进行关联：

```
gurobi> x = {}  # creates an empty dictionary
gurobi> x[('Pens', 'Denver', 'New York')] = 1
gurobi> print x[('Pens', 'Denver', 'New York')]
1
```

在 Python 中，通过元组访问字典时可以省略圆括号，因此以下语句同样也是同样有效的：

```
gurobi> x = {}
gurobi> x['Pens', 'Denver', 'New York'] = 2
gurobi> print x['Pens', 'Denver', 'New York']
2
```

在此例中我们通过字典来存储整数值，然而字典也可以用于保存任意其他的对象。特别须要指出的是，我们可以使用它们来存储Gurobi的决策变量：

```
gurobi> x['Pens', 'Denver', 'New York'] = model.addVar()
gurobi> print x['Pens', 'Denver', 'New York']
<gurobi.Var *Awaiting Model Update*>
```

字典对象最简单的初始化方法就是为每一个键值都进行赋值：

```
gurobi> values = {}
gurobi> values['zero'] = 0
gurobi> values['one'] = 1
gurobi> values['two'] = 2
```

当然，您也可以直接使用Python中字典的初始化构造方法：

```
gurobi> values = { 'zero': 0, 'one': 1, 'two': 2 }
gurobi> print values['zero']
0
gurobi> print values['one']
1
```

Gurobi的Python接口中包含了一个实用的方法，通过它能够大大简化数学建模中经常须要执行的字典初始化操作。使用*multidict*函数使您可以在一条语句中初始化一个或多个字典对象。该函数的输入参数为一个字典对象，其中每一个键值都对应着一个长度为*n*的列表。该函数会将其中每一个列表都拆分成*n*个单项，从而创建出*n*个独立的字典。函数的返回值是一个列表，其中第一个结果是共享的键值列表，后面紧跟着的是新创建的*n*个字典对象：

```
gurobi> names, lower, upper = multidict({ 'x': [0, 1], 'y': [1, 2], 'z':
[0, 3] })
gurobi> print names
['x', 'y', 'z']
gurobi> print lower
{'x': 0, 'y': 1, 'z': 0}
gurobi> print upper
{'x': 1, 'y': 2, 'z': 3}
```

请注意，您也可以将一个每个键值只映射单个标量的字典传入该函数。在这种情况下，该函数只返回两个结果，其中第一个是键值列表；而第二个则是传入的那个字典对象。

您将看到，我们在许多Python示例中都会使用到这个函数。

列表解析

列表解析是Python中一个十分重要的功能，它使您能够以非常简洁的方式来生成列表。举一个简单的例子，以下列表解析所生成的列表中包含了从1到5每个数字的平方值：

```
gurobi> print [x*x for x in [1, 2, 3, 4, 5]]
[1, 4, 9, 16, 25]
```

一个列表解析中可以包含多个“*for*”子句，同时它也可以包含一个或多个“*if*”子句。以下示例会生成一个元组列表，其中包含了所有的 *x*、*y* 数值对，只要 x 与 y 都小于 3 且不相等即可：

```
gurobi> print [(x,y) for x in range(3) for y in range(3) if x != y]
[(0, 1), (0, 2), (1, 0), (1, 2) (2, 0), (2, 1)]
```

（对于 range 函数的介绍请参见此处）。我们的 Python 示例中会大量使用列表解析。

tuplelist 类

接下来让我们介绍最后一种重要的数据结构——*tuplelist* 类。这个类继承于 Python 中的列表类，使用它能够有效地在元组列表中检索符合要求的子列表。具体来说，您可以使用 *tuplelist* 对象中的 *select* 方法返回所有满足特定条件（使用特定字段来匹配一个或多个特定值）的元组子集。

举一个简单的例子：首先我们通过以下语句创建一个简单的 *tuplelist* 对象（将一个元组列表传入 *tuplelist* 类的构造函数）：

```
gurobi> l = tuplelist([(1, 2), (1, 3), (2, 3), (2, 4)])
```

我们可以在 *select* 方法的输入参数中为每一个元组元素都指定其须要匹配的期望值。*select* 方法的参数数量应该等同于 *tuplelist* 对象中元组成员的元素个数（所有元组的元素个数都必须相同）。通过字符串“*”来表示元组中相应位置的元素可以匹配任何值。

本例中的每个元组都包含了两个元素，因此我们可以执行以下语句：

```
gurobi> print l.select(1, '*')
[(1, 2), (1, 3)]
gurobi> print l.select('*', 3)
[(1, 3), (2, 3)]
gurobi> print l.select(1, 3)
[(1, 3)]
gurobi> print l.select('*', '*')
[(1, 2), (1, 3), (2, 3), (2, 4)]
```

您或许注意到，通过列表解析其实也可以获得相同的检索结果。例如：

```
gurobi> print l.select(1, '*')
[(1, 2), (1, 3)]
gurobi> print [(x,y) for x,y in l if x == 1]
[(1, 2), (1, 3)]
```

问题在于，第二条语句（列表解析）会遍历列表中的每一个成员，这对于较大的列表来说是十分低效的。而 *select* 方法则可以通过一些内部数据结构来提高数据选择的效率。

请注意，*tuplelist* 是列表类的一个子类，因此您可以使用标准的列表方法来访问或修改一个 *tuplelist* 对象：

```
gurobi> print l[1]
(1,3)
gurobi> l += [(3, 4)]
gurobi> print l
```

```
[(1, 2), (1, 3), (2, 3), (2, 4), (3, 4)]
```

回到最初的网络流量问题，一旦生成了一个包含了网络中所有商品-输出城市-目标城市的有效组合（我们将称之为“流”）的 *tuplelist* 对象，我们就可以通过以下所示的方法来检索流入某个目标城市的所有网络路径：

```
gurobi> inbound = flows.select('*', '*', 'New York')
```

我们现在通过一个示例来展示如何使用以上章节中所介绍的所有这些概念。

netflow.py 示例

我们将要介绍的这个示例旨在求解一个基于小型网络的多商品流动模型。在该示例中，有两种商品（铅笔与钢笔），它们在两个城市（底特律与丹佛）中进行生产，然后须要运送到位于另外三个城市（波士顿、纽约、与西雅图）的仓库中以满足给定的需求。运输网络中的每一条路径都有与之相对应的运输成本以及总运能。

以下是这个示例完整的源代码（也可参见 *<installdir>/examples/python/netflow.py* 文件）…

```
from gurobipy import *

# Model data

commodities = ['Pencils', 'Pens']
nodes = ['Detroit', 'Denver', 'Boston', 'New York', 'Seattle']

arcs, capacity = multidict({
  ('Detroit', 'Boston'):   100,
  ('Detroit', 'New York'):  80,
  ('Detroit', 'Seattle'):  120,
  ('Denver',  'Boston'):   120,
  ('Denver',  'New York'): 120,
  ('Denver',  'Seattle'):  120 })
arcs = tuplelist(arcs)

cost = {
  ('Pencils', 'Detroit', 'Boston'):   10,
  ('Pencils', 'Detroit', 'New York'): 20,
  ('Pencils', 'Detroit', 'Seattle'):  60,
  ('Pencils', 'Denver',  'Boston'):   40,
  ('Pencils', 'Denver',  'New York'): 40,
  ('Pencils', 'Denver',  'Seattle'):  30,
  ('Pens',    'Detroit', 'Boston'):   20,
  ('Pens',    'Detroit', 'New York'): 20,
  ('Pens',    'Detroit', 'Seattle'):  80,
  ('Pens',    'Denver',  'Boston'):   60,
  ('Pens',    'Denver',  'New York'): 70,
  ('Pens',    'Denver',  'Seattle'):  30 }

inflow = {
  ('Pencils', 'Detroit'):   50,
  ('Pencils', 'Denver'):    60,
  ('Pencils', 'Boston'):   -50,
  ('Pencils', 'New York'): -50,
  ('Pencils', 'Seattle'):  -10,
  ('Pens',    'Detroit'):   60,
  ('Pens',    'Denver'):    40,
  ('Pens',    'Boston'):   -40,
  ('Pens',    'New York'): -30,
  ('Pens',    'Seattle'):  -30 }

# Create optimization model
m = Model('netflow')

# Create variables
flow = {}
for h in commodities:
    for i,j in arcs:
        flow[h,i,j] = m.addVar(ub=capacity[i,j], obj=cost[h,i,j],
                               name='flow_%s_%s_%s' % (h, i, j))
m.update()

# Arc capacity constraints
for i,j in arcs:
    m.addConstr(quicksum(flow[h,i,j] for h in commodities) <=
capacity[i,j],
                'cap_%s_%s' % (i, j))

# Flow conservation constraints
for h in commodities:
    for j in nodes:
        m.addConstr(
          quicksum(flow[h,i,j] for i,j in arcs.select('*',j)) +
              inflow[h,j] ==
          quicksum(flow[h,j,k] for j,k in arcs.select(j,'*')),
                   'node_%s_%s' % (h, j))

# Compute optimal solution
m.optimize()

# Print solution
if m.status == GRB.status.OPTIMAL:
    for h in commodities:
        print '\nOptimal flows for', h, ':'
        for i,j in arcs:
            if flow[h,i,j].x > 0:
                print i, '->', j, ':', flow[h,i,j].x
```

示例详解

现在让我们从头到尾一行一行地解读这些代码，理解它究竟是如何实现网络流量的优化与求解的。与之前所介绍的那些简单的 Python 示例一样，本例一开始也是先导入了 Gurobi 中的函数与类：

```
from gurobipy import *
```

之后我们创建了一些列表用以包含模型数据：

```
commodities = ['Pencils', 'Pens']
nodes = ['Detroit', 'Denver', 'Boston', 'New York', 'Seattle']
```

```
arcs, capacity = multidict({
  ('Detroit', 'Boston'):   100,
  ('Detroit', 'New York'):  80,
  ('Detroit', 'Seattle'):  120,
  ('Denver',  'Boston'):   120,
  ('Denver',  'New York'): 120,
  ('Denver',  'Seattle'):  120 })
arcs = tuplelist(arcs)
```

模型中定义了两种商品（铅笔与钢笔）、以及一个包含了 5 个节点与 6 条路径的网络结构。我们将 *commodities* 与 *nodes* 定义为简单的 Python 列表，并使用 Gurobi 中的 *multidict* 函数来初始化 *arcs*（键值列表）与 *capacity*（字典对象）。

在该示例中，我们打算之后生成约束的时候能够通过 *arcs* 来选择网络路径中的子集。因此，我们将 *multidict* 函数所返回的元组列表转而传入 *tuplelist* 的构造函数，从而创建了一个 *tuplelist* 类型的对象。

模型中还须要为每一组商品与网络路径定义运输成本数据：

```
cost = {
  ('Pencils', 'Detroit', 'Boston'):   10,
  ('Pencils', 'Detroit', 'New York'): 20,
  ('Pencils', 'Detroit', 'Seattle'):  60,
  ('Pencils', 'Denver',  'Boston'):   40,
  ('Pencils', 'Denver',  'New York'): 40,
  ('Pencils', 'Denver',  'Seattle'):  30,
  ('Pens',    'Detroit', 'Boston'):   20,
  ('Pens',    'Detroit', 'New York'): 20,
  ('Pens',    'Detroit', 'Seattle'):  80,
  ('Pens',    'Denver',  'Boston'):   60,
  ('Pens',    'Denver',  'New York'): 70,
  ('Pens',    'Denver',  'Seattle'):  30 }
```

一旦创建了这样一个字典对象，商品 *h* 从节点 *i* 到节点 *j* 的运输成本就可以通过 *cost[(h, i, j)]* 来表达了。回忆一下，我们之前提到过在 Python 中使用元组来索引字典时可以省略元组两端的圆括号，因此运输成本也可以简写为 *cost[h, i, j]*。

接下来我们使用一个类似的数据结构来初始化节点的需求数据：

```
inflow = {
  ('Pencils', 'Detroit'):   50,
  ('Pencils', 'Denver'):    60,
  ('Pencils', 'Boston'):   -50,
  ('Pencils', 'New York'): -50,
  ('Pencils', 'Seattle'):  -10,
  ('Pens',    'Detroit'):   60,
  ('Pens',    'Denver'):    40,
  ('Pens',    'Boston'):   -40,
  ('Pens',    'New York'): -30,
  ('Pens',    'Seattle'):  -30 }
```

生成一个变量的多维数组

创建了空白的 *Model* 对象之后，接下来我们就要在模型中添加决策变量了。所有的变量都存储在字典对象 *flow* 之中：

```
flow = {}
for h in commodities:
  for i,j in arcs:
    flow[h,i,j] = m.addVar(ub=capacity[i,j], cost=cost[h,i,j],
                           name='flow_%s_%s_%s' % (h, i, j))
m.update()
```

flow 变量有三重下标：商品、源节点、与目标节点。请注意，字典对象中仅包含 *arcs* 中存在的源节点与目标节点所组成的变量。因此，所有这些变量所对应的网络路径都是有效的。

运能约束

我们从最简单的一组约束开始定义。一条路径上所有流动变量之和必须小于等于该网络路径所拥有的运能：

```
for i,j in arcs:
  m.addConstr(quicksum(flow[h,i,j] for h in commodities) <=
capacity[i,j],
              'cap_%s_%s' % (i, j))
```

请注意，我们在此处使用了列表解析来生成所有与路径 *(i, j)* 相关的变量的列表：

```
flow[h,i,j] for h in commodities
```

（精确来说，我们这里使用的其实是 Python 中所谓的生成器表达式，但它与列表解析十分相似，对于理解本示例不会产生任何影响，所以您可以忽略它们之间的区别）。结果列表传入 *quicksum* 函数之后会创建一个 Gurobi 的线性表达式，表示所有这些变量之和。Gurobi 的 *quicksum* 函数是 Python 中 *sum* 函数的一个替代品，使用它能够更快地生成大型表达式。

流量守恒约束

接下来我们要添加的是流量守恒约束。这组约束要求，对于每一种商品来说，流入每个节点的商品数量之和加上外部流入该节点（生产与需求）的数量，必须等于这一商品流出该节点的数量之和。

```
for h in commodities:
  for j in nodes:
    m.addConstr(
      quicksum(flow[h,i,j] for i,j in arcs.select('*',j)) + inflow[h,j]
==
      quicksum(flow[h,j,k] for j,k in arcs.select(j,'*')),
               'node_%s_%s' % (h, j))
```

结果

一旦添加了模型约束，我们就可以调用 *optimize* 函数并输出模型的最优解：

```
if m.status == GRB.status.OPTIMAL:
  for h in commodities:
    print '\nOptimal flows for', h, ':'
    for i,j in arcs:
      if flow[h,i,j].x > 0:
        print i, '->', j, ':', flow[h,i,j].x
```

如果运行该示例（在 Windows 上为 *gurobi.bat netflow.py*；在 Linux 与 Mac 上为 *gurobi.sh netflow.py*），您应该就能看到以下输出：

```
Optimize a model with 16 rows, 12 columns and 36 nonzeros
Presolve removed 16 rows and 12 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    5.5000000e+03   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds
Optimal objective  5.500000000e+03

Optimal flows for Pencils :
Detroit -> Boston : 50.0
Denver -> New York : 50.0
Denver -> Seattle : 10.0

Optimal flows for Pens :
Detroit -> Boston : 30.0
Detroit -> New York : 30.0
Denver -> Boston : 10.0
Denver -> Seattle : 30.0
```

页面：Building and running the examples

示例的生成与运行

Python 是一种解释语言，所以运行该示例并不需要显式的编译步骤。对于 Windows 平台，只须要简单地在 Gurobi 的 Python 示例目录（*<installdir>/examples/python*）中键入以下命令：

```
gurobi.bat mip1.py
```

对于 Linux 或 Mac OS 平台，则须要键入：

```
gurobi.sh mip1.py
```

如果您是 Python 用户，并且想要在您自己的 Python 环境中使用 Gurobi，则可以直接在您的环境中安装 *gurobipy* 模块。在 *<installdir>* 中发出以下命令即可：

```
python setup.py install
```

在 Linux 或 Mac OS 系统中，除非您所使用的是您自己私有的 Python 安装，否则您就须要作为超级用户来运行这条命令。*gurobipy* 一旦安装成功，您就可以键入“python mip1.py”（更为通用的做法是在您的 Python 环境中键入“*from gurobipy import **”）。

页面：FileOverview

文件概述

本节会简要地描述 Gurobi 发布中一些比较重要的文件的用途。Windows、Linux、和 Mac OS 发布中大部分的文件结构是相同的，但也有一些文件有所区别，所以我们会分别对它们进行说明。

请注意，下面的列表可能会和您的安装目录有所出入。我们忽略了一些不那么重要的文件。另外，一些文件名取决于所安装的 Gurobi 优化器的精确版本。

Windows 文件结构

下面的文件和目录会在您的安装目录（对于 32-位的 Windows 发布，默认为 *c:\gurobi460\win3*）中创建：

- EULA.doc – Gurobi 最终用户许可协议 – Microsoft Word 格式
- EULA.pdf – Gurobi 最终用户许可协议 – PDF 格式
- ReleaseNotes.html – 发布说明
- bin
  - Gurobi46.NET.XML – .NET 包装的 Visual Studio 帮助
  - Gurobi46.NET.dll – .NET 包装
  - GurobiJni46.dll – Java JNI 包装
  - grbd.exe – Gurobi 许可证管理器的可执行文件
  - grbgetkey – 从 Gurobi 密钥服务器中检索 Gurobi 许可证密钥
  - grbprobe – 探测系统细节（通常不会使用）
  - grbvalidate – 刷新教学许可证（通常不会使用）
  - gurobi.bat – 启动 Gurobi 交互式命令解释器
  - gurobi.env – 参数初始化示例文件
  - gurobi46.dll – Gurobi 的本地 DLL（适用于所有的 Gurobi 接口）
  - gurobi_cl.exe – 简单的命令行二进制文件
- docs
  - examples – 示例导览（打开该目录中的 index.html 文件）
  - quickstart – 快速入门指南（打开该目录中的 index.html 文件）
  - refman – 参考手册（打开该目录中的 index.html 文件）
- examples
  - build – C、C++、C#、和 Visual Basic 示例的 Visual Studio 项目文件；Java 和 Python 示例的 run*.bat 文件
  - c – C 示例的源代码
  - c# – C# 示例的源代码
  - c++ – C++ 示例的源代码
  - data – 示例的数据文件
  - java – Java 示例的源代码
  - python – Python 示例的源代码
  - vb – Visual Basic 示例的源代码
- include

  - gurobi_c++.h – C++ 头文件
  - gurobi_c.h – C 头文件
- lib
  - gurobi.jar – Java 接口
  - gurobi.py – Python 的启动文件
  - gurobi46.lib – Gurobi 的库导入文件
  - gurobi_c++md2008.lib – C++ 接口（Visual Studio 2008 中使用 -MD 编译开关)
  - gurobi_c++md2010.lib – C++ 接口（Visual Studio 2010 中使用 -MD 编译开关)
  - gurobi_c++mdd2008.lib – C++ 接口（Visual Studio 2008 中使用 -MDd 编译开关）
  - gurobi_c++mdd2010.lib – C++ 接口（Visual Studio 2010 中使用 -MDd 编译开关）
  - gurobi_c++mt2008.lib – C++ 接口（Visual Studio 2008 中使用 -MT 编译开关）
  - gurobi_c++mt2010.lib – C++ 接口（Visual Studio 2010 中使用 -MT 编译开关）
  - gurobi_c++mtd2008.lib – C++ 接口（Visual Studio 2008 中使用 -MTd 编译开关）
  - gurobi_c++mtd2010.lib – C++ 接口（Visual Studio 2010 中使用 -MTd 编译开关）
- python27 – 交互式命令解释器和 Python 接口所使用的 Python 2.7 文件（不须要查看该目录）
- python25 – Python 2.5 文件（不须要查看该目录）
- python26 – Python 2.6 文件（不须要查看该目录）
- setup.py – Python 的安装文件 – 用于在您自己的 Python 环境中安装 gurobipy 模块

Linux 文件结构

下面的文件和目录会在您的安装目录（对于 64-位的 Linux 发布，通常为 */opt/gurobi460/linux64*）中创建：

- EULA.pdf – Gurobi 最终用户许可协议 – PDF 格式
- ReleaseNotes.html – 发布说明
- bin
  - grbd – Gurobi 许可证管理器的可执行文件
  - grbgetkey – 从 Gurobi 密钥服务器中检索 Gurobi 许可证密钥
  - grbprobe – 探测系统细节（通常不会使用）
  - grbvalidate – 刷新教学许可证（通常不会使用）
  - gurobi_cl – 简单的命令行二进制文件
  - gurobi.env – 参数初始化示例文件
  - gurobi.sh – 启动 Gurobi 交互式命令解释器
  - python2.7 – Python 解释器
- docs

  - examples – 示例导览（打开该目录中的 index.html 文件）
  - quickstart – 快速入门指南（打开该目录中的 index.html 文件）
  - refman – 参考手册（打开该目录中的 index.html 文件）
- examples
  - build – C、C++、Java、和 Python 示例的 Makefile 文件
  - c – C 示例的源代码
  - c# – C# 示例的源代码（适用于 Windows）
  - c++ – C++ 示例的源代码
  - data – 示例的数据文件
  - java – Java 示例的源代码
  - python – Python 示例的源代码
  - vb – Visual Basic 示例的源代码（适用于 Windows）
- include
  - gurobi_c.h – C 头文件
  - gurobi_c++.h – C++ 头文件
  - Python 2.7 – 虚拟的 Python 头文件（不须要查看该目录）
- lib
  - gurobi.jar – Java 接口
  - gurobi.py – Python 的启动文件
  - libgurobi46.so – Gurobi 库（当前版本的符号链接）
  - libgurobi_c++.a – C++ 接口（符号链接）
  - libgurobi_g++4.1.a – C++ 接口（g++ 4.1 时使用 – 例如，Red Hat 5 系统）
  - libgurobi_g++4.2.a – C++ 接口（适用于 g++ 4.2 或之后的版本）
  - libGurobiJni46.so – Java JNI 包装
  - libgurobi.so.4.6.0 – Gurobi 本地库（适用于所有的接口）
  - python2.7 – 交互式命令解释器和 Python 接口所使用的 Python 文件（不须要查看该目录）
  - python2.5 – Python 2.5 文件（不须要查看该目录）
  - python2.5_utf16 – UTF-16 Python 版本中所使用的 Python 2.5 文件（不须要查看该目录）
  - python2.6 – Python 2.6 文件（不须要查看该目录）
  - python2.6_utf16 – UTF-16 Python 版本中所使用的 Python 2.6 文件（不须要查看该目录）
  - python2.7_utf16 – UTF-16 Python 版本中所使用的 Python 2.7 文件（不须要查看该目录）
- setup.py – Python 的安装文件 – 用于在您自己的 Python 环境中安装 gurobipy 模块

Mac OS 文件结构

下面的文件和目录会在您的安装目录（通常为 */Library/gurobi460/mac64*）中创建：

- EULA.pdf – Gurobi 最终用户许可协议 – PDF 格式
- ReleaseNotes.html – 发布说明
- bin
  - grbd – Gurobi 许可证管理器的可执行文件
  - grbgetkey – 从 Gurobi 密钥服务器中检索 Gurobi 许可证密钥

  - grbprobe – 探测系统细节（通常不会使用）
  - grbvalidate – 刷新教学许可证（通常不会使用）
  - gurobi.env – 参数初始化示例文件
  - gurobi.sh – 启动 Gurobi 交互式命令解释器
  - gurobi_cl – 简单的命令行二进制文件
- docs
  - examples – 示例导览（打开该目录中的 index.html 文件）
  - quickstart – 快速入门指南（打开该目录中的 index.html 文件）
  - refman – 参考手册（打开该目录中的 index.html 文件）
- examples
  - build – C、C++、Java、和 Python 示例的 Makefile 文件
  - c – C 示例的源代码
  - c# – C# 示例的源代码（适用于 Windows）
  - c++ – C++ 示例的源代码
  - data – 示例的数据文件
  - java – Java 示例的源代码
  - python – Python 示例的源代码
  - vb – Visual Basic 示例的源代码（适用于 Windows）
- include
  - gurobi_c.h – C 头文件
  - gurobi_c++.h – C++ 头文件
- lib
  - gurobi.jar – Java 接口
  - gurobi.py – Python 的启动文件
  - gurobipy – 交互式命令解释器和 Python 接口所使用的 Python 文件（不须要查看该目录）
  - libGurobiJni46.jnilib – Java JNI 包装
  - libgurobi.so.4.6.0 – Gurobi 本地库（适用于所有的接口)
  - libgurobi46.so – Gurobi 库（当前版本的符号链接）
  - libgurobi_c++.a – C++ 接口（符号链接）
  - libgurobi_g++4.2.a – C++ 接口
- setup.py – Python 的安装文件 – 安装程序用于在您的 Python 环境中安装 gurobipy 模块

页面：Gurobi Optimizer Example Tour

Gurobi 优化器示例导览

版本4.6，Copyright © 2011，Gurobi Optimization，Inc.

Gurobi^{TM} 的发布包中包含了许多示例，用于展示 Gurobi 库的常用功能。其中大部分的示例都有 C、C++、C#、Java、Visual Basic、以及 Python 等不同语言的版本。然而，也有一些示例专门用于展示 Python 接口所特有的功能。

本文为这些示例提供了一个简要的导览。我们不会详细地介绍其中每一个示例。相反，我们首先会对您通过 Gurobi 优化器可以执行的一系列任务进行一个概述。之后的小节将介绍特定的示例如何完成其中每一个任务。我们建议您在阅读本文的同时查看一下示例的源代码（可以在文本编辑器中，也可以通过其他的浏览器窗口查看）。本文包含了所有示例的源代码（所有支持的语言版本）。源文件也可以在 Gurobi 发布包的 *examples* 目录中找到。

如果您想要进一步详细了解这些示例中所使用的 Gurobi 方法，请参见 Gurobi 参考手册。

- feasopt_c.c
- fixanddive_c.c
- lp_c.c
- lpmethod_c.c
- lpmod_c.c
- mip1_c.c
- mip2_c.c
- params_c.c
- qp1_c.c
- sensitivity_c.c
- sos_c.c
- sudoku_c.c
- workforce1_c.c
- workforce2_c.c
- workforce3_c.c
- workforce4_c.c

- C++ 示例
  - callback_c++.cpp
  - diet_c++.cpp
  - facility_c++.cpp
  - feasopt_c++.cpp
  - fixanddive_c++.cpp
  - lp_c++.cpp
  - lpmethod_c++.cpp
  - lpmod_c++.cpp

    - Sos.java
    - Sudoku.java
    - Workforce1.java
    - Workforce2.java
    - Workforce3.java
    - Workforce4.java
  - C# 示例
    - callback_cs.cs
    - diet_cs.cs
    - facility_cs.cs
    - feasopt_cs.cs
    - fixanddive_cs.cs
    - lp_cs.cs
    - lpmethod_cs.cs
    - lpmod_cs.cs
    - mip1_cs.cs
    - mip2_cs.cs
    - params_cs.cs
    - qp1_cs.cs
    - sensitivity_cs.cs
    - sos_cs.cs
    - sudoku_cs.cs
    - workforce1_cs.cs
    - workforce2_cs.cs
    - workforce3_cs.cs

- diet2.py
- diet3.py
- diet4.py
- dietmodel.py
- facility.py
- feasopt.py
- fixanddive.py
- lp.py
- lpmethod.py
- lpmod.py
- mip1.py
- mip2.py
- netflow.py
- params.py
- qp1.py
- sensitivity.py
- sos.py
- sudoku.py
- workforce1.py
- workforce2.py
- workforce3.py
- workforce4.py

页面：Example tour

示例导览

本文为 Gurobi 示例提供了一个快速的导览。我们试图去高亮显示这些示例中一些最为重要的特性。本文中还提供了这些示例的完整源代码，所以您可以自由地查看全部细节。

只要有可能，我们就会试图以一种与编程语言无关的方式来讨论这些示例。对于每一个示例，我们都通过一个简要的、与语言无关的名称对其进行引用。您可以将这一示例名称与特定语言的源文件名称进行映射。例如，*facility* 示例对应了六种不同的实现：C 语言版本（facility_c.c）、C++ 版本（facility_c++.cpp）、Java 版本（Facility.java）、C# 版本（facility_cs.cs）、Visual Basic 版本（facility_vb.vb）、以及 Python 版本（facility.py）。如果您想要查看某个特定示例的语言实现，请参见对应的示例源文件。

Gurobi 示例

下面是 Gurobi 发布包中包含的所有示例的列表。这些示例的源代码可以在发布包的 *examples* 目录、或在本文的源代码节中找到。

- **callback**——展示如何使用 Gurobi 回调。

- **diet**——生成并求解经典的食谱问题。展示模型构造和简单的模型修改——初始模型求解之后，添加一个约束来限制乳制品的份数。

- **diet2**、**diet3**、**diet4**、**dietmodel**——Python diet 示例的修改版本，用于展示 Python 中的模型-数据分离。

- **facility**——简单的设施选址模型：给定一些工厂与一些仓库、以及它们之间交通运输的成本，该示例选择开放一些工厂以满足产品的需求并使成本最低。该示例展示如何使用 MIP 开始解——示例计算得到一个初始的启发式解，并将这个解传入 MIP 求解器。

- **feasopt**——从文件中读取一个 MIP 模型，添加人工松弛变量来放宽所有的约束，然后最小化人工变量的个数。该示例通过添加松弛变量展示了如何进行简单的模型修改。

- **fixanddive**——实现了一个简单的 MIP 启发式算法。该示例从文件中读取一个 MIP 模型，放宽整性条件，然后求解松弛解。示例之后会从松弛解中挑选一组已经取整数值或接近整数值的整数变量，将它们的值固定为最接近的那个整数，并重新求解松弛解。重复该步骤直到所求得的松弛解整数可行或者线性不可行。该示例展示了不同类型的模型修改（放宽整性条件、修改变量边界，等等）。

- **lp**——一个非常简单的示例，从文件中读取一个连续模型，对它进行优化，并将解决方案写入文件。如果模型不可行，则改为将不可约不一致子系统（IIS）写入文件。
- **lpmethod**——展示如何使用不同的 LP 算法。从文件中读取一个连续模型，使用不同的算法对它进行求解，并报告对于该模型来说哪一种算法最快。
- **lpmod**——展示在 LP 中如何使用开始解。从文件中读取一个连续模型，对它进行求解，然后修改一个变量的边界。示例以两种不同的方式对修改之后的模型进行求解：从原来模型的解开始，或从头开始重新求解。
- **mip1**——生成一个简单的 MIP 模型，对它进行求解，并打印解决方案的值。
- **mip2**——从文件中读取一个 MIP 模型，对它进行优化，然后求解该 MIP 模型的固定版本模型。
- **netflow**——一个求解多商品网络流动模型的 Python 示例。该示例展示了多种不同 Python 建模结构的使用，包括字典、元组、以及 tuplelist 对象。
- **params**——展示如何使用 Gurobi 参数。从文件中读取一个 MIP 模型，然后花 5 秒时间分别使用 *MIPFocus* 参数的四个不同的值对模型进行进行求解。该示例会比较四次不同优化运行的最优差距，并选择产生最小差距的那个 *MIPFocus* 参数值继续求解。
- **qp1**——生成一个简单的 QP 模型，对它进行求解，将其转换为一个 MIQP 模型，并再次对它进行求解。
- **sensitivity**——MIP 的灵敏度分析。读取一个 MIP 模型，对它进行求解，然后计算将模型中每一个二元变量都固定为 0 或 1 时对于目标的影响。通过修改变量边界来展示如何进行简单的 MIP 模型修改。
- **sos**——生成并求解一个简单的 SOS 模型。
- **Sudoku**——从文件中读取一组 Sudoku 谜题集，生成一个对应的 MIP 模型，对它进行求解，并打印解决方案的值。
- **workforce1**——规划并求解一个人力调度的模型。如果模型不可行，该示例就会计算并打印不可约不一致子系统（IIS）。
- **workforce2**——*workforce1* 的一个增强版本。该示例会对相同的人力调度模型进行求解，但如果模型不可行，它就会计算 IIS，从模型中删除一个相关联的属性，并重新求解。重复该步骤直到模型变为可行。展示了如何删除约束。

- **workforce3**——*workforce1* 的另一个增强版本。该示例会对相同的人力调度模型进行求解，但如果模型不可行，它就会在每一个约束中都添加人工变量并最小化人工变量的个数。这相当于寻找为使模型变得可行而所需的右侧向量的最小改动。展示了如何添加变量。

- **workforce4**——*workforce3* 的一个增强版本。该示例对相同的人力调度模型进行求解，但会从每个约束的人工变量开始。该示例首先最小化人工变量的个数。然后，它引入一个新的二次目标来平衡工人之间的工作量。展示了如何优化多个目标函数。

示例导览

对于 Gurobi 示例的介绍可以从最简单的示例——从文件中加载并求解模型开始。这些示例展示了 Gurobi 库中最为基本的功能。同时它们也展示了如何使用 Gurobi 优化器中一个非常重要的概念：模型属性。

一旦熟悉了这些示例，您就可以进一步关注那些从头开始生成模型的示例。它们演示了如何创建变量和约束，并将它们添加到优化模型中去。同时这些示例也说明了*懒惰更新*是如何工作的，要使用 Gurobi 库，您就须要理解这一概念。

本文所涵盖的下一个主题是模型的修改。Gurobi 发布包中包含了一些示例用于演示如何添加并删除约束、添加变量、以及修改变量的类型、边界、和目标系数。修改模型的过程在很大程度上与从头开始生成一个模型十分相似，但也有一些重要的区别，包括解信息的使用。

接下来，本文会介绍参数的修改。*params* 示例展示了如何对参数进行修改，特别是对于不同的模型如何使用不同的参数设置。

不可行问题节给出了一些示例，说明如何处理不可行的模型。一些示例使用不可约不一致子系统（IIS）来处理不可行问题；而另一些则放宽了约束。

MIP 开始解是一个十分有用的 MIP 特性，须要理解它的作用。MIP 开始解允许您为 MIP 求解器指定一个已知可行的解。这个可行解为最佳解决方案的目标值提供了边界，从而有效地限制了 MIP 搜索。它还为 Gurobi MIP 求解器所使用的局部搜索启发式算法提供了一个潜在的起点。

Python 接口可以帮助您实现建模语言中常见的模型-数据分离，但是您必须使用 Python 模块才能做到。模型-数据分离节中通过一个示例来演示如何实现模型与数据的分离。它包含了两个不同版本的 diet 示例，分别从不同的地方获取模型数据。然而，这两个示例却使用了相同的文件来建立并求解实际的优化模型。

本文中我们将要涵盖的最后一个主题是 Gurobi 回调。回调允许用户定期地获取与优化相关的进度信息。

小节

页面：Load and solve a model from a file

从文件中加载并求解模型

示例：callback、feasopt、fixanddive、lp、lpmethod、lpmod、mip2、params、sensitivity

通过 Gurobi 库可以执行的最为基本的任务之一就是从文件中读取一个模型，对它进行优化，并且报告优化的结果。*lp*（lp_c.c、lp_c++.cpp、lp_cs.cs、Lp.java、lp.py、lp_vb.vb）和 *mip2*（mip2_c.c、mip2_c++.cpp、mip2_cs.cs、Mip2.java、mip2.py、mip2_vb.vb）示例都说明了如何在 Gurobi 所支持的各种语言中做到这一点。尽管在不同的语言中实现的细节也会有所不同，但对所有语言来说基本结构都是一致的。

初始化 Gurobi 环境之后，示例就可以开始从指定的文件中读取模型。在 C 语言中，须要调用 *GRBreadmodel()* 函数：

```
  error = GRBreadmodel(masterenv, argv[1], &model);
```

在 C++ 中，可以通过构造一个 *GRBModel* 对象来实现：

```
  GRBModel model = GRBModel(env, argv[1]);
```

同样地，在 C# 和 Java 中也可以通过构造一个 *GRBModel* 对象来实现：

```
  GRBModel model = new GRBModel(env, args[0]);
```

在 Python 中，通过 *read* 全局函数来实现：

```
  model = read(sys.argv[1])
```

下一步就是调用 Gurobi 优化器来求解模型。在 C 语言中，可以调用 *model* 变量的 *GRBoptimize()* 函数：

```
  error = GRBoptimize(model);
```

在 C++、Java、和 Python 中，则是通过调用 *model* 对象的 *optimize* 方法来实现的：

```
  model.optimize();
```

在 C# 中，方法名称的首字母须要大写：

```
  model.Optimize();
```

成功的 *optimize* 方法调用会填充模型中一系列的解属性。例如，调用完成之后，*X* 变量属性就包含了每个变量的解决方案的值。相似地，对于连续模型，*Pi* 约束属性会包含每个约束的双重值。

示例之后检索了模型的 *Status* 属性从而判断优化的结果。在 *lp* 示例中，会将最优解写入一个解文件（*model.sol*）。

一旦读取并求解了模型，就可以做许多其他的事情。例如，*lp* 示例检查了解的状态——这是强烈推荐的做法。如果发现模型是不可行的，该示例就会计算不可约不一致子系统（IIS）来隔离导致不可行的源头。

页面：Build a model

模型的生成

示例：diet、facility、mip1、qp1、sos、sudoku、workforce1、workforce2、workforce3、workforce4

有一些 Gurobi 示例是从头开始生成模型的。我们首先讨论其中的两个：*mip1* 和 *sos*，它们生成了非常简单的模型，可以用于展示模型生成的基本过程。

通常来说，生成模型的第一步是创建一个空的模型。在 C 语言中可以通过 *GRBnewmodel* 函数来实现：

```
  error = GRBnewmodel(env, &model, "mip1", 0, NULL, NULL, NULL, NULL);
```

在创建模型的同时可以选择性地创建一组变量，并指定变量边界、目标系数、以及这些变量的名称。在这些示例中，新变量是另外添加的。

在 C++、C#、和 Java 中，通过 *GRBModel* 构造函数可以创建一个新的模型。在 Java 中，方法调用如下所示：

```
  GRBModel model = new GRBModel(env);
```

在 Python 中，模型类称为 *Model*，它的构造函数与 C++ 和 Java 中 *GRBModel* 的构造函数相似。

模型一旦创建之后，下一步通常就是添加变量。在 C 语言中，通过 *GRBaddvars* 函数来添加一个或多个变量：

```
  error = GRBaddvars(model, 3, 0, NULL, NULL, NULL, obj, NULL, NULL,
vtype, NULL);
```

在 C++、Java、和 Python 中，通过 *Model* 对象的 *addVar* 方法（C# 中的 *AddVar* 方法）来添加变量。在 Java 中，方法调用如下所示：

```
  GRBVar x = model.addVar(0.0, 1.0, -1.0, GRB.BINARY, "x");
```

新变量的下界值、上界值、目标系数、类型、以及名称都可以作为方法的参数指定。在 C++ 和 Python 中，可以忽略这些参数而使用默认值。更多信息请参见 Gurobi 参考手册。

在模型中添加变量之后，下一步就是调用更新函数（C 语言中的 *GRBupdatemodel()* 函数，C++、Java、和 Python 中的 *model.update()* 方法，以及 C# 中的 *model.Update()* 方法）。在 Gurobi 优化器中，对于模型的修改是以一种*懒惰*方式来执行的——直到下一次调用更新或优化函数时这些修改才会真正影响模型。所以在调用更新函数之前还不能（例如，在约束中）使用这些新添加的变量。

下一步是在模型中添加约束。在 C 语言中线性约束可以通过 *GRBaddconstr* 函数添加：

```
  error = GRBaddconstr(model, 3, ind, val, GRB_LESS_EQUAL, 4.0, "c0");
```

在 C 语言中添加线性约束时，必须指定一组约束左侧端的变量索引和系数、约束含义（例如，GRB_LESS_EQUAL）、以及约束右侧端常量。此外，还可以为约束起一个名字；如果省略约束名称，Gurobi 则会为约束指定一个默认名称。

在 C++、C#、Java、和 Python 中，生成线性约束时首先须要为约束的左侧和右侧端生成线性表达式。 Java 中不支持运算符重载，可以通过以下方法来生成表达式：

```
  GRBLinExpr expr = new GRBLinExpr();
  expr.addTerm(1.0, x); expr.addTerm(2.0, y); expr.addTerm(3.0, z);
```

之后可以使用 *Model* 对象的 *addConstr* 方法使用这些左侧和右侧端的线性表达式来添加一个约束：

```
  model.addConstr(expr, GRB_LESS_EQUAL, 4.0, "c0");
```

C++、C#、和 Python 中重载了诸如 +、*、<= 之类的标准数学运算符，从而使得线性表达式与传统的数学表达式十分相似。在 C++ 中，生成线性约束的代码如下所示：

```
  model.addConstr(x + 2 * y + 3 * z <= 4, "c0");
```

添加特殊有序集（SOS）约束是十分相似的。在 C 语言中，通过 *GRBaddsos* 函数来添加一个或多个 SOS 约束：

```
  error = GRBaddsos(model, 1, 2, sostype, sosbeg, sosind, soswt);
```

对于每个 SOS 约束，都必须指定一系列成员以及每个成员的权重。

在 C++、C#、Java、和 Python 中，可以调用 *Model* 对象的 *addSOS* 方法：

```
  model.addSOS(sosv1, soswt1, GRB.SOS_TYPE1);
```

一旦生成了模型，下一步通常就是对它进行优化（通过 C 语言中的 *GRBoptimize* 函数，C++、Java、和 Python 中的 *model.optimize* 方法，或者 C# 中的 *model.Optimize* 方法）。然后就可以查询变量的 *X* 属性来检索解的值（以及通过 *VarName* 属性来检索每个变量的变量名称）。在 C 语言中，*X* 属性的检索如下所示：

```
  error = GRBgetdblattrarray(model, GRB_DBL_ATTR_X, 0, 3, sol);
```

C++ 中的代码如下所示：

```
  cout << x.get(GRB_StringAttr_VarName) << " "
       << x.get(GRB_DoubleAttr_X) << endl;
  cout << y.get(GRB_StringAttr_VarName) << " "
       << y.get(GRB_DoubleAttr_X) << endl;
  cout << z.get(GRB_StringAttr_VarName) << " "
       << z.get(GRB_DoubleAttr_X) << endl;
```

Java 中的代码如下所示：

```
  System.out.println(x.get(GRB.StringAttr.VarName) +
                     " " + x.get(GRB.DoubleAttr.X));
  System.out.println(y.get(GRB.StringAttr.VarName) +
                     " " + y.get(GRB.DoubleAttr.X));
  System.out.println(z.get(GRB.StringAttr.VarName) +
                     " " + z.get(GRB.DoubleAttr.X));
```

C# 中的代码如下所示：

```
  Console.WriteLine(x.Get(GRB.StringAttr.VarName) +
                    " " + x.Get(GRB.DoubleAttr.X));
  Console.WriteLine(y.Get(GRB.StringAttr.VarName) +
                    " " + y.Get(GRB.DoubleAttr.X));
  Console.WriteLine(z.Get(GRB.StringAttr.VarName) +
                    " " + z.Get(GRB.DoubleAttr.X));
```

Python 中的代码如下所示：

```
  for v in m.getVars():
    print v.varName, v.x
```

当查询或修改约束或变量数组的属性值时，对于整个数组所执行的操作通常更有效率。对于 C 语言接口这是十分自然的，因为大多数的属性函数都带有数组参数。在 C++、C#、和 Java 接口中，可以通过 *Model* 对象的 *get* 和 *set* 方法来直接操作属性值的数组。在 *sudoku* 的 Java 示例中，存在以下代码：

```
  double[][][] x = model.get(GRB.DoubleAttr.X, vars);
```

请注意，Python 接口没有提供任何数组查询的方法。Python 是一种解释语言，所以与单个属性查询相关联的开销对于整体运行来说不会有太大的影响。

页面：Modify a model

模型的修改

示例：diet、feasopt、fixanddive、lpmod、sensitivity、workforce3、workforce4

本节将介绍模型的修改。模型修改可以包含多种形式，例如约束或变量的添加与删除、约束和变量属性的修改、约束系数的改动，等等。以下是一些示例，展示如何进行不同类型的模型修改。

diet

该示例生成了一个线性模型以求解经典的食谱问题：寻找一份成本最低的食谱以满足一系列的日常营养需求。一旦规划并求解了模型，该示例会在模型中添加一个额外的约束以限制乳制品的份数，并且再次对模型进行求解。让我们一起关注如何对模型进行修改。

在一个已经求解的模型中添加约束其实和在构造初始模型时添加约束没有什么区别。在 Python 中，可以通过以下约束来引入乳制品的份数限制（6 份）：

```
  m.addConstr(buy[7] + buy[8] <= 6, "limit_dairy")
```

对于线性模型，之前计算得到的解决方案可以作为修改之后模型的一个有效的*热启动解*。Gurobi 求解器保留了之前的解，因此下一次的*优化*调用会自动从之前的解决方案开始。

lpmod

变量边界的修改同样也是十分简单的。*lpmod* 示例修改了单个变量的边界，然后以两种不同的方式重新求解模型。变量边界可以通过变量的 *UB* 或 *LB* 属性来修改。在 C 语言中，函数调用如下所示：

```
  error = GRBsetdblattrelement(model, GRB_DBL_ATTR_UB, var, 0.0);
```

Python 中的代码如下所示：

```
  minVar.ub = 0
```

简单地再次调用 *optimize* 方法对模型进行重新求解。对于连续模型，重新求解是从之前的解决方案开始的。为了说明从初始的、未求解的状态开始对模型进行求解有什么样的区别，*lpmod* 示例调用了 *reset* 函数。在 C 语言中，函数调用如下所示：

```
  error = GRBresetmodel(model);
```

C++、Java、和 Python 中的代码如下所示：

```
  m.reset()
```

C# 中的代码如下所示：

```
  m.Reset()
```

当我们在重置模型之后再调用 optimize 方法时，就会从头开始重新优化。尽管对于这样简单的示例来说在计算时间上的差异并不是十分显著，但是对于大型模型，热启动可以极大地提高优化的速度。

fixanddive

fixanddive 示例同样也展示了如何对变量边界进行修改。在该示例中，我们反复修改一组变量的边界值，并且每次都通过热启动重新优化。在 C 语言中，固定变量的代码如下所示：

```
  for (j = 0; j < nfix; ++j)
  {
    fixval = floor(fractional[j].X + 0.5);
    error = GRBsetdblattrelement(model, "LB", fractional[j].index,
fixval);
    if (error) goto QUIT;
    error = GRBsetdblattrelement(model, "UB", fractional[j].index,
fixval);
    if (error) goto QUIT;
  }
```

Python 中的代码如下所示：

```
  for i in range(nfix):
    v = fractional[i]
    fixval = int(v.x + 0.5)
    v.lb = fixval
    v.ub = fixval
```

同样地，之后对于 *optimize* 的调用会从之前的解开始优化。

sensitivity

sensitivity 示例会计算当所有的二元变量都固定为 0 或者 1 时最优目标值的改变量。对于每一个二元变量，该示例都会以新的上下边界值创建一个模型副本并对它进行求解。由于该示例求解的是一个 MIP 模型，Gurobi 无法使用高级开始解信息。所以每次变量边界修改之后模型都会从头开始重新开始求解。

feasopt

我们这里要讨论的最后一个关于模型修改的示例是 *feasopt*。该示例展示如何在现有的约束中添加变量，以及如何修改优化目标。将目标函数设置为零十分简单：调用 *setObjective* 函数并将输入参数设置为零即可：

```
  m.setObjective(0)
```

添加新变量会稍微复杂一些。在该示例中，我们想要在每个约束中都添加人工变量来放宽约束。对于等式约束，我们须要使用两个人工变量；而对于所有其他的约束使用一个就可以了。在约束 *c* 中添加单个人工变量的 Python 代码如下所示：

```
  feasModel.addVar(obj=1.0, name="ArtP_" + c.Constrname,
column=Column([1], [c]))
```

我们使用 *addVar* 方法中的 *column* 参数来指定包含新变量的那组约束、以及相关联的系数。在该示例中，只有须要放宽的约束才会包含这个新变量。在这里除了目标系数和变量名称，其他变量属性都取默认值。

页面：Change parameters

参数的修改

示例：callback、fixanddive、lp、lpmethod、mip2、params、sensitivity

本节将展示如何使用 Gurobi 参数。示例 *params* 从文件中读取一个 MIP 模型、然后使用 *MIPFocus* 参数的四个不同的值分别对该模型进行求解、每个值运行五秒钟时间（*MIPFocus* 参数用于选择 MIP 求解器所使用的高级求解策略）。然后该示例会选择产生最小 MIP 差距的那个参数值，继续对模型进行求解直至其达到最优。

设置参数的方法十分简单。在 C 语言中设置 *MIPFocus* 参数的代码如下所示：

```
  GRBsetintparam(GRBgetenv(model), GRB_INT_PAR_MIPFOCUS, i);
```

C++ 中的代码如下所示：

```
 model.getEnv().set(GRB_IntParam_MIPFocus, i);
```

Java 中的代码如下所示：

```
  model.getEnv().set(GRB.IntParam.MIPFocus, i);
```

C# 中的代码如下所示：

```
  model.GetEnv().Set(GRB.IntParam.MIPFocus, i);
```

Python 中的代码如下所示：

```
  model.params.MIPFocus = i
```

请注意参数设置是如何影响不同模型的行为的。当我们设置基本模型的 *TimeLimit* 参数，并且生成该模型的一个副本时，参数设置会传递到副本之中。副本模型所拥有的环境是原来环境的一个副本。当我们设置副本模型的 *MIPFocus* 参数时，该参数修改对于原来的模型或者其他的副本模型都没有影响。

页面：Diagnose and cope with infeasibility

不可行问题的诊断与处理

示例：feasopt、lp、workforce1、workforce2、workforce3、workforce4

在求解优化模型时，有一些情况下指定的约束条件无法满足。当这种情况发生时，通常须要发现并修正导致问题不可行的根本原因、或者找到一系列须要放宽的约束从而使模型变为可行。*workforce1*、*workforce2*、以及 *workforce3* 都展示了这些不同的策略。

从它们之中最为简单的示例开始，*workforce1* 规划了一个简单的人力调度模型并对它进行求解。如果模型是不可行的，它就会计算该模型的不可约不一致子系统（IIS）。然后用户就可以看到这些信息，从而能够更好地理解并进一步解决模型中不可行性的根源。

workforce2 是一个十分相似的示例，惟一的不同在于如果模型是不可行的，该示例就会反复计算 IIS 并从模型中删除一个相关联的约束，直到模型变为可行。请注意，从 IIS 中删除一个约束就足够可以解决不可行性的那个根源，但是这一个 IIS 也许并不能描述不可行性的所有根源。因此须要重复这一步骤直到模型变为可行。

示例 *workforce3* 采取了一种不同的方法来解决问题的不可行性。它并不是去找到并删除 IIS 的成员，相反地，它允许放宽模型中的约束。和 *feasopt* 示例一样，它在每一个约束中都添加了人工变量。该示例将原有变量的目标系数都设置为零，然后求解模型以最小化约束松弛的总量。

页面：MIP starts

MIP 开始解

示例：facility

MIP 建模者通常知道如何去计算问题的可行解。如果 MIP 求解器在寻找初始可行解时进展十分缓慢，建模者就可以在建立模型时一并提供一个可行解，这对于求解器来说将是十分有用的。可以通过变量的 *Start* 属性做到这一点，正如 *facility* 示例中所展示的那样。

facility 示例求解了一个简单的设施选址问题。模型中包含了一些仓库、以及一些生产仓库所需产品的工厂。其中每个工厂都有其最大的产能和固定的经营成本。此外，将产品从工厂运输到每个仓库也都有相对应的成本。问题的目标是在给定的产能和成本前提下决定如何安排工厂进行生产从而满足产品的需求。

该示例使用了一种简单的启发式算法来选择初始解：关闭固定成本最高的那个工厂。与之相对应的解不一定是最优的，但它一定是 MIP 优化的一个合理的开始解。在开始优化之前通过设置 *Start* 属性将 MIP 初始解传入 MIP 求解器。在 C 语言中，我们通过以下代码来设置开始解属性，从而开放所有的工厂：

```
for (p = 0; p < nPlants; ++p)
{
  error = GRBsetdblattrelement(model, "Start", opencol(p), 1.0);
  if (error) goto QUIT;
}
```

Python 中的代码如下所示：

```
for p in range(nPlants)
  open[p].start = 1.0
```

当运行该示例时，MIP 求解器报告说开始解构成了一个可行的初始解：

```
Loaded MIP start with objective 210500
```

对于示例中的数据来说，该初始解证明是最优的。尽管对于这样简单的示例来说在计算时间上的差异并不是十分显著，但是提供一个好的开始解有时可以大大地帮助那些较难求解的模型。

请注意，该示例中的 MIP 开始解仅仅指定了部分变量的值——那些用于决定保留开放哪些工厂而关闭哪些工厂的变量。Gurobi MIP 求解器会使用所有提供了的开始信息以试图构建一个完整的解。

页面：Model-Data Separation in Python

Python 中的模型-数据分离

示例：diet2.py、diet3.py、diet4.py

在建模语言中生成优化模型时，通常会将优化模型本身与用于创建模型实例的数据进行分离。这两个模型元素通常会存储在完全不同的文件中。我们将通过 *diet2.py*、*diet3.py*、以及 *diet4.py* 等示例来演示如何在我们的 Python 接口中实现相似的功能。这些示例展示了为优化模型提供数据所常用的几种不同的方法：*diet2.py* 将数据包含在源文件中；*diet3.py* 从一个 SQL 数据库中读取数据（通过 Python 中的 *sqlite3* 组件）；而 *diet4.py* 则会从一个 Excel 数据表格中读取数据（通过 Python 中的 *xlrd* 组件）。*dietmodel.py* 文件中包含了优化模型本身的实现。diet2.py、diet3.py、以及 diet4.py 都会使用这个模型。

Python 中的模块是使模型能够与数据分离开来的关键结构 。模块其实就是存储在文件中的一组函数与变量，通过 *import* 语句可以将它们导入程序。*diet2.py*、*diet3.py*、以及 *diet4.py* 都是通过以下这些语句将准备好的模型数据变量传入 *dietmodel* 模块的 *solve* 函数：

```
import dietmodel
dietmodel.solve(categories, minNutrition, maxNutrition, foods, cost,
nutritionValues)
```

其中第一条语句导入了 *dietmodel* 模块，该模块必须存储在当前目录的 *dietmodel.py* 文件中；第二条语句将模型数据传入新导入的这个模块的 *solve* 函数。

页面：Callbacks

回调

示例：callback

我们所要讨论的最后一个示例是 *callback*，它展示了如何使用 Gurobi 的回调函数。回调可以用于报告优化的进度或者修改 Gurobi 优化器的行为。使用回调时，用户须要编写回调函数来实现所需的行为。回调函数会在优化开始时传入 Gurobi 优化器，并且在优化的过程当中定期地被调用。用户函数中有一个参数是 *where* 值，用于指定回调函数是在优化过程中的何处被调用的。用户回调函数可以通过优化库的调用来查询某些值。读者如果想要了解更多的细节，我们建议您参考 Gurobi 参考手册的回调节。

我们的回调示例实现了一种简单的终止方案：用户将一个节点数量传入回调函数，当达到该节点数量时回调即要求优化器终止优化。在 C 语言中实现的代码如下所示：

```
  GRBcbget(cbdata, where, GRB_CB_MIP_NODCNT, &nodecnt);
  if (nodecnt > limit)
    GRBterminate(model);
```

在 Python 中，是这样实现的：

```
  nodecnt = model.cbGet(GRB.callback.MIP_NODCNT)
  if nodecnt > model._mynodelimit:
    model.terminate()
```

用户回调函数可以调用 *cbget* 方法（在 C 语言中为 *GRBcbget* 函数，在 C++、C#、Java、和 Python 中为 *cbGet* 方法）来获取当前的节点数量。

我们的回调示例还打印了优化的进度信息。在 C 语言中的代码如下所示：

```
  GRBcbget(cbdata, where, GRB_CB_MIP_NODCNT, &nodecnt);
  if (nodecnt - mydata->lastmsg >= 100) {
    ...
    printf("%7.0f ...", nodecnt, ...);
  }
```

Python 中的代码如下所示：

```
  nodecnt = model.cbGet(GRB.callback.MIP_NODCNT)
  if nodecnt % 100 == 0:
    print int(nodecnt), ...
```

同样地，用户回调函数调用了 *cbGet* 方法来查询优化的状态。

页面：Example Source Code

示例源代码

在本节中我们包含了所有发布示例的源代码。相同的示例源代码也可以在 Gurobi 发布包的 *examples* 目录中找到。

小节

  - callback_c++.cpp
  - diet_c++.cpp
  - facility_c++.cpp
  - feasopt_c++.cpp
  - fixanddive_c++.cpp
  - lp_c++.cpp
  - lpmethod_c++.cpp
  - lpmod_c++.cpp
  - mip1_c++.cpp
  - mip2_c++.cpp
  - params_c++.cpp
  - sensitivity_c++.cpp
  - qp1_c++.cpp
  - sos_c++.cpp
  - sudoku_c++.cpp
  - workforce1_c++.cpp
  - workforce2_c++.cpp
  - workforce3_c++.cpp
  - workforce4_c++.cpp
- Java 示例
  - Callback.java
  - Diet.java
  - Facility.java
  - Feasopt.java
  - Fixanddive.java

  - Lp.java
  - Lpmethod.java
  - Lpmod.java
  - Mip1.java
  - Mip2.java
  - Params.java
  - Qp1.java
  - Sensitivity.java
  - Sos.java
  - Sudoku.java
  - Workforce1.java
  - Workforce2.java
  - Workforce3.java
  - Workforce4.java
- C# 示例
  - callback_cs.cs
  - diet_cs.cs
  - facility_cs.cs
  - feasopt_cs.cs
  - fixanddive_cs.cs
  - lp_cs.cs
  - lpmethod_cs.cs
  - lpmod_cs.cs
  - mip1_cs.cs
  - mip2_cs.cs

- sudoku.py
- workforce1.py
- workforce2.py
- workforce3.py
- workforce4.py

页面：C Examples

C 示例

本节包含了所有 Gurobi C 语言的示例源代码。相同的源代码也可以在 Gurobi 发布包的 *examples/c* 目录中找到。

小节

- callback_c.c
- diet_c.c
- facility_c.c
- feasopt_c.c
- fixanddive_c.c
- lp_c.c
- lpmethod_c.c
- lpmod_c.c
- mip1_c.c
- mip2_c.c
- params_c.c
- qp1_c.c
- sensitivity_c.c
- sos_c.c
- sudoku_c.c
- workforce1_c.c
- workforce2_c.c
- workforce3_c.c
- workforce4_c.c

页面：callback_c.c

callback_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP or a MIP from a file, sets a callback
   to monitor the optimization progress and to output some progress
   information to the screen and to a log file. If it is a MIP and 10%
   gap is reached, then it aborts */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "gurobi_c.h"


/* Define structure to pass my data to the callback function */

struct callback_data {
  double  lastmsg;
  FILE   *logfile;
  double *solution;
};

int __stdcall
mycallback(GRBmodel *model,
           void     *cbdata,
           int       where,
           void     *usrdata)
{
  struct callback_data *mydata = (struct callback_data *) usrdata;
  char *msg;

  if (where == GRB_CB_POLLING) {
    /* Ignore polling callback */
  } else if (where == GRB_CB_MESSAGE) {
    GRBcbget(cbdata, where, GRB_CB_MSG_STRING, &msg);
    fprintf(mydata->logfile, "%s", msg);
  } else if (where == GRB_CB_SIMPLEX) {
    double pinf, dinf, obj, itrcnt;
    int    ispert;
    char   ch;
    GRBcbget(cbdata, where, GRB_CB_SPX_ITRCNT, &itrcnt);
    if (((int) itrcnt)%100 == 1) {
      GRBcbget(cbdata, where, GRB_CB_SPX_ISPERT, &ispert);
      GRBcbget(cbdata, where, GRB_CB_SPX_OBJVAL, &obj);
      GRBcbget(cbdata, where, GRB_CB_SPX_PRIMINF, &pinf);
      GRBcbget(cbdata, where, GRB_CB_SPX_DUALINF, &dinf);
      if      (ispert == 0) ch = ' ';
      else if (ispert == 1) ch = 'S';
      else                  ch = 'P';
      printf("%7.0f %14.7e%c %13.6e %13.6e\n", itrcnt, obj, ch, pinf,
dinf);
    }
  } else if (where == GRB_CB_PRESOLVE) {
    int cdels, rdels;
    GRBcbget(cbdata, where, GRB_CB_PRE_COLDEL, &cdels);
    GRBcbget(cbdata, where, GRB_CB_PRE_ROWDEL, &rdels);
    if (cdels || rdels) {
      printf("%7d columns and %7d rows are removed\n", cdels, rdels);
    }
  } else if (where == GRB_CB_MIP) {
    double objbst, objbnd, nodecnt, actnodes, itrcnt;
    int    cutcnt, solcnt;
    GRBcbget(cbdata, where, GRB_CB_MIP_NODCNT, &nodecnt);
    if (nodecnt - mydata->lastmsg >= 100) {
      mydata->lastmsg = nodecnt;
      GRBcbget(cbdata, where, GRB_CB_MIP_OBJBST, &objbst);
      GRBcbget(cbdata, where, GRB_CB_MIP_OBJBND, &objbnd);
      /* if gap is small, then quit */
      if (fabs(objbst - objbnd) < 0.1 * (1.0 + fabs(objbst)))
        GRBterminate(model);
      GRBcbget(cbdata, where, GRB_CB_MIP_NODLFT, &actnodes);
      GRBcbget(cbdata, where, GRB_CB_MIP_ITRCNT, &itrcnt);
      GRBcbget(cbdata, where, GRB_CB_MIP_SOLCNT, &solcnt);
      GRBcbget(cbdata, where, GRB_CB_MIP_CUTCNT, &cutcnt);
      printf("%7.0f %7.0f %8.0f %13.6e %13.6e %7d %7d\n",
             nodecnt, actnodes, itrcnt, objbst, objbnd, solcnt, cutcnt);
    }
  } else if (where == GRB_CB_MIPSOL) {
    double obj, nodecnt;
    int solcnt;
    GRBcbget(cbdata, where, GRB_CB_MIPSOL_OBJ, &obj);
    GRBcbget(cbdata, where, GRB_CB_MIPSOL_NODCNT, &nodecnt);
    GRBcbget(cbdata, where, GRB_CB_MIPSOL_SOLCNT, &solcnt);
    GRBcbget(cbdata, where, GRB_CB_MIPSOL_SOL, mydata->solution);

    printf("**** New solution at node %.0f, obj %g, sol %d, x[0] = %.2f
****\n",
           nodecnt, obj, solcnt, mydata->solution[0]);
  } else if (where == GRB_CB_BARRIER) {
    double primobj, dualobj, priminf, dualinf, compl;
    int iter;
    GRBcbget(cbdata, where, GRB_CB_BARRIER_ITRCNT,  &iter);
    GRBcbget(cbdata, where, GRB_CB_BARRIER_PRIMOBJ, &primobj);
    GRBcbget(cbdata, where, GRB_CB_BARRIER_DUALOBJ, &dualobj);
    GRBcbget(cbdata, where, GRB_CB_BARRIER_PRIMINF, &priminf);
    GRBcbget(cbdata, where, GRB_CB_BARRIER_DUALINF, &dualinf);
    GRBcbget(cbdata, where, GRB_CB_BARRIER_COMPL,   &compl);

    printf("Barrier iter: %d %.4e %.4e %.4e %.4e %.4e\n",
           iter, primobj, dualobj, priminf, dualinf, compl);
  }
  return 0;
}

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
```

```
int       error = 0;
int       vars, optimstatus;
double    objval;
struct callback_data mydata;

mydata.lastmsg  = -GRB_INFINITY;
mydata.logfile  = NULL;
mydata.solution = NULL;

if (argc < 2) {
  fprintf(stderr, "Usage: callback_c filename\n");
  goto QUIT;
}

mydata.logfile = fopen("cb.log", "w");
if (!mydata.logfile) {
  fprintf(stderr, "Cannot open cb.log for callback message\n");
  goto QUIT;
}

/* Create environment */

error = GRBloadenv(&env, NULL);
if (error || env == NULL) {
  fprintf(stderr, "Error: could not create environment\n");
  exit(1);
}

/* Turn off display */

error = GRBsetintparam(env, GRB_INT_PAR_OUTPUTFLAG, 0);
if (error) goto QUIT;

/* Read model from file */

error = GRBreadmodel(env, argv[1], &model);
if (error) goto QUIT;

/* Allocate space for solution */

error = GRBgetintattr(model, GRB_INT_ATTR_NUMVARS, &vars);
if (error) goto QUIT;

mydata.solution = malloc(vars*sizeof(double));
if (mydata.solution == NULL) {
  fprintf(stderr, "Failed to allocate memory\n");
  exit(1);
}

/* Set callback function */

error = GRBsetcallbackfunc(model, mycallback, (void *) &mydata);
if (error) goto QUIT;

/* Solve model */

error = GRBoptimize(model);
```

```
  if (error) goto QUIT;

  /* Capture solution information */

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;

  error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
  if (error) goto QUIT;

  printf("\nOptimization complete\n");
  if (optimstatus == GRB_OPTIMAL)
    printf("Optimal objective: %.4e\n", objval);
  else if (optimstatus == GRB_INF_OR_UNBD)
    printf("Model is infeasible or unbounded\n");
  else
    printf("Optimization was stopped early\n");
  printf("\n");

QUIT:

  /* Error reporting */

  if (error) {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Close file */

  if (mydata.logfile)
    fclose(mydata.logfile);

  /* Free solution */

  if (mydata.solution)
    free(mydata.solution);

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：diet_c.c

diet_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Solve the classic diet model, showing how to add constraints
   to an existing model. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "gurobi_c.h"

int printSolution(GRBmodel* model, int nCategories, int nFoods);


int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0;
  int       i, j;
  int      *cbeg, *cind, idx;
  double   *cval, *rhs;
  char     *sense;

  /* Nutrition guidelines, based on
     USDA Dietary Guidelines for Americans, 2005
     http://www.health.gov/DietaryGuidelines/dga2005/ */

  const int nCategories = 4;
  char *Categories[] =
    { "calories", "protein", "fat", "sodium" };
  double minNutrition[] = { 1800, 91, 0, 0 };
  double maxNutrition[] = { 2200, GRB_INFINITY, 65, 1779 };

  /* Set of foods */
  const int nFoods = 9;
  char* Foods[] =
    { "hamburger", "chicken", "hot dog", "fries",
      "macaroni", "pizza", "salad", "milk", "ice cream" };
  double cost[] =
    { 2.49, 2.89, 1.50, 1.89, 2.09, 1.99, 2.49, 0.89, 1.59 };

  /* Nutrition values for the foods */
  double nutritionValues[][4] = {
                                 { 410, 24, 26, 730 },
                                 { 420, 32, 10, 1190 },
                                 { 560, 20, 32, 1800 },
                                 { 380, 4, 19, 270 },
                                 { 320, 12, 10, 930 },
                                 { 320, 15, 12, 820 },
                                 { 320, 31, 12, 1230 },
                                  { 100, 8, 2.5, 125 },
                                  { 330, 8, 10, 180 }
                                };

  /* Create environment */
  error = GRBloadenv(&env, "diet.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create initial model */
  error = GRBnewmodel(env, &model, "diet", nFoods + nCategories,
                      NULL, NULL, NULL, NULL, NULL);
  if (error) goto QUIT;

  /* Initialize decision variables for the foods to buy */
  for (j = 0; j < nFoods; ++j)
  {
    error = GRBsetdblattrelement(model, "Obj", j, cost[j]);
    if (error) goto QUIT;
    error = GRBsetstrattrelement(model, "VarName", j, Foods[j]);
    if (error) goto QUIT;
  }

  /* Initialize decision variables for the nutrition information,
     which we limit via bounds */
  for (j = 0; j < nCategories; ++j)
  {
    error = GRBsetdblattrelement(model, "LB", j + nFoods,
minNutrition[j]);
    if (error) goto QUIT;
    error = GRBsetdblattrelement(model, "UB", j + nFoods,
maxNutrition[j]);
    if (error) goto QUIT;
    error = GRBsetstrattrelement(model, "VarName", j + nFoods,
Categories[j]);
    if (error) goto QUIT;
  }

  /* The objective is to minimize the costs */
  error = GRBsetintattr(model, "ModelSense", 1);
  if (error) goto QUIT;

  /* Nutrition constraints */
  cbeg = malloc(sizeof(int) * nCategories);
  if (!cbeg) goto QUIT;
  cind = malloc(sizeof(int) * nCategories * (nFoods + 1));
  if (!cind) goto QUIT;
  cval = malloc(sizeof(double) * nCategories * (nFoods + 1));
  if (!cval) goto QUIT;
  rhs = malloc(sizeof(double) * nCategories);
  if (!rhs) goto QUIT;
  sense = malloc(sizeof(char) * nCategories);
  if (!sense) goto QUIT;
  idx = 0;
  for (i = 0; i < nCategories; ++i)
  {
    cbeg[i] = idx;
    rhs[i] = 0.0;
    sense[i] = GRB_EQUAL;
    for (j = 0; j < nFoods; ++j)
    {
      cind[idx] = j;
      cval[idx++] = nutritionValues[j][i];
    }
    cind[idx] = nFoods + i;
    cval[idx++] = -1.0;
  }

  error = GRBaddconstrs(model, nCategories, idx, cbeg, cind, cval,
sense,
                        rhs, Categories);
  if (error) goto QUIT;

  /* Use barrier to solve model */
  error = GRBsetintparam(GRBgetenv(model),
                         GRB_INT_PAR_METHOD,
                         GRB_METHOD_BARRIER);
  if (error) goto QUIT;

  /* Solve */
  error = GRBoptimize(model);
  if (error) goto QUIT;
  error = printSolution(model, nCategories, nFoods);
  if (error) goto QUIT;

  printf("\nAdding constraint: at most 6 servings of dairy\n");
  cind[0] = 7;
  cval[0] = 1.0;
  cind[1] = 8;
  cval[1] = 1.0;
  error = GRBaddconstr(model, 2, cind, cval, GRB_LESS_EQUAL, 6.0,
                       "limit_dairy");
  if (error) goto QUIT;

  /* Solve */
  error = GRBoptimize(model);
  if (error) goto QUIT;
  error = printSolution(model, nCategories, nFoods);
  if (error) goto QUIT;



QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }
  /* Free data */

  free(cbeg);
  free(cind);
  free(cval);
  free(rhs);
  free(sense);

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}

int printSolution(GRBmodel* model, int nCategories, int nFoods)
{
  int error, status, i, j;
  double obj, x;
  char* vname;

  error = GRBgetintattr(model, "Status", &status);
  if (error) return error;
  if (status == GRB_OPTIMAL)
  {
    error = GRBgetdblattr(model, "ObjVal", &obj);
    if (error) return error;
    printf("\nCost: %f\n\nBuy:\n", obj);
    for (j = 0; j < nFoods; ++j)
    {
      error = GRBgetdblattrelement(model, "X", j, &x);
      if (error) return error;
      if (x > 0.0001)
      {
        error = GRBgetstrattrelement(model, "VarName", j, &vname);
        if (error) return error;
        printf("%s %f\n", vname, x);
      }
    }
    printf("\nNutrition:\n");
    for (i = 0; i < nCategories; ++i)
    {
      error = GRBgetdblattrelement(model, "X", i + nFoods, &x);
      if (error) return error;
      error = GRBgetstrattrelement(model, "VarName", i + nFoods,
&vname);
      if (error) return error;
      printf("%s %f\n", vname, x);
    }
  }
  else
  {
    printf("No solution\n");
  }

  return 0;
}
```

页面：facility_c.c

facility_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Facility location: a company currently ships its product from 5
plants
   to 4 warehouses. It is considering closing some plants to reduce
   costs. What plant(s) should the company close, in order to minimize
   transportation and fixed costs?

   Based on an example from Frontline Systems:
   http://www.solver.com/disfacility.htm
   Used with permission.
 */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "gurobi_c.h"


#define opencol(p)          p
#define transportcol(w,p)   nPlants*(w+1)+p
#define MAXSTR              128

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0;
  int       p, w, col;
  int      *cbeg = NULL;
  int      *cind = NULL;
  int       idx, rowct;
  double   *cval = NULL;
  double   *rhs = NULL;
  char     *sense = NULL;
  char     vname[MAXSTR];
  int      cnamect = 0;
  char   **cname = NULL;
  double   maxFixed = -GRB_INFINITY, sol, obj;

  /* Number of plants and warehouses */
  const int nPlants = 5;
  const int nWarehouses = 4;

  /* Warehouse demand in thousands of units */
  double Demand[] = { 15, 18, 14, 20 };

  /* Plant capacity in thousands of units */
  double Capacity[] = { 20, 22, 17, 19, 18 };
  /* Fixed costs for each plant */
  double FixedCosts[] =
    { 12000, 15000, 17000, 13000, 16000 };

  // Transportation costs per thousand units
  double TransCosts[4][5] = {
                              { 4000, 2000, 3000, 2500, 4500 },
                              { 2500, 2600, 3400, 3000, 4000 },
                              { 1200, 1800, 2600, 4100, 3000 },
                              { 2200, 2600, 3100, 3700, 3200 }
                            };

  /* Create environment */
  error = GRBloadenv(&env, "facility.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create initial model */
  error = GRBnewmodel(env, &model, "facility", nPlants * (nWarehouses +
1),
                      NULL, NULL, NULL, NULL, NULL);
  if (error) goto QUIT;

  /* Initialize decision variables for plant open variables */
  for (p = 0; p < nPlants; ++p)
  {
    col = opencol(p);
    error = GRBsetcharattrelement(model, "VType", col, GRB_BINARY);
    if (error) goto QUIT;
    error = GRBsetdblattrelement(model, "Obj", col, FixedCosts[p]);
    if (error) goto QUIT;
    sprintf(vname, "Open%i", p);
    error = GRBsetstrattrelement(model, "VarName", col, vname);
    if (error) goto QUIT;
  }

  /* Initialize decision variables for transportation decision
variables:
     how much to transport from a plant p to a warehouse w */
  for (w = 0; w < nWarehouses; ++w)
  {
    for (p = 0; p < nPlants; ++p)
    {
      col = transportcol(w, p);
      error = GRBsetdblattrelement(model, "Obj", col, TransCosts[w][p]);
      if (error) goto QUIT;
      sprintf(vname, "Trans%i.%i", p, w);
      error = GRBsetstrattrelement(model, "VarName", col, vname);
      if (error) goto QUIT;
    }
  }

  /* The objective is to minimize the total fixed and variable costs */
  error = GRBsetintattr(model, "ModelSense", 1);
  if (error) goto QUIT;

  /* Make space for constraint data */
  rowct = (nPlants > nWarehouses) ? nPlants : nWarehouses;
  cbeg = malloc(sizeof(int) * rowct);
  if (!cbeg) goto QUIT;
  cind = malloc(sizeof(int) * (nPlants * (nWarehouses + 1)));
  if (!cind) goto QUIT;
  cval = malloc(sizeof(double) * (nPlants * (nWarehouses + 1)));
  if (!cval) goto QUIT;
  rhs = malloc(sizeof(double) * rowct);
  if (!rhs) goto QUIT;
  sense = malloc(sizeof(char) * rowct);
  if (!sense) goto QUIT;
  cname = calloc(rowct, sizeof(char*));
  if (!cname) goto QUIT;

  /* Production constraints
     Note that the limit sets the production to zero if
     the plant is closed */
  idx = 0;
  for (p = 0; p < nPlants; ++p)
  {
    cbeg[p] = idx;
    rhs[p] = 0.0;
    sense[p] = GRB_LESS_EQUAL;
    cname[p] = malloc(sizeof(char) * MAXSTR);
    if (!cname[p]) goto QUIT;
    cnamect++;
    sprintf(cname[p], "Capacity%i", p);
    for (w = 0; w < nWarehouses; ++w)
    {
      cind[idx] = transportcol(w, p);
      cval[idx++] = 1.0;
    }
    cind[idx] = opencol(p);
    cval[idx++] = -Capacity[p];
  }
  error = GRBaddconstrs(model, nPlants, idx, cbeg, cind, cval, sense,
                        rhs, cname);
  if (error) goto QUIT;

  /* Demand constraints */
  idx = 0;
  for (w = 0; w < nWarehouses; ++w)
  {
    cbeg[w] = idx;
    sense[w] = GRB_EQUAL;
    sprintf(cname[w], "Demand%i", w);
    for (p = 0; p < nPlants; ++p)
    {
      cind[idx] = transportcol(w, p);
      cval[idx++] = 1.0;
    }
  }
  error = GRBaddconstrs(model, nWarehouses, idx, cbeg, cind, cval,
sense,
                        Demand, cname);
  if (error) goto QUIT;

  /* Guess at the starting point: close the plant with the highest
     fixed costs; open all others */

  /* First, open all plants */
  for (p = 0; p < nPlants; ++p)
  {
    error = GRBsetdblattrelement(model, "Start", opencol(p), 1.0);
    if (error) goto QUIT;
  }

  /* Now close the plant with the highest fixed cost */
  printf("Initial guess:\n");
  for (p = 0; p < nPlants; ++p)
  {
    if (FixedCosts[p] > maxFixed)
    {
      maxFixed = FixedCosts[p];
    }
  }
  for (p = 0; p < nPlants; ++p)
  {
    if (FixedCosts[p] == maxFixed)
    {
      error = GRBsetdblattrelement(model, "Start", opencol(p), 0.0);
      if (error) goto QUIT;
      printf("Closing plant %i\n\n", p);
      break;
    }
  }

  /* Use barrier to solve root relaxation */
  error = GRBsetintparam(GRBgetenv(model),
                         GRB_INT_PAR_METHOD,
                         GRB_METHOD_BARRIER);
  if (error) goto QUIT;

  /* Solve */
  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Print solution */
  error = GRBgetdblattr(model, "ObjVal", &obj);
  if (error) goto QUIT;
  printf("\nTOTAL COSTS: %f\n", obj);
  printf("SOLUTION:\n");
  for (p = 0; p < nPlants; ++p)
  {
    error = GRBgetdblattrelement(model, "X", opencol(p), &sol);
    if (error) goto QUIT;
    if (sol == 1.0)
    {
      printf("Plant %i open:\n", p);
      for (w = 0; w < nWarehouses; ++w)
      {
        error = GRBgetdblattrelement(model, "X", transportcol(w, p),
&sol);
        if (error) goto QUIT;
        if (sol > 0.0001)
        {
          printf("  Transport %f units to warehouse %i\n", sol, w);
        }
      }
    }
    else
    {
      printf("Plant %i closed!\n", p);
    }
  }


QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free data */

  free(cbeg);
  free(cind);
  free(cval);
  free(rhs);
  free(sense);
  for (p = 0; p < cnamect; ++p) {
    free(cname[p]);
  }
  free(cname);

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：feasopt_c.c

feasopt_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads a MIP model from a file, adds artificial
   variables to each constraint, and then minimizes the sum of the
   artificial variables.  A solution with objective zero corresponds
   to a feasible solution to the input model. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0;
  int       i, j;
  int       numvars, numconstrs;
  char      sense;
  int       vind[1];
  double    vval[1];
  char      *cname, *vname;

  if (argc < 2)
  {
    fprintf(stderr, "Usage: feasopt_c filename\n");
    exit(1);
  }

  error = GRBloadenv(&env, "feasopt.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  error = GRBreadmodel(env, argv[1], &model);
  if (error) goto QUIT;

  /* clear objective */
  error = GRBgetintattr(model, "NumVars", &numvars);
  if (error) goto QUIT;
  for (j = 0; j < numvars; ++j)
  {
    error = GRBsetdblattrelement(model, "Obj", j, 0.0);
    if (error) goto QUIT;
  }
  /* add slack variables */
  error = GRBgetintattr(model, "NumConstrs", &numconstrs);
  if (error) goto QUIT;
  for (i = 0; i < numconstrs; ++i)
  {
    error = GRBgetcharattrelement(model, "Sense", i, &sense);
    if (error) goto QUIT;
    if (sense != '>')
    {
      error = GRBgetstrattrelement(model, "ConstrName", i, &cname);
      if (error) goto QUIT;
      vname = malloc(sizeof(char) * (6 + strlen(cname)));
      if (!vname) goto QUIT;
      strcpy(vname, "ArtN_");
      strcat(vname, cname);
      vind[0] = i;
      vval[0] = -1.0;
      error = GRBaddvar(model, 1, vind, vval, 1.0, 0.0, GRB_INFINITY,
                        GRB_CONTINUOUS, vname);
      if (error) goto QUIT;
      free(vname);
    }
    if (sense != '<')
    {
      error = GRBgetstrattrelement(model, "ConstrName", i, &cname);
      if (error) goto QUIT;
      vname = malloc(sizeof(char) * (6 + strlen(cname)));
      if (!vname) goto QUIT;
      strcpy(vname, "ArtP_");
      strcat(vname, cname);
      vind[0] = i;
      vval[0] = 1.0;
      error = GRBaddvar(model, 1, vind, vval, 1.0, 0.0, GRB_INFINITY,
                        GRB_CONTINUOUS, vname);
      if (error) goto QUIT;
      free(vname);
    }
  }
  error = GRBupdatemodel(model);
  if (error) goto QUIT;

  /* optimize modified model */
  error = GRBwrite(model, "feasopt.lp");
  if (error) goto QUIT;

  error = GRBoptimize(model);
  if (error) goto QUIT;


QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：fixanddive_c.c

fixanddive_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Implement a simple MIP heuristic.  Relax the model,
   sort variables based on fractionality, and fix the 25% of
   the fractional variables that are closest to integer variables.
   Repeat until either the relaxation is integer feasible or
   linearly infeasible. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "gurobi_c.h"

typedef struct
{
  int index;
  double X;
}
var_t ;

int vcomp(const void* v1, const void* v2);


int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL, *modelenv = NULL;
  GRBmodel *model = NULL;
  int       error = 0;
  int       j, numfractional, iter, nfix;
  int       numintvars;
  int      *intvars = NULL;
  int       status;
  char      vtype, *vname;
  double    sol, obj, fixval;
  var_t    *fractional = NULL;

  if (argc < 2)
  {
    fprintf(stderr, "Usage: fixanddive_c filename\n");
    exit(1);
  }

  error = GRBloadenv(&env, "fixanddive.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Read model */
```

```
error = GRBreadmodel(env, argv[1], &model);
if (error) goto QUIT;

/* Collect integer variables and relax them */
error = GRBgetintattr(model, "NumIntVars", &numintvars);
if (error) goto QUIT;
intvars = malloc(sizeof(int) * numintvars);
if (!intvars) goto QUIT;
fractional = malloc(sizeof(var_t) * numintvars);
if (!fractional) goto QUIT;
numfractional = 0;
for (j = 0; j < numintvars; ++j)
{
  error = GRBgetcharattrelement(model, "VType", j, &vtype);
  if (error) goto QUIT;
  if (vtype != GRB_CONTINUOUS)
  {
    intvars[numfractional++] = j;
    error = GRBsetcharattrelement(model, "VType", j, GRB_CONTINUOUS);
    if (error) goto QUIT;
  }
}

modelenv = GRBgetenv(model);
if (!modelenv) goto QUIT;
error = GRBsetintparam(modelenv, "OutputFlag", 0);
if (error) goto QUIT;
error = GRBoptimize(model);
if (error) goto QUIT;

/* Perform multiple iterations. In each iteration, identify the first
   quartile of integer variables that are closest to an integer value
   in the relaxation, fix them to the nearest integer, and repeat. */

for (iter = 0; iter < 1000; ++iter)
{

  /* create a list of fractional variables, sorted in order of
     increasing distance from the relaxation solution to the nearest
     integer value */

  numfractional = 0;
  for (j = 0; j < numintvars; ++j)
  {
    error = GRBgetdblattrelement(model, "X", intvars[j], &sol);
    if (error) goto QUIT;
    if (fabs(sol - floor(sol + 0.5)) > 1e-5)
    {
      fractional[numfractional].index = intvars[j];
      fractional[numfractional++].X = sol;
    }
  }

  error = GRBgetdblattr(model, "ObjVal", &obj);
  if (error) goto QUIT;
  printf("Iteration %i, obj %f, fractional %i\n",
         iter, obj, numfractional);
    if (numfractional == 0)
    {
      printf("Found feasible solution - objective %f\n", obj);
      break;
    }

    /* Fix the first quartile to the nearest integer value */
    qsort(fractional, numfractional, sizeof(var_t), vcomp);
    nfix = numfractional / 4;
    nfix = (nfix > 1) ? nfix : 1;
    for (j = 0; j < nfix; ++j)
    {
      fixval = floor(fractional[j].X + 0.5);
      error = GRBsetdblattrelement(model, "LB", fractional[j].index,
fixval);
      if (error) goto QUIT;
      error = GRBsetdblattrelement(model, "UB", fractional[j].index,
fixval);
      if (error) goto QUIT;
      error = GRBgetstrattrelement(model, "VarName",
                                   fractional[j].index, &vname);
      printf("  Fix %s to %f ( rel %f )\n", vname, fixval,
fractional[j].X);
    }

    error = GRBoptimize(model);
    if (error) goto QUIT;

    /* Check optimization result */

    error = GRBgetintattr(model, "Status", &status);
    if (error) goto QUIT;
    if (status != GRB_OPTIMAL)
    {
      printf("Relaxation is infeasible\n");
      break;
    }
  }


QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free data */

  free(intvars);
  free(fractional);

  /* Free model */
  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}


int vcomp(const void* v1, const void* v2)
{
  double sol1, sol2, frac1, frac2;
  sol1 = fabs(((var_t *)v1)->X);
  sol2 = fabs(((var_t *)v2)->X);
  frac1 = fabs(sol1 - floor(sol1 + 0.5));
  frac2 = fabs(sol2 - floor(sol2 + 0.5));
  return (frac1 < frac2) ? -1 : ((frac1 == frac2) ? 0 : 1);
}
```

页面：lp_c.c

lp_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP model from a file and solves it.
   If the model is infeasible or unbounded, the example turns off
   presolve and solves the model again. If the model is infeasible,
   the example computes an Irreducible Infeasible Subsystem (IIS),
   and writes it to a file */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *masterenv = NULL;
  GRBmodel *model     = NULL;
  GRBenv   *modelenv  = NULL;
  int       error     = 0;
  int       optimstatus;
  double    objval;

  if (argc < 2) {
    fprintf(stderr, "Usage: lp_c filename\n");
    exit(1);
  }

  /* Create environment */

  error = GRBloadenv(&masterenv, "lp.log");
  if (error || masterenv == NULL) {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Read model from file */

  error = GRBreadmodel(masterenv, argv[1], &model);
  if (error) goto QUIT;

  /* Solve model */

  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Capture solution information */

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;
  /* If model is infeasible or unbounded, turn off presolve and resolve
*/

  if (optimstatus == GRB_INF_OR_UNBD) {
    modelenv = GRBgetenv(model);
    if (!modelenv) {
      fprintf(stderr, "Error: could not get model environment\n");
      goto QUIT;
    }

    /* Change parameter on model environment.  The model now has
       a copy of the master environment, so changing the master will
       no longer affect the model.  */

    error = GRBsetintparam(modelenv, "PRESOLVE", 0);
    if (error) goto QUIT;

    error = GRBoptimize(model);
    if (error) goto QUIT;

    error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
    if (error) goto QUIT;
  }

  if (optimstatus == GRB_OPTIMAL) {
    error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
    if (error) goto QUIT;
    printf("Optimal objective: %.4e\n\n", objval);
  } else if (optimstatus == GRB_INFEASIBLE) {
    printf("Model is infeasible\n\n");

    error = GRBcomputeIIS(model);
    if (error) goto QUIT;

    error = GRBwrite(model, "model.ilp");
    if (error) goto QUIT;
  } else if (optimstatus == GRB_UNBOUNDED) {
    printf("Model is unbounded\n\n");
  } else {
    printf("Optimization was stopped with status = %d\n\n",
optimstatus);
  }

QUIT:

  /* Error reporting */

  if (error) {
    printf("ERROR: %s\n", GRBgeterrormsg(masterenv));
    exit(1);
  }

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */
  GRBfreeenv(masterenv);

  return 0;
}
```

页面：lpmethod_c.c

lpmethod_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Solve a model with different values of the Method parameter;
   show which value gives the shortest solve time. */

#include <stdlib.h>
#include <stdio.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env = NULL, *menv;
  GRBmodel *m = NULL;
  int       error = 0;
  int       i;
  int       optimstatus;
  int       bestMethod = -1;
  double    bestTime;

  if (argc < 2)
  {
    fprintf(stderr, "Usage: lpmethod_c filename\n");
    exit(1);
  }

  error = GRBloadenv(&env, "lpmethod.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Read model */
  error = GRBreadmodel(env, argv[1], &m);
  if (error) goto QUIT;
  menv = GRBgetenv(m);
  error = GRBgetdblparam(menv, "TimeLimit", &bestTime);
  if (error) goto QUIT;

  /* Solve the model with different values of Method */
  for (i = 0; i <= 2; ++i)
  {
    error = GRBresetmodel(m);
    if (error) goto QUIT;
    error = GRBsetintparam(menv, "Method", i);
    if (error) goto QUIT;
    error = GRBoptimize(m);
    if (error) goto QUIT;
    error = GRBgetintattr(m, "Status", &optimstatus);
    if (error) goto QUIT;
```

```
      if (optimstatus == GRB_OPTIMAL) {
        error = GRBgetdblattr(m, "Runtime", &bestTime);
        if (error) goto QUIT;
        bestMethod = i;
        /* Reduce the TimeLimit parameter to save time
           with other methods */
        error = GRBsetdblparam(menv, "TimeLimit", bestTime);
        if (error) goto QUIT;
      }
    }

    /* Report which method was fastest */
    if (bestMethod == -1) {
      printf("Unable to solve this model\n");
    } else {
      printf("Solved in %f seconds with Method: %i\n",
             bestTime, bestMethod);
    }

QUIT:

    /* Error reporting */

    if (error)
    {
      printf("ERROR: %s\n", GRBgeterrormsg(env));
      exit(1);
    }

    /* Free model */

    GRBfreemodel(m);

    /* Free environment */

    GRBfreeenv(env);

    return 0;
}
```

页面：lpmod_c.c

lpmod_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP model from a file and solves it.
   If the model can be solved, then it finds the smallest positive
variable,
   sets its upper bound to zero, and resolves the model two ways:
   first with an advanced start, then without an advanced start
   (i.e. 'from scratch'). */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0;
  int       j;
  int       numvars, isMIP, status, minVar = 0;
  double    minVal = GRB_INFINITY, sol, lb;
  char     *varname;
  double    warmCount, warmTime, coldCount, coldTime;

  if (argc < 2)
  {
    fprintf(stderr, "Usage: lpmod_c filename\n");
    exit(1);
  }

  error = GRBloadenv(&env, "lpmod.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Read model and determine whether it is an LP */
  error = GRBreadmodel(env, argv[1], &model);
  if (error) goto QUIT;
  error = GRBgetintattr(model, "IsMIP", &isMIP);
  if (error) goto QUIT;
  if (isMIP)
  {
    printf("The model is not a linear program\n");
    goto QUIT;
  }
```

```
error = GRBoptimize(model);
if (error) goto QUIT;

error = GRBgetintattr(model, "Status", &status);
if (error) goto QUIT;

if ((status == GRB_INF_OR_UNBD) || (status == GRB_INFEASIBLE) ||
    (status == GRB_UNBOUNDED))
{
  printf("The model cannot be solved because it is ");
  printf("infeasible or unbounded\n");
  goto QUIT;
}

if (status != GRB_OPTIMAL)
{
  printf("Optimization was stopped with status %i\n", status);
  goto QUIT;
}

/* Find the smallest variable value */
error = GRBgetintattr(model, "NumVars", &numvars);
if (error) goto QUIT;
for (j = 0; j < numvars; ++j)
{
  error = GRBgetdblattrelement(model, "X", j, &sol);
  if (error) goto QUIT;
  error = GRBgetdblattrelement(model, "LB", j, &lb);
  if (error) goto QUIT;
  if ((sol > 0.0001) && (sol < minVal) &&
      (lb == 0.0))
  {
    minVal = sol;
    minVar = j;
  }
}

error = GRBgetstrattrelement(model, "VarName", minVar, &varname);
if (error) goto QUIT;
printf("\n*** Setting %s from %f to zero ***\n\n", varname, minVal);
error = GRBsetdblattrelement(model, "LB", minVar, 0.0);
if (error) goto QUIT;

/* Solve from this starting point */
error = GRBoptimize(model);
if (error) goto QUIT;

/* Save iteration & time info */
error = GRBgetdblattr(model, "IterCount", &warmCount);
if (error) goto QUIT;
error = GRBgetdblattr(model, "Runtime", &warmTime);
if (error) goto QUIT;

/* Reset the model and resolve */
printf("\n*** Resetting and solving ");
printf("without an advanced start ***\n\n");
error = GRBresetmodel(model);
```

```
  if (error) goto QUIT;
  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Save iteration & time info */
  error = GRBgetdblattr(model, "IterCount", &coldCount);
  if (error) goto QUIT;
  error = GRBgetdblattr(model, "Runtime", &coldTime);
  if (error) goto QUIT;

  printf("\n*** Warm start: %f iterations, %f seconds\n",
         warmCount, warmTime);
  printf("*** Cold start: %f iterations, %f seconds\n",
         coldCount, coldTime);


QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：mip1_c.c

mip1_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example formulates and solves the following simple MIP model:

     maximize    x +   y + 2 z
     subject to  x + 2 y + 3 z <= 4
                 x +   y       >= 1
     x, y, z binary
*/

#include <stdlib.h>
#include <stdio.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0;
  double    sol[3];
  int       ind[3];
  double    val[3];
  double    obj[3];
  char      vtype[3];
  int       optimstatus;
  double    objval;
  int       zero = 0;

  /* Create environment */

  error = GRBloadenv(&env, "mip1.log");
  if (error || env == NULL) {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create an empty model */

  error = GRBnewmodel(env, &model, "mip1", 0, NULL, NULL, NULL, NULL,
NULL);
  if (error) goto QUIT;


  /* Add variables */

  obj[0] = -1; obj[1] = -1; obj[2] = -2;
  vtype[0] = GRB_BINARY; vtype[1] = GRB_BINARY; vtype[2] = GRB_BINARY;
  error = GRBaddvars(model, 3, 0, NULL, NULL, NULL, obj, NULL, NULL,
vtype,
                     NULL);
  if (error) goto QUIT;

  /* Integrate new variables */

  error = GRBupdatemodel(model);
  if (error) goto QUIT;


  /* First constraint: x + 2 y + 3 z <= 4 */

  ind[0] = 0; ind[1] = 1; ind[2] = 2;
  val[0] = 1; val[1] = 2; val[2] = 3;

  error = GRBaddconstr(model, 3, ind, val, GRB_LESS_EQUAL, 4.0, "c0");
  if (error) goto QUIT;

  /* Second constraint: x + y >= 1 */

  ind[0] = 0; ind[1] = 1;
  val[0] = 1; val[1] = 1;

  error = GRBaddconstr(model, 2, ind, val, GRB_GREATER_EQUAL, 1.0,
"c1");
  if (error) goto QUIT;

  /* Optimize model */

  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Write model to 'mip1.lp' */

  error = GRBwrite(model, "mip1.lp");
  if (error) goto QUIT;

  /* Capture solution information */

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;

  error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
  if (error) goto QUIT;

  error = GRBgetdblattrarray(model, GRB_DBL_ATTR_X, 0, 3, sol);
  if (error) goto QUIT;

  printf("\nOptimization complete\n");
  if (optimstatus == GRB_OPTIMAL) {
    printf("Optimal objective: %.4e\n", objval);

    printf("  x=%.0f, y=%.0f, z=%.0f\n", sol[0], sol[1], sol[2]);
  } else if (optimstatus == GRB_INF_OR_UNBD) {
    printf("Model is infeasible or unbounded\n");
  } else {
    printf("Optimization was stopped early\n");
  }
```

```
QUIT:

  /* Error reporting */

  if (error) {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：mip2_c.c

mip2_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads a MIP model from a file, solves it and
   prints the objective values from all feasible solutions
   generated while solving the MIP. Then it creates the fixed
   model and solves that model. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  GRBmodel *fixed = NULL;
  int       error = 0;
  int       ismip;
  int       j, k, solcount, numvars;
  double    objn, vobj, xn;
  int       optimstatus, foptimstatus;
  double    objval, fobjval;
  char      *varname;
  double    x;

  /* To change settings for a loaded model, we need to get
     the model environment, which will be freed when the model
     is freed. */

  GRBenv   *menv, *fenv;

  if (argc < 2) {
    fprintf(stderr, "Usage: mip2_c filename\n");
    exit(1);
  }

  /* Create environment */

  error = GRBloadenv(&env, "mip2.log");
  if (error || env == NULL) {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Read model from file */

  error = GRBreadmodel(env, argv[1], &model);
  if (error) goto QUIT;
  error = GRBgetintattr(model, "IsMIP", &ismip);
  if (error) goto QUIT;

  if (ismip == 0) {
    printf("Model is not a MIP\n");
    goto QUIT;
  }

  /* Get model environment */

  menv = GRBgetenv(model);
  if (!menv) {
    fprintf(stderr, "Error: could not get model environment\n");
    goto QUIT;
  }

  /* Solve model */

  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Capture solution information */

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;

  printf("\nOptimization complete\n");
  if (optimstatus == GRB_OPTIMAL) {
    error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
    if (error) goto QUIT;
    printf("Optimal objective: %.4e\n\n", objval);
  } else if (optimstatus == GRB_INF_OR_UNBD) {
    printf("Model is infeasible or unbounded\n\n");
    goto QUIT;
  } else if (optimstatus == GRB_INFEASIBLE) {
    printf("Model is infeasible\n\n");
    goto QUIT;
  } else if (optimstatus == GRB_UNBOUNDED) {
    printf("Model is unbounded\n\n");
    goto QUIT;
  } else {
    printf("Optimization was stopped with status = %d\n\n",
optimstatus);
    goto QUIT;
  }

  /* Iterate over the solutions and compute the objectives */

  error = GRBsetintparam(menv, "OutputFlag", 0);
  if (error) goto QUIT;
  error = GRBgetintattr(model, "SolCount", &solcount);
  if (error) goto QUIT;
  error = GRBgetintattr(model, "NumVars", &numvars);
  if (error) goto QUIT;

  printf("\n");
  for ( k = 0; k < solcount; ++k ) {
    error = GRBsetintparam(menv, "SolutionNumber", k);
    objn = 0.0;
    for ( j = 0; j < numvars; ++j ) {
      error = GRBgetdblattrelement(model, "Obj", j, &vobj);
      if (error) goto QUIT;
      error = GRBgetdblattrelement(model, "Xn", j, &xn);
      if (error) goto QUIT;
      objn += vobj * xn;
    }
    printf("Solution %i has objective: %f\n", k, objn);
  }
  printf("\n");

  error = GRBsetintparam(menv, "OutputFlag", 1);
  if (error) goto QUIT;

  /* Create a fixed model, turn off presolve and solve */

  fixed = GRBfixedmodel(model);
  if (!fixed) {
    fprintf(stderr, "Error: could not create fixed model\n");
    goto QUIT;
  }

  fenv = GRBgetenv(fixed);
  if (!fenv) {
    fprintf(stderr, "Error: could not get fixed model environment\n");
    goto QUIT;
  }

  error = GRBsetintparam(fenv, "PRESOLVE", 0);
  if (error) goto QUIT;

  error = GRBoptimize(fixed);
  if (error) goto QUIT;

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &foptimstatus);
  if (error) goto QUIT;

  if (foptimstatus != GRB_OPTIMAL) {
    fprintf(stderr, "Error: fixed model isn't optimal\n");
    goto QUIT;
  }

  error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &fobjval);
  if (error) goto QUIT;

  if (fabs(fobjval - objval) > 1.0e-6 * (1.0 + fabs(objval))) {
    fprintf(stderr, "Error: objective values are different\n");
  }

  /* Print values of nonzero variables */
  for ( j = 0; j < numvars; ++j ) {
    error = GRBgetstrattrelement(fixed, "VarName", j, &varname);
    if (error) goto QUIT;
    error = GRBgetdblattrelement(fixed, "X", j, &x);
    if (error) goto QUIT;
    if (x != 0.0) {
      printf("%s %f\n", varname, x);
    }
  }


QUIT:

  /* Error reporting */

  if (error) {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free models */

  GRBfreemodel(model);
  GRBfreemodel(fixed);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：params_c.c

params_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Use parameters that are associated with a model.

   A MIP is solved for 5 seconds with different sets of parameters.
   The one with the smallest MIP gap is selected, and the optimization
   is resumed until the optimal solution is found.
*/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "gurobi_c.h"

int gap(GRBmodel*, double*);

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL, *baseenv = NULL, *menv = NULL, *bestenv =
NULL;
  GRBmodel *base  = NULL, *m = NULL, *bestModel = NULL;
  int       error = 0;
  int       i;
  int       ismip;
  int       mipfocus;
  double    bestGap = GRB_INFINITY, currGap;

  if (argc < 2)
  {
    fprintf(stderr, "Usage: params_c filename\n");
    exit(1);
  }

  error = GRBloadenv(&env, "params.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Read model and verify that it is a MIP */
  error = GRBreadmodel(env, argv[1], &base);
  if (error) goto QUIT;
  error = GRBgetintattr(base, "IsMIP", &ismip);
  if (error) goto QUIT;
  if (ismip == 0)
  {
    printf("The model is not an integer program\n");
    goto QUIT;
  }
  /* Set a 5 second time limit */
  baseenv = GRBgetenv(base);
  if (!baseenv) goto QUIT;
  error = GRBsetdblparam(baseenv, "TimeLimit", 5);
  if (error) goto QUIT;

  /* Now solve the model with different values of MIPFocus */
  for (i = 0; i <= 3; ++i)
  {
    m = GRBcopymodel(base);
    if (!m) goto QUIT;
    menv = GRBgetenv(m);
    if (!menv) goto QUIT;
    error = GRBsetintparam(menv, "MIPFocus", i);
    if (error) goto QUIT;
    error = GRBoptimize(m);
    if (error) goto QUIT;
    error = gap(m, &currGap);
    if (error) goto QUIT;
    if (bestModel == NULL || bestGap > currGap)
    {
      GRBfreemodel(bestModel);
      bestModel = m;
      bestGap = currGap;
    }
    else
    {
      GRBfreemodel(m);
    }
  }

  /* Finally, reset the time limit and continue to solve the
     best model to optimality */
  bestenv = GRBgetenv(bestModel);
  if (!bestenv) goto QUIT;
  error = GRBsetdblparam(bestenv, "TimeLimit", GRB_INFINITY);
  if (error) goto QUIT;
  error = GRBoptimize(bestModel);
  if (error) goto QUIT;
  error = GRBgetintparam(bestenv, "MIPFocus", &mipfocus);
  if (error) goto QUIT;
  printf("Solved with MIPFocus: %i", mipfocus);


QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free models */
  GRBfreemodel(bestModel);
  GRBfreemodel(base);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}


/* Simple function to determine the MIP gap */
int gap(GRBmodel *model, double *gap)
{
  int error;
  int solcount;
  double objval, objbound;

  error = GRBgetintattr(model, "SolCount", &solcount);
  if (error) return error;
  error = GRBgetdblattr(model, "ObjVal", &objval);
  if (error) return error;

  if ((solcount == 0) || (fabs(objval) < 1e-6))
  {
    *gap = GRB_INFINITY;
    return 0;
  }
  error = GRBgetdblattr(model, "ObjBound", &objbound);
  if (error) return error;

  *gap = fabs(objbound - objval) / fabs(objval);
  return 0;
}
```

页面：qp1 c.c

qp1_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example formulates and solves the following simple QP model:

     minimize    x^2 + x*y + y^2 + y*z + z^2
     subject to  x + 2 y + 3 z >= 4
                 x +   y       >= 1

   It solves it once as a continuous model, and once as an integer
model.
*/

#include <stdlib.h>
#include <stdio.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0;
  double    sol[3];
  int       ind[3];
  double    val[3];
  int       qrow[5];
  int       qcol[5];
  double    qval[5];
  char      vtype[3];
  int       optimstatus;
  double    objval;

  /* Create environment */

  error = GRBloadenv(&env, "qp1.log");
  if (error || env == NULL) {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create an empty model */

  error = GRBnewmodel(env, &model, "qp1", 0, NULL, NULL, NULL, NULL,
NULL);
  if (error) goto QUIT;


  /* Add variables */

  error = GRBaddvars(model, 3, 0, NULL, NULL, NULL, NULL, NULL, NULL,
NULL,
                     NULL);
  if (error) goto QUIT;

  /* Integrate new variables */

  error = GRBupdatemodel(model);
  if (error) goto QUIT;

  /* Quadratic objective */
  qrow[0] = 0; qrow[1] = 0; qrow[2] = 1; qrow[3] = 1; qrow[4] = 2;
  qcol[0] = 0; qcol[1] = 1; qcol[2] = 1; qcol[3] = 2; qcol[4] = 2;
  qval[0] = 1; qval[1] = 1; qval[2] = 1; qval[3] = 1; qval[4] = 1;

  error = GRBaddqpterms(model, 5, qrow, qcol, qval);
  if (error) goto QUIT;

  /* First constraint: x + 2 y + 3 z <= 4 */

  ind[0] = 0; ind[1] = 1; ind[2] = 2;
  val[0] = 1; val[1] = 2; val[2] = 3;

  error = GRBaddconstr(model, 3, ind, val, GRB_GREATER_EQUAL, 4.0,
"c0");
  if (error) goto QUIT;

  /* Second constraint: x + y >= 1 */

  ind[0] = 0; ind[1] = 1;
  val[0] = 1; val[1] = 1;

  error = GRBaddconstr(model, 2, ind, val, GRB_GREATER_EQUAL, 1.0,
"c1");
  if (error) goto QUIT;

  /* Optimize model */

  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Write model to 'qp1.lp' */

  error = GRBwrite(model, "qp1.lp");
  if (error) goto QUIT;

  /* Capture solution information */

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;

  error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
  if (error) goto QUIT;

  error = GRBgetdblattrarray(model, GRB_DBL_ATTR_X, 0, 3, sol);
  if (error) goto QUIT;

  printf("\nOptimization complete\n");
  if (optimstatus == GRB_OPTIMAL) {
    printf("Optimal objective: %.4e\n", objval);

    printf("  x=%.0f, y=%.0f, z=%.0f\n", sol[0], sol[1], sol[2]);
  } else if (optimstatus == GRB_INF_OR_UNBD) {
    printf("Model is infeasible or unbounded\n");
  } else {
    printf("Optimization was stopped early\n");
  }


  /* Modify variable types */

  vtype[0] = GRB_INTEGER; vtype[1] = GRB_INTEGER; vtype[2] =
GRB_INTEGER;

  error = GRBsetcharattrarray(model, GRB_CHAR_ATTR_VTYPE, 0, 3, vtype);
  if (error) goto QUIT;

  /* Optimize model */

  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Write model to 'qp2.lp' */

  error = GRBwrite(model, "qp2.lp");
  if (error) goto QUIT;

  /* Capture solution information */

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;

  error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
  if (error) goto QUIT;

  error = GRBgetdblattrarray(model, GRB_DBL_ATTR_X, 0, 3, sol);
  if (error) goto QUIT;

  printf("\nOptimization complete\n");
  if (optimstatus == GRB_OPTIMAL) {
    printf("Optimal objective: %.4e\n", objval);

    printf("  x=%.4f, y=%.4f, z=%.4f\n", sol[0], sol[1], sol[2]);
  } else if (optimstatus == GRB_INF_OR_UNBD) {
    printf("Model is infeasible or unbounded\n");
  } else {
    printf("Optimization was stopped early\n");
  }

QUIT:

  /* Error reporting */

  if (error) {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：sensitivity_c.c

sensitivity_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Simple MIP sensitivity analysis example.
   For each integer variable, fix it to its lower and upper bound
   and check the impact on the objective. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env = NULL, *aenv;
  GRBmodel *a = NULL, *b = NULL;
  int       error = 0;
  int       i, numvars, status;
  char      vtype, *vname;
  double    x, bnd, aobj, bobj, objchg;

  if (argc < 2)
  {
    fprintf(stderr, "Usage: sensitivity_c filename\n");
    exit(1);
  }

  error = GRBloadenv(&env, "sensitivity.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Read model */
  error = GRBreadmodel(env, argv[1], &a);
  if (error) goto QUIT;
  error = GRBoptimize(a);
  if (error) goto QUIT;
  error = GRBgetdblattr(a, "ObjVal", &aobj);
  if (error) goto QUIT;
  aenv = GRBgetenv(a);
  if (!aenv) goto QUIT;
  error = GRBsetintparam(aenv, "OutputFlag", 0);
  if (error) goto QUIT;

  /* Iterate over all variables */
  error = GRBgetintattr(a, "NumVars", &numvars);
  if (error) goto QUIT;
  for (i = 0; i < numvars; ++i)
  {
    error = GRBgetcharattrelement(a, "VType", i, &vtype);
    if (error) goto QUIT;

    if (vtype == GRB_BINARY)
    {

      /* Create clone and fix variable */
      b = GRBcopymodel(a);
      if (!b) goto QUIT;
      error = GRBgetstrattrelement(a, "VarName", i, &vname);
      if (error) goto QUIT;
      error = GRBgetdblattrelement(a, "X", i, &x);
      if (error) goto QUIT;
      error = GRBgetdblattrelement(a, "LB", i, &bnd);
      if (error) goto QUIT;
      if (x - bnd < 0.5)
      {
        error = GRBgetdblattrelement(b, "UB", i, &bnd);
        if (error) goto QUIT;
        error = GRBsetdblattrelement(b, "LB", i, bnd);
        if (error) goto QUIT;
      }
      else
      {
        error = GRBgetdblattrelement(b, "LB", i, &bnd);
        if (error) goto QUIT;
        error = GRBsetdblattrelement(b, "UB", i, bnd);
        if (error) goto QUIT;
      }

      error = GRBoptimize(b);
      if (error) goto QUIT;

      error = GRBgetintattr(b, "Status", &status);
      if (error) goto QUIT;
      if (status == GRB_OPTIMAL)
      {
        error = GRBgetdblattr(b, "ObjVal", &bobj);
        if (error) goto QUIT;
        objchg = bobj - aobj;
        if (objchg < 0)
        {
          objchg = 0;
        }
        printf("Objective sensitivity for variable %s is %f\n", vname,
objchg);
      }
      else
      {
        printf("Objective sensitivity for variable %s is infinite\n",
vname);
      }

      GRBfreemodel(b);
      b = NULL;
    }
  }
```

```
QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free models */

  GRBfreemodel(a);
  GRBfreemodel(b);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：sos_c.c

sos_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example creates a very simple Special Ordered Set (SOS) model.
   The model consists of 3 continuous variables, no linear constraints,
   and a pair of SOS constraints of type 1. */

#include <stdlib.h>
#include <stdio.h>
#include "gurobi_c.h"

int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0;
  double    x[3];
  double    obj[3];
  double    ub[3];
  int       sostype[2];
  int       sosbeg[2];
  int       sosind[4];
  double    soswt[4];
  int       optimstatus;
  double    objval;
  int       zero = 0;

  /* Create environment */

  error = GRBloadenv(&env, "sos.log");
  if (error || env == NULL) {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create an empty model */

  error = GRBnewmodel(env, &model, "sos", 0, NULL, NULL, NULL, NULL,
NULL);
  if (error) goto QUIT;


  /* Add variables */

  obj[0] = -2; obj[1] = -1; obj[2] = -1;
  ub[0] = 1.0; ub[1] = 1.0; ub[2] = 2.0;
  error = GRBaddvars(model, 3, 0, NULL, NULL, NULL, obj, NULL, ub, NULL,
                     NULL);
  if (error) goto QUIT;

  /* Integrate new variables */
  error = GRBupdatemodel(model);
  if (error) goto QUIT;

  /* Build first SOS1: x0=0 or x1=0 */

  sosind[0] = 0; sosind[1] = 1;
  soswt[0] = 1.0; soswt[1] = 2.0;
  sosbeg[0] = 0; sostype[0] = GRB_SOS_TYPE1;

  /* Build second SOS1: x0=0 or x2=0 */

  sosind[2] = 0; sosind[3] = 2;
  soswt[2] = 1.0; soswt[3] = 2.0;
  sosbeg[1] = 2; sostype[1] = GRB_SOS_TYPE1;

  /* Add SOSs to model */

  error = GRBaddsos(model, 2, 4, sostype, sosbeg, sosind, soswt);
  if (error) goto QUIT;

  /* Optimize model */

  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Write model to 'sos.lp' */

  error = GRBwrite(model, "sos.lp");
  if (error) goto QUIT;

  /* Capture solution information */

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;

  error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
  if (error) goto QUIT;

  error = GRBgetdblattrarray(model, GRB_DBL_ATTR_X, 0, 3, x);
  if (error) goto QUIT;

  printf("\nOptimization complete\n");
  if (optimstatus == GRB_OPTIMAL) {
    printf("Optimal objective: %.4e\n", objval);

    printf("  x=%.0f, y=%.0f, z=%.0f\n", x[0], x[1], x[2]);
  } else if (optimstatus == GRB_INF_OR_UNBD) {
    printf("Model is infeasible or unbounded\n");
  } else {
    printf("Optimization was stopped early\n");
  }

QUIT:

  /* Error reporting */
  if (error) {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：sudoku_c.c

Sudoku_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */
/*
  Sudoku example.

  The Sudoku board is a 9x9 grid, which is further divided into a 3x3
grid
  of 3x3 grids.  Each cell in the grid must take a value from 0 to 9.
  No two grid cells in the same row, column, or 3x3 subgrid may take
the
  same value.

  In the MIP formulation, binary variables x[i,j,v] indicate whether
  cell <i,j> takes value 'v'.  The constraints are as follows:
    1. Each cell must take exactly one value (sum_v x[i,j,v] = 1)
    2. Each value is used exactly once per row (sum_i x[i,j,v] = 1)
    3. Each value is used exactly once per column (sum_j x[i,j,v] = 1)
    4. Each value is used exactly once per 3x3 subgrid (sum_grid
x[i,j,v] = 1)
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "gurobi_c.h"

#define SUBDIM  3
#define DIM    (SUBDIM*SUBDIM)

int
main(int   argc,
     char *argv[])
{
  FILE     *fp    = NULL;
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       board[DIM][DIM];
  char      inputline[100];
  int       ind[DIM];
  double    val[DIM];
  double    lb[DIM*DIM*DIM];
  char      vtype[DIM*DIM*DIM];
  char     *names[DIM*DIM*DIM];
  char      namestorage[10*DIM*DIM*DIM];
  char     *cursor;
  int       optimstatus;
  double    objval;
  int       zero = 0;
  int       i, j, v, ig, jg, count;
  int       error = 0;

  if (argc < 1) {
    fprintf(stderr, "Usage: sudoku_c datafile\n");
    exit(1);
  }

  fp = fopen(argv[1], "r");
  if (fp == NULL) {
    fprintf(stderr, "Error: unable to open input file %s\n", argv[1]);
    exit(1);
  }

  for (i = 0; i < DIM; i++) {
    fgets(inputline, 100, fp);
    if (strlen(inputline) < 9) {
      fprintf(stderr, "Error: not enough board positions specified\n");
      exit(1);
    }
    for (j = 0; j < DIM; j++) {
      board[i][j] = (int) inputline[j] - (int) '1';
      if (board[i][j] < 0 || board[i][j] >= DIM)
        board[i][j] = -1;
    }
  }

  /* Create an empty model */

  cursor = namestorage;
  for (i = 0; i < DIM; i++) {
    for (j = 0; j < DIM; j++) {
      for (v = 0; v < DIM; v++) {
        if (board[i][j] == v)
          lb[i*DIM*DIM+j*DIM+v] = 1;
        else
          lb[i*DIM*DIM+j*DIM+v] = 0;
        vtype[i*DIM*DIM+j*DIM+v] = GRB_BINARY;

        names[i*DIM*DIM+j*DIM+v] = cursor;
        sprintf(names[i*DIM*DIM+j*DIM+v], "x[%d,%d,%d]", i, j, v+1);
        cursor += strlen(names[i*DIM*DIM+j*DIM+v]) + 1;
      }
    }
  }

  /* Create environment */

  error = GRBloadenv(&env, "sudoku.log");
  if (error || env == NULL) {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create new model */

  error = GRBnewmodel(env, &model, "sudoku", DIM*DIM*DIM, NULL, lb, 
NULL,
                      vtype, names);
  if (error) goto QUIT;

  /* Each cell gets a value */
  for (i = 0; i < DIM; i++) {
    for (j = 0; j < DIM; j++) {
      for (v = 0; v < DIM; v++) {
        ind[v] = i*DIM*DIM + j*DIM + v;
        val[v] = 1.0;
      }

      error = GRBaddconstr(model, DIM, ind, val, GRB_EQUAL, 1.0, NULL);
      if (error) goto QUIT;
    }
  }

  /* Each value must appear once in each row */

  for (v = 0; v < DIM; v++) {
    for (j = 0; j < DIM; j++) {
      for (i = 0; i < DIM; i++) {
        ind[i] = i*DIM*DIM + j*DIM + v;
        val[i] = 1.0;
      }

      error = GRBaddconstr(model, DIM, ind, val, GRB_EQUAL, 1.0, NULL);
      if (error) goto QUIT;
    }
  }

  /* Each value must appear once in each column */

  for (v = 0; v < DIM; v++) {
    for (i = 0; i < DIM; i++) {
      for (j = 0; j < DIM; j++) {
        ind[j] = i*DIM*DIM + j*DIM + v;
        val[j] = 1.0;
      }

      error = GRBaddconstr(model, DIM, ind, val, GRB_EQUAL, 1.0, NULL);
      if (error) goto QUIT;
    }
  }

  /* Each value must appear once in each subgrid */

  for (v = 0; v < DIM; v++) {
    for (ig = 0; ig < SUBDIM; ig++) {
      for (jg = 0; jg < SUBDIM; jg++) {
        count = 0;
        for (i = ig*SUBDIM; i < (ig+1)*SUBDIM; i++) {
          for (j = jg*SUBDIM; j < (jg+1)*SUBDIM; j++) {
            ind[count] = i*DIM*DIM + j*DIM + v;
            val[count] = 1.0;
            count++;
          }
        }

        error = GRBaddconstr(model, DIM, ind, val, GRB_EQUAL, 1.0,
NULL);
```

```
        if (error) goto QUIT;
      }
    }
  }

  /* Optimize model */

  error = GRBoptimize(model);
  if (error) goto QUIT;

  /* Write model to 'sudoku.lp' */

  error = GRBwrite(model, "sudoku.lp");
  if (error) goto QUIT;

  /* Capture solution information */

  error = GRBgetintattr(model, GRB_INT_ATTR_STATUS, &optimstatus);
  if (error) goto QUIT;

  error = GRBgetdblattr(model, GRB_DBL_ATTR_OBJVAL, &objval);
  if (error) goto QUIT;

  printf("\nOptimization complete\n");
  if (optimstatus == GRB_OPTIMAL)
    printf("Optimal objective: %.4e\n", objval);
  else if (optimstatus == GRB_INF_OR_UNBD)
    printf("Model is infeasible or unbounded\n");
  else
    printf("Optimization was stopped early\n");
  printf("\n");

QUIT:

  /* Error reporting */

  if (error) {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：workforce1_c.c

workforce1_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, use IIS to find a
set of
   conflicting constraints. Note that there may be additional conflicts
   besides what is reported via IIS. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "gurobi_c.h"


#define xcol(w,s)  nShifts*w+s
#define MAXSTR     128


int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0, status;
  int       s, w, col;
  int      *cbeg = NULL;
  int      *cind = NULL;
  int       idx;
  double   *cval = NULL;
  char     *sense = NULL;
  char      vname[MAXSTR];
  double    obj;
  int       i, iis, numconstrs;
  char     *cname;

  /* Sample data */
  const int nShifts = 14;
  const int nWorkers = 7;

  /* Sets of days and workers */
  char* Shifts[] =
    { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
      "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12", "Sat13",
      "Sun14" };
  char* Workers[] =
    { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred", "Gu" };

  /* Number of workers required for each shift */
  double shiftRequirements[] =
    { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };
  /* Amount each worker is paid to work one shift */
  double pay[] = { 10, 12, 10, 8, 8, 9, 11 };

  /* Worker availability: 0 if the worker is unavailable for a shift */
  double availability[][14] =
    { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
      { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
      { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
      { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
      { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
      { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
      { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

  /* Create environment */
  error = GRBloadenv(&env, "workforce1.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create initial model */
  error = GRBnewmodel(env, &model, "workforce1", nWorkers * nShifts,
                      NULL, NULL, NULL, NULL, NULL);
  if (error) goto QUIT;

  /* Initialize assignment decision variables:
     x[w][s] == 1 if worker w is assigned
     to shift s. Since an assignment model always produces integer
     solutions, we use continuous variables and solve as an LP. */
  for (w = 0; w < nWorkers; ++w)
  {
    for (s = 0; s < nShifts; ++s)
    {
      col = xcol(w, s);
      sprintf(vname, "%s.%s", Workers[w], Shifts[s]);
      error = GRBsetdblattrelement(model, "UB", col,
availability[w][s]);
      if (error) goto QUIT;
      error = GRBsetdblattrelement(model, "Obj", col, pay[w]);
      if (error) goto QUIT;
      error = GRBsetstrattrelement(model, "VarName", col, vname);
      if (error) goto QUIT;
    }
  }

  /* The objective is to minimize the total pay costs */
  error = GRBsetintattr(model, "ModelSense", 1);
  if (error) goto QUIT;

  /* Make space for constraint data */
  cbeg = malloc(sizeof(int) * nShifts);
  if (!cbeg) goto QUIT;
  cind = malloc(sizeof(int) * nShifts * nWorkers);
  if (!cind) goto QUIT;
  cval = malloc(sizeof(double) * nShifts * nWorkers);
```

```
if (!cval) goto QUIT;
sense = malloc(sizeof(char) * nShifts);
if (!sense) goto QUIT;

/* Constraint: assign exactly shiftRequirements[s] workers
   to each shift s */
idx = 0;
for (s = 0; s < nShifts; ++s)
{
  cbeg[s] = idx;
  sense[s] = GRB_EQUAL;
  for (w = 0; w < nWorkers; ++w)
  {
    cind[idx] = xcol(w, s);
    cval[idx++] = 1.0;
  }
}
error = GRBaddconstrs(model, nShifts, idx, cbeg, cind, cval, sense,
                      shiftRequirements, Shifts);
if (error) goto QUIT;

/* Optimize */
error = GRBoptimize(model);
if (error) goto QUIT;
error = GRBgetintattr(model, "Status", &status);
if (error) goto QUIT;
if (status == GRB_UNBOUNDED)
{
  printf("The model cannot be solved because it is unbounded\n");
  goto QUIT;
}
if (status == GRB_OPTIMAL)
{
  error = GRBgetdblattr(model, "ObjVal", &obj);
  if (error) goto QUIT;
  printf("The optimal objective is %f\n", obj);
  goto QUIT;
}
if ((status != GRB_INF_OR_UNBD) && (status != GRB_INFEASIBLE))
{
  printf("Optimization was stopped with status %i\n", status);
  goto QUIT;
}

/* do IIS */
printf("The model is infeasible; computing IIS\n");
error = GRBcomputeIIS(model);
if (error) goto QUIT;
printf("\nThe following constraint(s) cannot be satisfied:\n");
error = GRBgetintattr(model, "NumConstrs", &numconstrs);
if (error) goto QUIT;
for (i = 0; i < numconstrs; ++i)
{
  error = GRBgetintattrelement(model, "IISConstr", i, &iis);
  if (error) goto QUIT;
  if (iis)
  {
      error = GRBgetstrattrelement(model, "ConstrName", i, &cname);
      if (error) goto QUIT;
      printf("%s\n", cname);
    }
  }


QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free data */

  free(cbeg);
  free(cind);
  free(cval);
  free(sense);

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：workforce2_c.c

workforce2_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, use IIS iteratively
to
   find all conflicting constraints. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "gurobi_c.h"


#define xcol(w,s)  nShifts*w+s
#define MAXSTR     128


int
main(int   argc,
     char *argv[])
{
  GRBenv   *env   = NULL;
  GRBmodel *model = NULL;
  int       error = 0, status;
  int       s, w, col;
  int      *cbeg = NULL;
  int      *cind = NULL;
  int       idx;
  double   *cval = NULL;
  char     *sense = NULL;
  char      vname[MAXSTR];
  double    obj;
  int       i, iis, numconstrs, numremoved = 0;
  char     *cname;
  char    **removed = NULL;

  /* Sample data */
  const int nShifts = 14;
  const int nWorkers = 7;

  /* Sets of days and workers */
  char* Shifts[] =
    { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
      "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12", "Sat13",
      "Sun14" };
  char* Workers[] =
    { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred", "Gu" };

  /* Number of workers required for each shift */
  double shiftRequirements[] =
    { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

  /* Amount each worker is paid to work one shift */
  double pay[] = { 10, 12, 10, 8, 8, 9, 11 };

  /* Worker availability: 0 if the worker is unavailable for a shift */
  double availability[][14] =
    { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
      { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
      { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
      { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
      { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
      { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
      { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

  /* Create environment */
  error = GRBloadenv(&env, "workforce2.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create initial model */
  error = GRBnewmodel(env, &model, "workforce2", nWorkers * nShifts,
                      NULL, NULL, NULL, NULL, NULL);
  if (error) goto QUIT;

  /* Initialize assignment decision variables:
     x[w][s] == 1 if worker w is assigned
     to shift s. Since an assignment model always produces integer
     solutions, we use continuous variables and solve as an LP. */
  for (w = 0; w < nWorkers; ++w)
  {
    for (s = 0; s < nShifts; ++s)
    {
      col = xcol(w, s);
      sprintf(vname, "%s.%s", Workers[w], Shifts[s]);
      error = GRBsetdblattrelement(model, "UB", col,
availability[w][s]);
      if (error) goto QUIT;
      error = GRBsetdblattrelement(model, "Obj", col, pay[w]);
      if (error) goto QUIT;
      error = GRBsetstrattrelement(model, "VarName", col, vname);
      if (error) goto QUIT;
    }
  }

  /* The objective is to minimize the total pay costs */
  error = GRBsetintattr(model, "ModelSense", 1);
  if (error) goto QUIT;

  /* Make space for constraint data */
  cbeg = malloc(sizeof(int) * nShifts);
  if (!cbeg) goto QUIT;
  cind = malloc(sizeof(int) * nShifts * nWorkers);
  if (!cind) goto QUIT;
```

```
cval = malloc(sizeof(double) * nShifts * nWorkers);
if (!cval) goto QUIT;
sense = malloc(sizeof(char) * nShifts);
if (!sense) goto QUIT;

/* Constraint: assign exactly shiftRequirements[s] workers
   to each shift s */
idx = 0;
for (s = 0; s < nShifts; ++s)
{
  cbeg[s] = idx;
  sense[s] = GRB_EQUAL;
  for (w = 0; w < nWorkers; ++w)
  {
    cind[idx] = xcol(w, s);
    cval[idx++] = 1.0;
  }
}
error = GRBaddconstrs(model, nShifts, idx, cbeg, cind, cval, sense,
                      shiftRequirements, Shifts);
if (error) goto QUIT;

/* Optimize */
error = GRBoptimize(model);
if (error) goto QUIT;
error = GRBgetintattr(model, "Status", &status);
if (error) goto QUIT;
if (status == GRB_UNBOUNDED)
{
  printf("The model cannot be solved because it is unbounded\n");
  goto QUIT;
}
if (status == GRB_OPTIMAL)
{
  error = GRBgetdblattr(model, "ObjVal", &obj);
  if (error) goto QUIT;
  printf("The optimal objective is %f\n", obj);
  goto QUIT;
}
if ((status != GRB_INF_OR_UNBD) && (status != GRB_INFEASIBLE))
{
  printf("Optimization was stopped with status %i\n", status);
  goto QUIT;
}

/* do IIS */
printf("The model is infeasible; computing IIS\n");

/* Loop until we reduce to a model that can be solved */
error = GRBgetintattr(model, "NumConstrs", &numconstrs);
if (error) goto QUIT;
removed = calloc(numconstrs, sizeof(char*));
if (!removed) goto QUIT;
while (1)
{
  error = GRBcomputeIIS(model);
  if (error) goto QUIT;
```

```
    printf("\nThe following constraint cannot be satisfied:\n");
    for (i = 0; i < numconstrs; ++i)
    {
      error = GRBgetintattrelement(model, "IISConstr", i, &iis);
      if (error) goto QUIT;
      if (iis)
      {
        error = GRBgetstrattrelement(model, "ConstrName", i, &cname);
        if (error) goto QUIT;
        printf("%s\n", cname);
        /* Remove a single constraint from the model */
        removed[numremoved] = malloc(sizeof(char) * (1+strlen(cname)));
        if (!removed[numremoved]) goto QUIT;
        strcpy(removed[numremoved++], cname);
        cind[0] = i;
        error = GRBdelconstrs(model, 1, cind);
        if (error) goto QUIT;
        break;
      }
    }

    printf("\n");
    error = GRBoptimize(model);
    if (error) goto QUIT;
    error = GRBgetintattr(model, "Status", &status);
    if (error) goto QUIT;
    if (status == GRB_UNBOUNDED)
    {
      printf("The model cannot be solved because it is unbounded\n");
      goto QUIT;
    }
    if (status == GRB_OPTIMAL)
    {
      break;
    }
    if ((status != GRB_INF_OR_UNBD) && (status != GRB_INFEASIBLE))
    {
      printf("Optimization was stopped with status %i\n", status);
      goto QUIT;
    }
  }

  printf("\nThe following constraints were removed to get a feasible
LP:\n");
  for (i = 0; i < numremoved; ++i)
  {
    printf("%s ", removed[i]);
  }
  printf("\n");



QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free data */

  free(cbeg);
  free(cind);
  free(cval);
  free(sense);
  for (i=0; i<numremoved; ++i) {
    free(removed[i]);
  }
  free(removed);

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：workforce3_c.c

workforce3_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, add slack variables
   to determine which constraints cannot be satisfied, and how much
   they need to be relaxed. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "gurobi_c.h"


#define xcol(w,s)  nShifts*w+s
#define MAXSTR     128


int
main(int   argc,
     char *argv[])
{
  GRBenv   *env = NULL;
  GRBmodel *model = NULL;
  int       error = 0, status;
  int       s, w, col;
  int      *cbeg = NULL;
  int      *cind = NULL;
  int       idx;
  double   *cval = NULL;
  char     *sense = NULL;
  char      vname[MAXSTR];
  double    obj;
  int       i, j, numvars, numconstrs;
  int      *vbeg = NULL;
  int      *vind = NULL;
  double   *vval = NULL;
  double   *vobj = NULL;
  double    sol;
  char     *cname, *sname;
  int       varnamesct = 0;
  char    **varnames = NULL;

  /* Sample data */
  const int nShifts = 14;
  const int nWorkers = 7;

  /* Sets of days and workers */
  char* Shifts[] =
    { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
      "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12", "Sat13",
      "Sun14" };
  char* Workers[] =
    { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred", "Gu" };

  /* Number of workers required for each shift */
  double shiftRequirements[] =
    { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

  /* Amount each worker is paid to work one shift */
  double pay[] = { 10, 12, 10, 8, 8, 9, 11 };

  /* Worker availability: 0 if the worker is unavailable for a shift */
  double availability[][14] =
    { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
      { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
      { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
      { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
      { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
      { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
      { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

  /* Create environment */
  error = GRBloadenv(&env, "workforce3.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create initial model */
  error = GRBnewmodel(env, &model, "workforce3", nWorkers * nShifts,
                      NULL, NULL, NULL, NULL, NULL);
  if (error) goto QUIT;

  /* Initialize assignment decision variables:
     x[w][s] == 1 if worker w is assigned
     to shift s. Since an assignment model always produces integer
     solutions, we use continuous variables and solve as an LP. */
  for (w = 0; w < nWorkers; ++w)
  {
    for (s = 0; s < nShifts; ++s)
    {
      col = xcol(w, s);
      sprintf(vname, "%s.%s", Workers[w], Shifts[s]);
      error = GRBsetdblattrelement(model, "UB", col,
availability[w][s]);
      if (error) goto QUIT;
      error = GRBsetdblattrelement(model, "Obj", col, pay[w]);
      if (error) goto QUIT;
      error = GRBsetstrattrelement(model, "VarName", col, vname);
      if (error) goto QUIT;
    }
  }

  /* The objective is to minimize the total pay costs */
  error = GRBsetintattr(model, "ModelSense", 1);
  if (error) goto QUIT;
```

```
/* Make space for constraint data */
cbeg = malloc(sizeof(int) * nShifts);
if (!cbeg) goto QUIT;
cind = malloc(sizeof(int) * nShifts * nWorkers);
if (!cind) goto QUIT;
cval = malloc(sizeof(double) * nShifts * nWorkers);
if (!cval) goto QUIT;
sense = malloc(sizeof(char) * nShifts);
if (!sense) goto QUIT;

/* Constraint: assign exactly shiftRequirements[s] workers
   to each shift s */
idx = 0;
for (s = 0; s < nShifts; ++s)
{
  cbeg[s] = idx;
  sense[s] = GRB_EQUAL;
  for (w = 0; w < nWorkers; ++w)
  {
    cind[idx] = xcol(w, s);
    cval[idx++] = 1.0;
  }
}
error = GRBaddconstrs(model, nShifts, idx, cbeg, cind, cval, sense,
                      shiftRequirements, Shifts);
if (error) goto QUIT;

/* Optimize */
error = GRBoptimize(model);
if (error) goto QUIT;
error = GRBgetintattr(model, "Status", &status);
if (error) goto QUIT;
if (status == GRB_UNBOUNDED)
{
  printf("The model cannot be solved because it is unbounded\n");
  goto QUIT;
}
if (status == GRB_OPTIMAL)
{
  error = GRBgetdblattr(model, "ObjVal", &obj);
  if (error) goto QUIT;
  printf("The optimal objective is %f\n", obj);
  goto QUIT;
}
if ((status != GRB_INF_OR_UNBD) && (status != GRB_INFEASIBLE))
{
  printf("Optimization was stopped with status %i\n", status);
  goto QUIT;
}

/* Add slack variables to make the model feasible */
printf("The model is infeasible; adding slack variables\n");

/* Determine the matrix size before adding the slacks */
error = GRBgetintattr(model, "NumVars", &numvars);
if (error) goto QUIT;
```

```
  error = GRBgetintattr(model, "NumConstrs", &numconstrs);
  if (error) goto QUIT;

  /* Set original objective coefficients to zero */
  for (j = 0; j < numvars; ++j)
  {
    error = GRBsetdblattrelement(model, "Obj", j, 0.0);
    if (error) goto QUIT;
  }

  /* Add a new slack variable to each shift constraint so that the
shifts
     can be satisfied */
  vbeg = malloc(sizeof(int) * numconstrs);
  if (!vbeg) goto QUIT;
  vind = malloc(sizeof(int) * numconstrs);
  if (!vind) goto QUIT;
  vval = malloc(sizeof(double) * numconstrs);
  if (!vval) goto QUIT;
  vobj = malloc(sizeof(double) * numconstrs);
  if (!vobj) goto QUIT;
  varnames = calloc(numconstrs, sizeof(char*));
  if (!varnames) goto QUIT;
  for (i = 0; i < numconstrs; ++i)
  {
    vbeg[i] = i;
    vind[i] = i;
    vval[i] = 1.0;
    vobj[i] = 1.0;
    error = GRBgetstrattrelement(model, "ConstrName", i, &cname);
    if (error) goto QUIT;
    varnames[i] = malloc(sizeof(char*) * (6 + strlen(cname)));
    if (!varnames[i]) goto QUIT;
    varnamesct++;
    strcpy(varnames[i], cname);
    strcat(varnames[i], "Slack");
  }
  error = GRBaddvars(model, numconstrs, numconstrs,
                     vbeg, vind, vval, vobj, NULL, NULL, NULL,
varnames);
  if (error) goto QUIT;

  error = GRBupdatemodel(model);
  if (error) goto QUIT;

  /* Solve the model with slacks */
  error = GRBoptimize(model);
  if (error) goto QUIT;
  error = GRBgetintattr(model, "Status", &status);
  if (error) goto QUIT;
  if ((status == GRB_INF_OR_UNBD) || (status == GRB_INFEASIBLE) ||
      (status == GRB_UNBOUNDED))
  {
    printf("The model with slacks cannot be solved "
           "because it is infeasible or unbounded\n");
    goto QUIT;
  }
  if (status != GRB_OPTIMAL)
  {
    printf("Optimization was stopped with status %i\n", status);
    goto QUIT;
  }

  printf("\nSlack values:\n");
  for (j = numvars; j < numvars + numconstrs; ++j)
  {
    error = GRBgetdblattrelement(model, "X", j, &sol);
    if (error) goto QUIT;
    if (sol > 1e-6)
    {
      error = GRBgetstrattrelement(model, "VarName", j, &sname);
      if (error) goto QUIT;
      printf("%s = %f\n", sname, sol);
    }
  }

QUIT:

  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free data */

  free(cbeg);
  free(cind);
  free(cval);
  free(sense);
  free(vbeg);
  free(vind);
  free(vval);
  free(vobj);
  for (i = 0; i < varnamesct; ++i)
  {
    free(varnames[i]);
  }
  free(varnames);

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}
```

页面：workforce4_c.c

workforce4_c.c

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. We use Pareto optimization to solve the model:
   first, we minimize the linear sum of the slacks. Then, we constrain
   the sum of the slacks, and we minimize a quadratic objective that
   tries to balance the workload among the workers. */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "gurobi_c.h"

int solveAndPrint(GRBmodel* model,
                  int nShifts, int nWorkers, char** Workers,
                  int* status);


#define xcol(w,s)         nShifts*w+s
#define slackcol(s)       nShifts*nWorkers+s
#define totSlackcol       nShifts*(nWorkers+1)
#define totShiftscol(w)   nShifts*(nWorkers+1)+1+w
#define avgShiftscol      (nShifts+1)*(nWorkers+1)
#define diffShiftscol(w)  (nShifts+1)*(nWorkers+1)+1+w
#define MAXSTR     128


int
main(int   argc,
     char *argv[])
{
  GRBenv   *env = NULL;
  GRBmodel *model = NULL;
  int       error = 0, status;
  int       s, w, col;
  int      *cbeg = NULL;
  int      *cind = NULL;
  int       idx;
  double   *cval = NULL;
  char     *sense = NULL;
  char      vname[MAXSTR], cname[MAXSTR];
  double    val;

  /* Sample data */
  const int nShifts = 14;
  const int nWorkers = 7;

  /* Sets of days and workers */
  char* Shifts[] =
    { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
      "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12", "Sat13",
      "Sun14" };
  char* Workers[] =
    { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred", "Gu" };

  /* Number of workers required for each shift */
  double shiftRequirements[] =
    { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

  /* Worker availability: 0 if the worker is unavailable for a shift */
  double availability[][14] =
    { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
      { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
      { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
      { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
      { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
      { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
      { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

  /* Create environment */
  error = GRBloadenv(&env, "workforce3.log");
  if (error || env == NULL)
  {
    fprintf(stderr, "Error: could not create environment\n");
    exit(1);
  }

  /* Create initial model */
  error = GRBnewmodel(env, &model, "workforce3",
                      (nShifts + 1) * (nWorkers + 1),
                      NULL, NULL, NULL, NULL, NULL);
  if (error) goto QUIT;

  /* Initialize assignment decision variables:
     x[w][s] == 1 if worker w is assigned to shift s.
     This is no longer a pure assignment model, so we must
     use binary variables. */
  for (w = 0; w < nWorkers; ++w)
  {
    for (s = 0; s < nShifts; ++s)
    {
      col = xcol(w, s);
      sprintf(vname, "%s.%s", Workers[w], Shifts[s]);
      error = GRBsetcharattrelement(model, "VType", col, GRB_BINARY);
      if (error) goto QUIT;
      error = GRBsetdblattrelement(model, "UB", col,
availability[w][s]);
      if (error) goto QUIT;
      error = GRBsetstrattrelement(model, "VarName", col, vname);
      if (error) goto QUIT;
    }
  }

  /* Initialize slack decision variables */
  for (s = 0; s < nShifts; ++s)
  {
    sprintf(vname, "%sSlack", Shifts[s]);
    error = GRBsetstrattrelement(model, "VarName", slackcol(s), vname);
    if (error) goto QUIT;
  }

  /* Initialize total slack decision variable */
  error = GRBsetstrattrelement(model, "VarName", totSlackcol,
"totSlack");
  if (error) goto QUIT;

  /* Initialize variables to count the total shifts worked by each
worker */
  for (w = 0; w < nWorkers; ++w)
  {
    sprintf(vname, "%sTotShifts", Workers[w]);
    error = GRBsetstrattrelement(model, "VarName", totShiftscol(w),
vname);
    if (error) goto QUIT;
  }

  /* The objective is to minimize the sum of the slacks */
  error = GRBsetintattr(model, "ModelSense", 1);
  if (error) goto QUIT;
  error = GRBsetdblattrelement(model, "Obj", totSlackcol, 1.0);
  if (error) goto QUIT;

  /* Make space for constraint data */
  cbeg = malloc(sizeof(int) * nShifts);
  if (!cbeg) goto QUIT;
  cind = malloc(sizeof(int) * nShifts * (nWorkers + 1));
  if (!cind) goto QUIT;
  cval = malloc(sizeof(double) * nShifts * (nWorkers + 1));
  if (!cval) goto QUIT;
  sense = malloc(sizeof(char) * nShifts);
  if (!sense) goto QUIT;

  /* Constraint: assign exactly shiftRequirements[s] workers
     to each shift s, plus the slack */
  idx = 0;
  for (s = 0; s < nShifts; ++s)
  {
    cbeg[s] = idx;
    sense[s] = GRB_EQUAL;
    for (w = 0; w < nWorkers; ++w)
    {
      cind[idx] = xcol(w, s);
      cval[idx++] = 1.0;
    }
    cind[idx] = slackcol(s);
    cval[idx++] = 1.0;
  }
  error = GRBaddconstrs(model, nShifts, idx, cbeg, cind, cval, sense,
                        shiftRequirements, Shifts);
  if (error) goto QUIT;

  /* Constraint: set totSlack column equal to the total slack */
  idx = 0;
  for (s = 0; s < nShifts; ++s)
  {
    cind[idx] = slackcol(s);
    cval[idx++] = 1.0;
  }
  cind[idx] = totSlackcol;
  cval[idx++] = -1.0;
  error = GRBaddconstr(model, idx, cind, cval, GRB_EQUAL,
                       0.0, "totSlack");
  if (error) goto QUIT;

  /* Constraint: compute the total number of shifts for each worker */
  for (w = 0; w < nWorkers; ++w)
  {
    idx = 0;
    for (s = 0; s < nShifts; ++s)
    {
      cind[idx] = xcol(w,s);
      cval[idx++] = 1.0;
    }
    sprintf(cname, "totShifts%s", Workers[w]);
    cind[idx] = totShiftscol(w);
    cval[idx++] = -1.0;
    error = GRBaddconstr(model, idx, cind, cval, GRB_EQUAL, 0.0, cname);
    if (error) goto QUIT;
  }

  /* Optimize */
  error = solveAndPrint(model, nShifts, nWorkers, Workers, &status);
  if (error) goto QUIT;
  if (status != GRB_OPTIMAL) goto QUIT;

  /* Constrain the slack by setting its upper and lower bounds */
  error = GRBgetdblattrelement(model, "X", totSlackcol, &val);
  if (error) goto QUIT;
  error = GRBsetdblattrelement(model, "UB", totSlackcol, val);
  if (error) goto QUIT;
  error = GRBsetdblattrelement(model, "LB", totSlackcol, val);
  if (error) goto QUIT;

  /* Variable to count the average number of shifts worked */
  error = GRBaddvar(model, 0, NULL, NULL, 0, 0, GRB_INFINITY,
GRB_CONTINUOUS,
                    "avgShifts");
  if (error) goto QUIT;

  /* Variables to count the difference from average for each worker;
     note that these variables can take negative values. */
  error = GRBaddvars(model, nWorkers, 0, NULL, NULL, NULL, NULL, NULL,
NULL,
                     NULL, NULL);
  if (error) goto QUIT;

  error = GRBupdatemodel(model);
  if (error) goto QUIT;

  for (w = 0; w < nWorkers; ++w)
  {
    sprintf(vname, "%sDiff", Workers[w]);
    error = GRBsetstrattrelement(model, "VarName", diffShiftscol(w),
vname);
    if (error) goto QUIT;
    error = GRBsetdblattrelement(model, "LB", diffShiftscol(w), -
GRB_INFINITY);
    if (error) goto QUIT;
  }

  /* Constraint: compute the average number of shifts worked */
  idx = 0;
  for (w = 0; w < nWorkers; ++w) {
    cind[idx] = totShiftscol(w);
    cval[idx++] = 1.0;
  }
  cind[idx] = avgShiftscol;
  cval[idx++] = -nWorkers;
  error = GRBaddconstr(model, idx, cind, cval, GRB_EQUAL, 0.0,
"avgShifts");
  if (error) goto QUIT;

  // Constraint: compute the difference from the average number of
shifts
  for (w = 0; w < nWorkers; ++w) {
    cind[0] = totShiftscol(w);
    cval[0] = 1.0;
    cind[1] = avgShiftscol;
    cval[1] = -1.0;
    cind[2] = diffShiftscol(w);
    cval[2] = -1.0;
    sprintf(cname, "%sDiff", Workers[w]);
    error = GRBaddconstr(model, 3, cind, cval, GRB_EQUAL, 0.0, cname);
    if (error) goto QUIT;
  }

  /* Objective: minimize the sum of the square of the difference from
the
     average number of shifts worked */
  error = GRBsetdblattrelement(model, "Obj", totSlackcol, 0.0);
  if (error) goto QUIT;

  for (w = 0; w < nWorkers; ++w)
  {
    cind[w] = diffShiftscol(w);
    cval[w] = 1.0;
  }
  error = GRBaddqpterms(model, nWorkers, cind, cind, cval);
  if (error) goto QUIT;

  /* Optimize */
  error = solveAndPrint(model, nShifts, nWorkers, Workers, &status);
  if (error) goto QUIT;
  if (status != GRB_OPTIMAL) goto QUIT;


QUIT:
  /* Error reporting */

  if (error)
  {
    printf("ERROR: %s\n", GRBgeterrormsg(env));
    exit(1);
  }

  /* Free data */

  free(cbeg);
  free(cind);
  free(cval);
  free(sense);

  /* Free model */

  GRBfreemodel(model);

  /* Free environment */

  GRBfreeenv(env);

  return 0;
}


int solveAndPrint(GRBmodel* model,
                  int nShifts, int nWorkers, char** Workers,
                  int* status)
{
  int error, w;
  double val;

  error = GRBoptimize(model);
  if (error) return error;

  error = GRBgetintattr(model, "Status", status);
  if (error) return error;

  if ((*status == GRB_INF_OR_UNBD) || (*status == GRB_INFEASIBLE) ||
      (*status == GRB_UNBOUNDED))
  {
    printf("The model cannot be solved "
           "because it is infeasible or unbounded\n");
    return 0;
  }
  if (*status != GRB_OPTIMAL)
  {
    printf("Optimization was stopped with status %i\n", *status);
    return 0;
  }

  /* Print total slack and the number of shifts worked for each worker
*/
  error = GRBgetdblattrelement(model, "X", totSlackcol, &val);
  if (error) return error;
  printf("\nTotal slack required: %f\n", val);
  for (w = 0; w < nWorkers; ++w)
  {
    error = GRBgetdblattrelement(model, "X", totShiftscol(w), &val);
    if (error) return error;
    printf("%s worked %f shifts\n", Workers[w], val);
  }
  printf("\n");
  return 0;
}
```

页面：C++ Examples

C++ 示例

本节包含了所有 Gurobi C++ 的示例源代码。相同的源代码也可以在 Gurobi 发布包的 *examples/c++* 目录中找到。

小节

页面：callback_c++.cpp

callback_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP or a MIP from a file, sets a callback
   to monitor the optimization progress and to output some progress
   information to the screen and to a log file. If it is a MIP and 10%
   gap is reached, then it aborts */

#include "gurobi_c++.h"
#include <cmath>
using namespace std;

class mycallback: public GRBCallback
{
  public:
    GRBVar* vars;
    int numvars;
    int lastmsg;
    mycallback(GRBVar* xvars, int xnumvars) {
      vars    = xvars;
      numvars = xnumvars;
      lastmsg = -100;
    }
  protected:
    void callback () {
      try {
        if (where == GRB_CB_MESSAGE) {
          string msg = getStringInfo(GRB_CB_MSG_STRING);
          cout << msg;
        } else if (where == GRB_CB_PRESOLVE) {
          int cdels = getIntInfo(GRB_CB_PRE_COLDEL);
          int rdels = getIntInfo(GRB_CB_PRE_ROWDEL);
          cout << cdels << " columns and " << rdels
               << " rows are removed" << endl;
        } else if (where == GRB_CB_SIMPLEX) {
          int itcnt = (int) getDoubleInfo(GRB_CB_SPX_ITRCNT);
          if (itcnt%100 == 0) {
            double obj  = getDoubleInfo(GRB_CB_SPX_OBJVAL);
            double pinf = getDoubleInfo(GRB_CB_SPX_PRIMINF);
            double dinf = getDoubleInfo(GRB_CB_SPX_DUALINF);
            int  ispert = getIntInfo(GRB_CB_SPX_ISPERT);
            char ch;
            if (ispert == 0)      ch = ' ';
            else if (ispert == 1) ch = 'S';
            else                  ch = 'P';
            cout << itcnt << "  " << obj << ch << "  " << pinf
                 << "  " << dinf << endl;
          }
        } else if (where == GRB_CB_MIP) {
          int nodecnt = (int) getDoubleInfo(GRB_CB_MIP_NODCNT);
          if (nodecnt - lastmsg >= 100) {
            lastmsg = nodecnt;
            double objbst = getDoubleInfo(GRB_CB_MIP_OBJBST);
            double objbnd = getDoubleInfo(GRB_CB_MIP_OBJBND);
            if (fabs(objbst - objbnd) < 0.1 * (1.0 + fabs(objbst)))
              abort();
            int actnodes = (int) getDoubleInfo(GRB_CB_MIP_NODLFT);
            int itcnt    = (int) getDoubleInfo(GRB_CB_MIP_ITRCNT);
            int solcnt   = getIntInfo(GRB_CB_MIP_SOLCNT);
            int cutcnt   = getIntInfo(GRB_CB_MIP_CUTCNT);
            cout << nodecnt << " " <<  actnodes << " " <<  itcnt << " "
                 << objbst << " " <<  objbnd << " "  <<  solcnt << " "
                 <<  cutcnt << endl;
          }
        } else if (where == GRB_CB_MIPSOL) {
          double obj     = getDoubleInfo(GRB_CB_MIPSOL_OBJ);
          int    nodecnt = (int) getDoubleInfo(GRB_CB_MIPSOL_NODCNT);
          double* x = getSolution(vars, numvars);
          cout << "**** New solution at node " << nodecnt << ", obj "
                             << obj << ", x[0] = " << x[0] << "****" <<
endl;
          delete[] x;
        }
      } catch (GRBException e) {
        cout << "Error number: " << e.getErrorCode() << endl;
        cout << e.getMessage() << endl;
      } catch (...) {
        cout << "Error during callback" << endl;
      }
    }
};

int
main(int   argc,
     char *argv[])
{
  if (argc < 2) {
    cout << "Usage: callback_c++ filename" << endl;
    return 1;
  }

  GRBEnv *env = 0;
  GRBVar *vars = 0;
  try {
    env = new GRBEnv();
    GRBModel model = GRBModel(*env, argv[1]);

    model.getEnv().set(GRB_IntParam_OutputFlag, 0);

    int numvars = model.get(GRB_IntAttr_NumVars);

    vars = model.getVars();

    mycallback cb = mycallback(vars, numvars);
    model.setCallback(&cb);

    model.optimize();

    for (int j = 0; j < numvars; j++) {
      GRBVar v = vars[j];
      cout << v.get(GRB_StringAttr_VarName) << " "
           << v.get(GRB_DoubleAttr_X) << endl;
    }
  } catch (GRBException e) {
    cout << "Error number: " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  } catch (...) {
    cout << "Error during optimization" << endl;
  }

  delete[] vars;
  delete env;
  return 0;
}
```

页面：diet_c++.cpp

diet_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Solve the classic diet model, showing how to add constraints
   to an existing model. */

#include "gurobi_c++.h"
using namespace std;

void printSolution(GRBModel& model, int nCategories, int nFoods,
                   GRBVar* buy, GRBVar* nutrition) throw(GRBException);

int
main(int argc,
     char *argv[])
{
  GRBEnv* env = 0;
  GRBVar* nutrition = 0;
  GRBVar* buy = 0;
  try
  {

    // Nutrition guidelines, based on
    // USDA Dietary Guidelines for Americans, 2005
    // http://www.health.gov/DietaryGuidelines/dga2005/
    const int nCategories = 4;
    string Categories[] =
      { "calories", "protein", "fat", "sodium" };
    double minNutrition[] = { 1800, 91, 0, 0 };
    double maxNutrition[] = { 2200, GRB_INFINITY, 65, 1779 };

    // Set of foods
    const int nFoods = 9;
    string Foods[] =
      { "hamburger", "chicken", "hot dog", "fries",
        "macaroni", "pizza", "salad", "milk", "ice cream" };
    double cost[] =
      { 2.49, 2.89, 1.50, 1.89, 2.09, 1.99, 2.49, 0.89, 1.59 };

    // Nutrition values for the foods
    double nutritionValues[][nCategories] = {
                      { 410, 24, 26, 730 },    // hamburger
                      { 420, 32, 10, 1190 },   // chicken
                      { 560, 20, 32, 1800 },   // hot dog
                      { 380, 4, 19, 270 },     // fries
                      { 320, 12, 10, 930 },    // macaroni
                      { 320, 15, 12, 820 },    // pizza
                      { 320, 31, 12, 1230 },   // salad
                      { 100, 8, 2.5, 125 },    // milk
                      { 330, 8, 10, 180 }      // ice cream
                    };

    // Model
    env = new GRBEnv();
    GRBModel model = GRBModel(*env);
    model.set(GRB_StringAttr_ModelName, "diet");

    // Create decision variables for the nutrition information,
    // which we limit via bounds
    nutrition = model.addVars(minNutrition, maxNutrition, 0, 0,
                              Categories, nCategories);

    // Create decision variables for the foods to buy
    buy = model.addVars(0, 0, cost, 0, Foods, nFoods);

    // The objective is to minimize the costs
    model.set(GRB_IntAttr_ModelSense, 1);

    // Update model to integrate new variables
    model.update();

    // Nutrition constraints
    for (int i = 0; i < nCategories; ++i)
    {
      GRBLinExpr ntot = 0;
      for (int j = 0; j < nFoods; ++j)
      {
        ntot += nutritionValues[j][i] * buy[j];
      }
      model.addConstr(ntot == nutrition[i], Categories[i]);
    }

    // Use barrier to solve model
    model.getEnv().set(GRB_IntParam_Method, GRB_METHOD_BARRIER);

    // Solve
    model.optimize();
    printSolution(model, nCategories, nFoods, buy, nutrition);

    cout << "\nAdding constraint: at most 6 servings of dairy" << endl;
    model.addConstr(buy[7] + buy[8] <= 6.0, "limit_dairy");

    // Solve
    model.optimize();
    printSolution(model, nCategories, nFoods, buy, nutrition);

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Exception during optimization" << endl;
  }

  delete[] nutrition;
  delete[] buy;
  delete env;
  return 0;
}

void printSolution(GRBModel& model, int nCategories, int nFoods,
                   GRBVar* buy, GRBVar* nutrition) throw(GRBException)
{
  if (model.get(GRB_IntAttr_Status) == GRB_OPTIMAL)
  {
    cout << "\nCost: " << model.get(GRB_DoubleAttr_ObjVal) << endl;
    cout << "\nBuy:" << endl;
    for (int j = 0; j < nFoods; ++j)
    {
      if (buy[j].get(GRB_DoubleAttr_X) > 0.0001)
      {
        cout << buy[j].get(GRB_StringAttr_VarName) << " " <<
        buy[j].get(GRB_DoubleAttr_X) << endl;
      }
    }
    cout << "\nNutrition:" << endl;
    for (int i = 0; i < nCategories; ++i)
    {
      cout << nutrition[i].get(GRB_StringAttr_VarName) << " " <<
      nutrition[i].get(GRB_DoubleAttr_X) << endl;
    }
  }
  else
  {
    cout << "No solution" << endl;
  }
}
```

页面：facility_c++.cpp

facility_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Facility location: a company currently ships its product from 5
plants
   to 4 warehouses. It is considering closing some plants to reduce
   costs. What plant(s) should the company close, in order to minimize
   transportation and fixed costs?

   Based on an example from Frontline Systems:
   http://www.solver.com/disfacility.htm
   Used with permission.
 */

#include "gurobi_c++.h"
#include <sstream>
using namespace std;

int
main(int argc,
     char *argv[])
{
  GRBEnv* env = 0;
  GRBVar* open = 0;
  GRBVar** transport = 0;
  int transportCt = 0;
  try
  {

    // Number of plants and warehouses
    const int nPlants = 5;
    const int nWarehouses = 4;

    // Warehouse demand in thousands of units
    double Demand[] = { 15, 18, 14, 20 };

    // Plant capacity in thousands of units
    double Capacity[] = { 20, 22, 17, 19, 18 };

    // Fixed costs for each plant
    double FixedCosts[] =
      { 12000, 15000, 17000, 13000, 16000 };

    // Transportation costs per thousand units
    double TransCosts[][nPlants] = {
                                     { 4000, 2000, 3000, 2500, 4500 },
                                     { 2500, 2600, 3400, 3000, 4000 },
                                     { 1200, 1800, 2600, 4100, 3000 },
                                     { 2200, 2600, 3100, 3700, 3200 }
                                   };

    // Model
    env = new GRBEnv();
```

```
GRBModel model = GRBModel(*env);
model.set(GRB_StringAttr_ModelName, "facility");

// Plant open decision variables: open[p] == 1 if plant p is open.
open = model.addVars(nPlants, GRB_BINARY);
model.update();
for (int p = 0; p < nPlants; ++p)
{
  ostringstream vname;
  vname << "Open" << p;
  open[p].set(GRB_DoubleAttr_Obj, FixedCosts[p]);
  open[p].set(GRB_StringAttr_VarName, vname.str());
}

// Transportation decision variables: how much to transport from
// a plant p to a warehouse w
transport = new GRBVar* [nWarehouses];
for (int w = 0; w < nWarehouses; ++w)
{
  transport[w] = model.addVars(nPlants);
  transportCt++;
  model.update();
  for (int p = 0; p < nPlants; ++p)
  {
    ostringstream vname;
    vname << "Trans" << p << "." << w;
    transport[w][p].set(GRB_DoubleAttr_Obj, TransCosts[w][p]);
    transport[w][p].set(GRB_StringAttr_VarName, vname.str());
  }
}

// The objective is to minimize the total fixed and variable costs
model.set(GRB_IntAttr_ModelSense, 1);

// Update model to integrate new variables
model.update();

// Production constraints
// Note that the right-hand limit sets the production to zero if
// the plant is closed
for (int p = 0; p < nPlants; ++p)
{
  GRBLinExpr ptot = 0;
  for (int w = 0; w < nWarehouses; ++w)
  {
    ptot += transport[w][p];
  }
  ostringstream cname;
  cname << "Capacity" << p;
  model.addConstr(ptot <= Capacity[p] * open[p], cname.str());
}

// Demand constraints
for (int w = 0; w < nWarehouses; ++w)
{
  GRBLinExpr dtot = 0;
  for (int p = 0; p < nPlants; ++p)
```

```
      {
        dtot += transport[w][p];
      }
      ostringstream cname;
      cname << "Demand" << w;
      model.addConstr(dtot == Demand[w], cname.str());
    }

    // Guess at the starting point: close the plant with the highest
    // fixed costs; open all others

    // First, open all plants
    for (int p = 0; p < nPlants; ++p)
    {
      open[p].set(GRB_DoubleAttr_Start, 1.0);
    }

    // Now close the plant with the highest fixed cost
    cout << "Initial guess:" << endl;
    double maxFixed = -GRB_INFINITY;
    for (int p = 0; p < nPlants; ++p)
    {
      if (FixedCosts[p] > maxFixed)
      {
        maxFixed = FixedCosts[p];
      }
    }
    for (int p = 0; p < nPlants; ++p)
    {
      if (FixedCosts[p] == maxFixed)
      {
        open[p].set(GRB_DoubleAttr_Start, 0.0);
        cout << "Closing plant " << p << endl << endl;
        break;
      }
    }

    // Use barrier to solve root relaxation
    model.getEnv().set(GRB_IntParam_Method, GRB_METHOD_BARRIER);

    // Solve
    model.optimize();

    // Print solution
    cout << "\nTOTAL COSTS: " << model.get(GRB_DoubleAttr_ObjVal) <<
endl;
    cout << "SOLUTION:" << endl;
    for (int p = 0; p < nPlants; ++p)
    {
      if (open[p].get(GRB_DoubleAttr_X) == 1.0)
      {
        cout << "Plant " << p << " open:" << endl;
        for (int w = 0; w < nWarehouses; ++w)
        {
          if (transport[w][p].get(GRB_DoubleAttr_X) > 0.0001)
          {
            cout << "  Transport " <<
            transport[w][p].get(GRB_DoubleAttr_X) <<
            " units to warehouse " << w << endl;
          }
        }
      }
      else
      {
        cout << "Plant " << p << " closed!" << endl;
      }
    }

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Exception during optimization" << endl;
  }

  delete[] open;
  for (int i = 0; i < transportCt; ++i) {
    delete[] transport[i];
  }
  delete[] transport;
  delete env;
  return 0;
}
```

页面：feasopt_c++.cpp

feasopt_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads a MIP model from a file, adds artificial
   variables to each constraint, and then minimizes the sum of the
   artificial variables.  A solution with objective zero corresponds
   to a feasible solution to the input model. */

#include "gurobi_c++.h"
using namespace std;

int
main(int argc,
     char *argv[])
{
  if (argc < 2)
  {
    cout << "Usage: feasopt_c++ filename" << endl;
    return 1;
  }

  GRBEnv* env = 0;
  GRBConstr* c = 0;
  try
  {
    env = new GRBEnv();
    GRBModel feasmodel = GRBModel(*env, argv[1]);

    // clear objective
    feasmodel.setObjective(GRBLinExpr(0.0));

    // add slack variables
    c = feasmodel.getConstrs();
    for (int i = 0; i < feasmodel.get(GRB_IntAttr_NumConstrs); ++i)
    {
      char sense = c[i].get(GRB_CharAttr_Sense);
      if (sense != '>')
      {
        double coef = -1.0;
        feasmodel.addVar(0.0, GRB_INFINITY, 1.0, GRB_CONTINUOUS, 1,
                         &c[i], &coef, "ArtN_" +
                         c[i].get(GRB_StringAttr_ConstrName));
      }
      if (sense != '<')
      {
        double coef = 1.0;
        feasmodel.addVar(0.0, GRB_INFINITY, 1.0, GRB_CONTINUOUS, 1,
                         &c[i], &coef, "ArtP_" +
                         c[i].get(GRB_StringAttr_ConstrName));
      }
    }
    feasmodel.update();
    // optimize modified model
    feasmodel.write("feasopt.lp");
    feasmodel.optimize();

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Error during optimization" << endl;
  }

  delete[] c;
  delete env;
  return 0;
}
```

页面：fixanddive_c++.cpp

fixanddive_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Implement a simple MIP heuristic.  Relax the model,
   sort variables based on fractionality, and fix the 25% of
   the fractional variables that are closest to integer variables.
   Repeat until either the relaxation is integer feasible or
   linearly infeasible. */

#include "gurobi_c++.h"
#include <algorithm>
#include <cmath>
#include <deque>
using namespace std;

bool vcomp(GRBVar*, GRBVar*) throw(GRBException);

int
main(int argc,
     char *argv[])
{
  if (argc < 2)
  {
    cout << "Usage: fixanddive_c++ filename" << endl;
    return 1;
  }

  GRBEnv* env = 0;
  GRBVar* x = 0;
  try
  {
    // Read model
    env = new GRBEnv();
    GRBModel model = GRBModel(*env, argv[1]);

    // Collect integer variables and relax them
    // Note that we use GRBVar* to copy variables
    deque<GRBVar*> intvars;
    x = model.getVars();
    for (int j = 0; j < model.get(GRB_IntAttr_NumVars); ++j)
    {
      if (x[j].get(GRB_CharAttr_VType) != GRB_CONTINUOUS)
      {
        intvars.push_back(&x[j]);
        x[j].set(GRB_CharAttr_VType, GRB_CONTINUOUS);
      }
    }

    model.getEnv().set(GRB_IntParam_OutputFlag, 0);
    model.optimize();

    // Perform multiple iterations. In each iteration, identify the
first
```

```
    // quartile of integer variables that are closest to an integer
value
    // in the relaxation, fix them to the nearest integer, and repeat.

    for (int iter = 0; iter < 1000; ++iter)
    {

      // create a list of fractional variables, sorted in order of
      // increasing distance from the relaxation solution to the
nearest
      // integer value

      deque<GRBVar*> fractional;
      for (size_t j = 0; j < intvars.size(); ++j)
      {
        double sol = fabs(intvars[j]->get(GRB_DoubleAttr_X));
        if (fabs(sol - floor(sol + 0.5)) > 1e-5)
        {
          fractional.push_back(intvars[j]);
        }
      }

      cout << "Iteration " << iter << ", obj " <<
      model.get(GRB_DoubleAttr_ObjVal) << ", fractional " <<
      fractional.size() << endl;

      if (fractional.size() == 0)
      {
        cout << "Found feasible solution - objective " <<
        model.get(GRB_DoubleAttr_ObjVal) << endl;
        break;
      }

      // Fix the first quartile to the nearest integer value
      sort(fractional.begin(), fractional.end(), vcomp);
      int nfix = fractional.size() / 4;
      nfix = (nfix > 1) ? nfix : 1;
      for (int i = 0; i < nfix; ++i)
      {
        GRBVar* v = fractional[i];
        double fixval = floor(v->get(GRB_DoubleAttr_X) + 0.5);
        v->set(GRB_DoubleAttr_LB, fixval);
        v->set(GRB_DoubleAttr_UB, fixval);
        cout << "  Fix " << v->get(GRB_StringAttr_VarName) << " to " <<
        fixval << " ( rel " << v->get(GRB_DoubleAttr_X) << " )" <<
        endl;
      }

      model.optimize();

      // Check optimization result

      if (model.get(GRB_IntAttr_Status) != GRB_OPTIMAL)
      {
        cout << "Relaxation is infeasible" << endl;
        break;
      }
    }

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Error during optimization" << endl;
  }

  delete[] x;
  delete env;
  return 0;
}


bool vcomp(GRBVar* v1,
           GRBVar* v2) throw(GRBException)
{
  double sol1 = fabs(v1->get(GRB_DoubleAttr_X));
  double sol2 = fabs(v2->get(GRB_DoubleAttr_X));
  double frac1 = fabs(sol1 - floor(sol1 + 0.5));
  double frac2 = fabs(sol2 - floor(sol2 + 0.5));
  return (frac1 < frac2);
}
```

页面：lp_c++.cpp

lp_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP model from a file and solves it.
   If the model is infeasible or unbounded, the example turns off
   presolve and solves the model again. If the model is infeasible,
   the example computes an Irreducible Infeasible Subsystem (IIS),
   and writes it to a file */

#include "gurobi_c++.h"
using namespace std;

int
main(int   argc,
     char *argv[])
{
  if (argc < 2) {
    cout << "Usage: lp_c++ filename" << endl;
    return 1;
  }

  try {
    GRBEnv env = GRBEnv();
    GRBModel model = GRBModel(env, argv[1]);

    model.optimize();

    int optimstatus = model.get(GRB_IntAttr_Status);

    if (optimstatus == GRB_INF_OR_UNBD) {
      model.getEnv().set(GRB_IntParam_Presolve, 0);
      model.optimize();
      optimstatus = model.get(GRB_IntAttr_Status);
    }

    if (optimstatus == GRB_OPTIMAL) {
      double objval = model.get(GRB_DoubleAttr_ObjVal);
      cout << "Optimal objective: " << objval << endl;
    } else if (optimstatus == GRB_INFEASIBLE) {
      cout << "Model is infeasible" << endl;

      // compute and write out IIS

      model.computeIIS();
      model.write("model.ilp");
    } else if (optimstatus == GRB_UNBOUNDED) {
      cout << "Model is unbounded" << endl;
    } else {
      cout << "Optimization was stopped with status = "
           << optimstatus << endl;
    }

  } catch(GRBException e) {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  } catch (...) {
    cout << "Error during optimization" << endl;
  }

  return 0;
}
```

页面：lpmethod_c++.cpp

lpmethod_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Solve a model with different values of the Method parameter;
   show which value gives the shortest solve time. */

#include "gurobi_c++.h"
using namespace std;

int
main(int   argc,
     char *argv[])
{

  if (argc < 2)
  {
    cout << "Usage: lpmethod_c++ filename" << endl;
    return 1;
  }

  try {
    // Read model
    GRBEnv env = GRBEnv();
    GRBModel m = GRBModel(env, argv[1]);
    GRBEnv menv = m.getEnv();

    // Solve the model with different values of Method
    int    bestMethod = -1;
    double bestTime = menv.get(GRB_DoubleParam_TimeLimit);
    for (int i = 0; i <= 2; ++i) {
      m.reset();
      menv.set(GRB_IntParam_Method, i);
      m.optimize();
      if (m.get(GRB_IntAttr_Status) == GRB_OPTIMAL) {
        bestTime = m.get(GRB_DoubleAttr_Runtime);
        bestMethod = i;
        // Reduce the TimeLimit parameter to save time
        // with other methods
        menv.set(GRB_DoubleParam_TimeLimit, bestTime);
      }
    }

    // Report which method was fastest
    if (bestMethod == -1) {
      cout << "Unable to solve this model" << endl;
    } else {
      cout << "Solved in " << bestTime
        << " seconds with Method: " << bestMethod << endl;
    }
  } catch(GRBException e) {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  } catch(...) {
    cout << "Exception during optimization" << endl;
  }

  return 0;
}
```

页面：lpmod_c++.cpp

lpmod_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP model from a file and solves it.
   If the model can be solved, then it finds the smallest positive
variable,
   sets its upper bound to zero, and resolves the model two ways:
   first with an advanced start, then without an advanced start
   (i.e. 'from scratch'). */

#include "gurobi_c++.h"
using namespace std;

int
main(int argc,
     char *argv[])
{
  if (argc < 2)
  {
    cout << "Usage: lpmod_c++ filename" << endl;
    return 1;
  }

  GRBEnv* env = 0;
  GRBVar* v = 0;
  try
  {
    // Read model and determine whether it is an LP
    env = new GRBEnv();
    GRBModel model = GRBModel(*env, argv[1]);
    if (model.get(GRB_IntAttr_IsMIP) != 0)
    {
      cout << "The model is not a linear program" << endl;
      return 1;
    }

    model.optimize();

    int status = model.get(GRB_IntAttr_Status);

    if ((status == GRB_INF_OR_UNBD) || (status == GRB_INFEASIBLE) ||
        (status == GRB_UNBOUNDED))
    {
      cout << "The model cannot be solved because it is "
      << "infeasible or unbounded" << endl;
      return 1;
    }

    if (status != GRB_OPTIMAL)
    {
      cout << "Optimization was stopped with status " << status << endl;
      return 0;
    }
```

}

```
    // Find the smallest variable value
    double minVal = GRB_INFINITY;
    int minVar = 0;
    v = model.getVars();
    for (int j = 0; j < model.get(GRB_IntAttr_NumVars); ++j)
    {
      double sol = v[j].get(GRB_DoubleAttr_X);
      if ((sol > 0.0001) && (sol < minVal) &&
          (v[j].get(GRB_DoubleAttr_LB) == 0.0))
      {
        minVal = sol;
        minVar = j;
      }
    }

    cout << "\n*** Setting " << v[minVar].get(GRB_StringAttr_VarName)
    << " from " << minVal << " to zero ***" << endl << endl;
    v[minVar].set(GRB_DoubleAttr_UB, 0.0);

    // Solve from this starting point
    model.optimize();

    // Save iteration & time info
    double warmCount = model.get(GRB_DoubleAttr_IterCount);
    double warmTime = model.get(GRB_DoubleAttr_Runtime);

    // Reset the model and resolve
    cout << "\n*** Resetting and solving "
    << "without an advanced start ***\n" << endl;
    model.reset();
    model.optimize();

    // Save iteration & time info
    double coldCount = model.get(GRB_DoubleAttr_IterCount);
    double coldTime = model.get(GRB_DoubleAttr_Runtime);

    cout << "\n*** Warm start: " << warmCount << " iterations, " <<
    warmTime << " seconds" << endl;
    cout << "*** Cold start: " << coldCount << " iterations, " <<
    coldTime << " seconds" << endl;

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Error during optimization" << endl;
  }

  delete[] v;
  delete env;
  return 0;
}
```

页面：mip1_c++.cpp

mip1_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example formulates and solves the following simple MIP model:

     maximize    x +   y + 2 z
     subject to  x + 2 y + 3 z <= 4
                 x +   y       >= 1
     x, y, z binary
*/

#include "gurobi_c++.h"
using namespace std;

int
main(int   argc,
     char *argv[])
{
  try {
    GRBEnv env = GRBEnv();

    GRBModel model = GRBModel(env);

    // Create variables

    GRBVar x = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "x");
    GRBVar y = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "y");
    GRBVar z = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "z");

    // Integrate new variables

    model.update();

    // Set objective: maximize x + y + 2 z

    model.setObjective(x + y + 2 * z, GRB_MAXIMIZE);

    // Add constraint: x + 2 y + 3 z <= 4

    model.addConstr(x + 2 * y + 3 * z <= 4, "c0");

    // Add constraint: x + y >= 1

    model.addConstr(x + y >= 1, "c1");

    // Optimize model

    model.optimize();

    cout << x.get(GRB_StringAttr_VarName) << " "
         << x.get(GRB_DoubleAttr_X) << endl;
    cout << y.get(GRB_StringAttr_VarName) << " "
         << y.get(GRB_DoubleAttr_X) << endl;
    cout << z.get(GRB_StringAttr_VarName) << " "
         << z.get(GRB_DoubleAttr_X) << endl;

    cout << "Obj: " << model.get(GRB_DoubleAttr_ObjVal) << endl;

  } catch(GRBException e) {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  } catch(...) {
    cout << "Exception during optimization" << endl;
  }

  return 0;
}
```

页面：mip2_c++.cpp

mip2_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads a MIP model from a file, solves it and
   prints the objective values from all feasible solutions
   generated while solving the MIP. Then it creates the fixed
   model and solves that model. */

#include "gurobi_c++.h"
#include <cmath>
using namespace std;

int
main(int   argc,
     char *argv[])
{
  if (argc < 2) {
    cout << "Usage: mip2_c++ filename" << endl;
    return 1;
  }

  GRBEnv *env = 0;
  GRBVar *vars = 0, *fvars = 0;
  try {
    env = new GRBEnv();
    GRBModel model = GRBModel(*env, argv[1]);

    if (model.get(GRB_IntAttr_IsMIP) == 0) {
      throw GRBException("Model is not a MIP");
    }

    model.optimize();

    int optimstatus = model.get(GRB_IntAttr_Status);

    cout << "Optimization complete" << endl;
    double objval = 0;
    if (optimstatus == GRB_OPTIMAL) {
      objval = model.get(GRB_DoubleAttr_ObjVal);
      cout << "Optimal objective: " << objval << endl;
    } else if (optimstatus == GRB_INF_OR_UNBD) {
      cout << "Model is infeasible or unbounded" << endl;
      return 0;
    } else if (optimstatus == GRB_INFEASIBLE) {
      cout << "Model is infeasible" << endl;
      return 0;
    } else if (optimstatus == GRB_UNBOUNDED) {
      cout << "Model is unbounded" << endl;
      return 0;
    } else {
      cout << "Optimization was stopped with status = "
           << optimstatus << endl;
      return 0;
    }

    /* Iterate over the solutions and compute the objectives */

    int numvars = model.get(GRB_IntAttr_NumVars);
    vars = model.getVars();
    model.getEnv().set(GRB_IntParam_OutputFlag, 0);

    cout << endl;
    for ( int k = 0; k < model.get(GRB_IntAttr_SolCount); ++k ) {
      model.getEnv().set(GRB_IntParam_SolutionNumber, k);
      double objn = 0.0;

      for (int j = 0; j < numvars; j++) {
        GRBVar v = vars[j];
        objn += v.get(GRB_DoubleAttr_Obj) * v.get(GRB_DoubleAttr_Xn);
      }

      cout << "Solution " << k << " has objective: " << objn << endl;
    }
    cout << endl;
    model.getEnv().set(GRB_IntParam_OutputFlag, 1);

    /* Create a fixed model, turn off presolve and solve */

    GRBModel fixed = model.fixedModel();

    fixed.getEnv().set(GRB_IntParam_Presolve, 0);

    fixed.optimize();

    int foptimstatus = fixed.get(GRB_IntAttr_Status);

    if (foptimstatus != GRB_OPTIMAL) {
      cerr << "Error: fixed model isn't optimal" << endl;
      return 0;
    }

    double fobjval = fixed.get(GRB_DoubleAttr_ObjVal);

    if (fabs(fobjval - objval) > 1.0e-6 * (1.0 + fabs(objval))) {
      cerr << "Error: objective values are different" << endl;
      return 0;
    }

    /* Print values of nonzero variables */
    fvars = fixed.getVars();
    for (int j = 0; j < numvars; j++) {
      GRBVar v = fvars[j];
      if (v.get(GRB_DoubleAttr_X) != 0.0) {
        cout << v.get(GRB_StringAttr_VarName) << " "
             << v.get(GRB_DoubleAttr_X) << endl;
      }
    }

  } catch(GRBException e) {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  } catch (...) {
    cout << "Error during optimization" << endl;
  }

  delete[] fvars;
  delete[] vars;
  delete env;
  return 0;
}
```

页面：params_c++.cpp

params_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Use parameters that are associated with a model.

   A MIP is solved for 5 seconds with different sets of parameters.
   The one with the smallest MIP gap is selected, and the optimization
   is resumed until the optimal solution is found.
*/

#include "gurobi_c++.h"
#include <cmath>
using namespace std;

double gap(GRBModel *model) throw(GRBException);

int
main(int argc,
     char *argv[])
{
  if (argc < 2)
  {
    cout << "Usage: params_c++ filename" << endl;
    return 1;
  }

  GRBEnv* env = 0;
  GRBModel *bestModel = 0, *m = 0;
  try
  {
    // Read model and verify that it is a MIP
    env = new GRBEnv();
    GRBModel base = GRBModel(*env, argv[1]);
    if (base.get(GRB_IntAttr_IsMIP) == 0)
    {
      cout << "The model is not an integer program" << endl;
      return 1;
    }

    // Set a 5 second time limit
    base.getEnv().set(GRB_DoubleParam_TimeLimit, 5);

    // Now solve the model with different values of MIPFocus,
    // using a pointer to save the best model
    double bestGap = GRB_INFINITY;
    for (int i = 0; i <= 3; ++i)
    {
      m = new GRBModel(base);
      m->getEnv().set(GRB_IntParam_MIPFocus, i);
      m->optimize();
      if (bestModel == 0 || bestGap > gap(m))
      {
        delete bestModel;
        bestModel = m;
        m = 0;
        bestGap = gap(bestModel);
      }
      else
      {
        delete m;
        m = 0;
      }
    }

    // Finally, reset the time limit and continue to solve the
    // best model to optimality
    bestModel->getEnv().set(GRB_DoubleParam_TimeLimit, GRB_INFINITY);
    bestModel->optimize();
    cout << "Solved with MIPFocus: " <<
    bestModel->getEnv().get(GRB_IntParam_MIPFocus) << endl;

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Error during optimization" << endl;
  }

  delete bestModel;
  delete m;
  delete env;
  return 0;
}

// Simple function to determine the MIP gap
double gap(GRBModel *model) throw(GRBException)
{
  if ((model->get(GRB_IntAttr_SolCount) == 0) ||
      (fabs(model->get(GRB_DoubleAttr_ObjVal)) < 1e-6))
  {
    return GRB_INFINITY;
  }
  return fabs(model->get(GRB_DoubleAttr_ObjBound) -
              model->get(GRB_DoubleAttr_ObjVal)) /
         fabs(model->get(GRB_DoubleAttr_ObjVal));
}
```

页面：sensitivity_c++.cpp

sensitivity_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Simple MIP sensitivity analysis example.
   For each integer variable, fix it to its lower and upper bound
   and check the impact on the objective. */

#include "gurobi_c++.h"
using namespace std;

int
main(int argc,
     char *argv[])
{
  if (argc < 2)
  {
    cout << "Usage: sensitivity_c++ filename" << endl;
    return 1;
  }

  GRBEnv* env = 0;
  GRBVar* avars = 0;
  GRBVar* bv = 0;
  try
  {
    // Read model
    env = new GRBEnv();
    GRBModel a = GRBModel(*env, argv[1]);
    a.optimize();
    a.getEnv().set(GRB_IntParam_OutputFlag, 0);

    // Extract variables from model
    avars = a.getVars();

    for (int i = 0; i < a.get(GRB_IntAttr_NumVars); ++i)
    {
      GRBVar v = avars[i];
      if (v.get(GRB_CharAttr_VType) == GRB_BINARY)
      {

        // Create clone and fix variable
        GRBModel b = GRBModel(a);
        bv = b.getVars();
        if (v.get(GRB_DoubleAttr_X) - v.get(GRB_DoubleAttr_LB) < 0.5)
        {
          bv[i].set(GRB_DoubleAttr_LB, bv[i].get(GRB_DoubleAttr_UB));
        }
        else
        {
          bv[i].set(GRB_DoubleAttr_UB, bv[i].get(GRB_DoubleAttr_LB));
        }
        delete[] bv;
        bv = 0;
        b.optimize();

        if (b.get(GRB_IntAttr_Status) == GRB_OPTIMAL)
        {
          double objchg =
            b.get(GRB_DoubleAttr_ObjVal)  - a.get(GRB_DoubleAttr_ObjVal);
          if (objchg < 0)
          {
            objchg = 0;
          }
          cout << "Objective sensitivity for variable " <<
          v.get(GRB_StringAttr_VarName) << " is " << objchg << endl;
        }
        else
        {
          cout << "Objective sensitivity for variable " <<
          v.get(GRB_StringAttr_VarName) << " is infinite" << endl;
        }
      }
    }

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Error during optimization" << endl;
  }

  delete[] avars;
  delete[] bv;
  delete env;
  return 0;
}
```

页面：qp1_c++.cpp

qp1_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example formulates and solves the following simple QP model:

     minimize    x^2 + x*y + y^2 + y*z + z^2
     subject to  x + 2 y + 3 z <= 4
                 x +   y       >= 1

   It solves it once as a continuous model, and once as an integer
model.
*/

#include "gurobi_c++.h"
using namespace std;

int
main(int   argc,
     char *argv[])
{
  try {
    GRBEnv env = GRBEnv();

    GRBModel model = GRBModel(env);

    // Create variables

    GRBVar x = model.addVar(0.0, 1.0, 0.0, GRB_CONTINUOUS, "x");
    GRBVar y = model.addVar(0.0, 1.0, 0.0, GRB_CONTINUOUS, "y");
    GRBVar z = model.addVar(0.0, 1.0, 0.0, GRB_CONTINUOUS, "z");

    // Integrate new variables

    model.update();

    // Set objective

    GRBQuadExpr obj = x*x + x*y + y*y + y*z + z*z;
    model.setObjective(obj);

    // Add constraint: x + 2 y + 3 z <= 4

    model.addConstr(x + 2 * y + 3 * z <= 4, "c0");

    // Add constraint: x + y >= 1

    model.addConstr(x + y >= 1, "c1");

    // Optimize model

    model.optimize();

    cout << x.get(GRB_StringAttr_VarName) << " "
```

```
         << x.get(GRB_DoubleAttr_X) << endl;
    cout << y.get(GRB_StringAttr_VarName) << " "
         << y.get(GRB_DoubleAttr_X) << endl;
    cout << z.get(GRB_StringAttr_VarName) << " "
         << z.get(GRB_DoubleAttr_X) << endl;

    cout << "Obj: " << model.get(GRB_DoubleAttr_ObjVal) << endl;

    // Change variable types to integer

    x.set(GRB_CharAttr_VType, GRB_INTEGER);
    y.set(GRB_CharAttr_VType, GRB_INTEGER);
    z.set(GRB_CharAttr_VType, GRB_INTEGER);

    // Optimize model

    model.optimize();

    cout << x.get(GRB_StringAttr_VarName) << " "
         << x.get(GRB_DoubleAttr_X) << endl;
    cout << y.get(GRB_StringAttr_VarName) << " "
         << y.get(GRB_DoubleAttr_X) << endl;
    cout << z.get(GRB_StringAttr_VarName) << " "
         << z.get(GRB_DoubleAttr_X) << endl;

    cout << "Obj: " << model.get(GRB_DoubleAttr_ObjVal) << endl;

  } catch(GRBException e) {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  } catch(...) {
    cout << "Exception during optimization" << endl;
  }

  return 0;
}
```

页面：sos_c++.cpp

sos_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example creates a very simple Special Ordered Set (SOS) model.
   The model consists of 3 continuous variables, no linear constraints,
   and a pair of SOS constraints of type 1. */

#include "gurobi_c++.h"
using namespace std;

int
main(int   argc,
     char *argv[])
{
  GRBEnv *env = 0;
  GRBVar *x = 0;
  try {
    env = new GRBEnv();

    GRBModel model = GRBModel(*env);

    // Create variables

    double ub[]    = {1, 1, 2};
    double obj[]   = {-2, -1, -1};
    string names[] = {"x0", "x1", "x2"};

    x = model.addVars(NULL, ub, obj, NULL, names, 3);

    // Integrate new variables

    model.update();

    // Add first SOS1: x0=0 or x1=0

    GRBVar sosv1[]  = {x[0], x[1]};
    double soswt1[] = {1, 2};

    model.addSOS(sosv1, soswt1, 2, GRB_SOS_TYPE1);

    // Add second SOS1: x0=0 or x2=0 */

    GRBVar sosv2[]  = {x[0], x[2]};
    double soswt2[] = {1, 2};

    model.addSOS(sosv2, soswt2, 2, GRB_SOS_TYPE1);

    // Optimize model

    model.optimize();

    for (int i = 0; i < 3; i++)
      cout << x[i].get(GRB_StringAttr_VarName) << " "
```

```
           << x[i].get(GRB_DoubleAttr_X) << endl;

    cout << "Obj: " << model.get(GRB_DoubleAttr_ObjVal) << endl;

  } catch(GRBException e) {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  } catch(...) {
    cout << "Exception during optimization" << endl;
  }

  delete[] x;
  delete env;
  return 0;
}
```

页面：sudoku_c++.cpp

Sudoku_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */
/*
  Sudoku example.

  The Sudoku board is a 9x9 grid, which is further divided into a 3x3
grid
  of 3x3 grids.  Each cell in the grid must take a value from 0 to 9.
  No two grid cells in the same row, column, or 3x3 subgrid may take
the
  same value.

  In the MIP formulation, binary variables x[i,j,v] indicate whether
  cell <i,j> takes value 'v'.  The constraints are as follows:
    1. Each cell must take exactly one value (sum_v x[i,j,v] = 1)
    2. Each value is used exactly once per row (sum_i x[i,j,v] = 1)
    3. Each value is used exactly once per column (sum_j x[i,j,v] = 1)
    4. Each value is used exactly once per 3x3 subgrid (sum_grid
x[i,j,v] = 1)
*/

#include "gurobi_c++.h"
#include <sstream>
using namespace std;

#define sd 3
#define n (sd*sd)

string itos(int i) {stringstream s; s << i; return s.str(); }

int
main(int   argc,
     char *argv[])
{
  try {
    GRBEnv env = GRBEnv();
    GRBModel model = GRBModel(env);

    GRBVar vars[n][n][n];

    // Create 3-D array of model variables

    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        for (int v = 0; v < n; v++) {
          string s = "G_" + itos(i) + "_" + itos(j) + "_" + itos(v);
          vars[i][j][v] = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, s);
        }
      }
    }

    // Integrate variables into model
```

```
model.update();

// Add constraints

// Each cell must take one value

for (int i = 0; i < n; i++) {
  for (int j = 0; j < n; j++) {
    GRBLinExpr expr = 0;
    for (int v = 0; v < n; v++)
      expr += vars[i][j][v];
    string s = "V_" + itos(i) + "_" + itos(j);
    model.addConstr(expr, GRB_EQUAL, 1.0, s);
  }
}

// Each value appears once per row

for (int i = 0; i < n; i++) {
  for (int v = 0; v < n; v++) {
    GRBLinExpr expr = 0;
    for (int j = 0; j < n; j++)
      expr += vars[i][j][v];
    string s = "R_" + itos(i) + "_" + itos(v);
    model.addConstr(expr == 1.0, s);
  }
}

// Each value appears once per column

for (int j = 0; j < n; j++) {
  for (int v = 0; v < n; v++) {
    GRBLinExpr expr = 0;
    for (int i = 0; i < n; i++)
      expr += vars[i][j][v];
    string s = "C_" + itos(j) + "_" + itos(v);
    model.addConstr(expr == 1.0, s);
  }
}

// Each value appears once per sub-grid

for (int v = 0; v < n; v++) {
  for (int i0 = 0; i0 < sd; i0++) {
    for (int j0 = 0; j0 < sd; j0++) {
      GRBLinExpr expr = 0;
      for (int i1 = 0; i1 < sd; i1++) {
        for (int j1 = 0; j1 < sd; j1++) {
          expr += vars[i0*sd+i1][j0*sd+j1][v];
        }
      }

      string s = "Sub_" + itos(v) + "_" + itos(i0) + "_" + itos(j0);
      model.addConstr(expr == 1.0, s);
    }
  }
}
```

```
    // Fix variables associated with pre-specified cells

    char input[10];
    for (int i = 0; i < n; i++) {
      cin >> input;
      for (int j = 0; j < n; j++) {
        int val = (int) input[j] - 48 - 1; // 0-based

        if (val >= 0)
          vars[i][j][val].set(GRB_DoubleAttr_LB, 1.0);
      }
    }

    // Optimize model

    model.optimize();

    // Write model to file

    model.write("sudoku.lp");

    cout << endl;
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        for (int v = 0; v < n; v++) {
          if (vars[i][j][v].get(GRB_DoubleAttr_X) > 0.5)
            cout << v+1;
        }
      }
      cout << endl;
    }
    cout << endl;
  } catch(GRBException e) {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  } catch (...) {
    cout << "Error during optimization" << endl;
  }

  return 0;
}
```

页面：workforce1_c++.cpp

workforce1_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, use IIS to find a
set of
   conflicting constraints. Note that there may be additional conflicts
   besides what is reported via IIS. */

#include "gurobi_c++.h"
#include <sstream>
using namespace std;

int
main(int argc,
     char *argv[])
{
  GRBEnv* env = 0;
  GRBConstr* c = 0;
  GRBVar** x = 0;
  int xCt = 0;
  try
  {

    // Sample data
    const int nShifts = 14;
    const int nWorkers = 7;

    // Sets of days and workers
    string Shifts[] =
      { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
        "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12", "Sat13",
        "Sun14" };
    string Workers[] =
      { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred", "Gu" };

    // Number of workers required for each shift
    double shiftRequirements[] =
      { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

    // Amount each worker is paid to work one shift
    double pay[] = { 10, 12, 10, 8, 8, 9, 11 };

    // Worker availability: 0 if the worker is unavailable for a shift
    double availability[][nShifts] =
      { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
        { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
        { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
        { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };
```

```
// Model
env = new GRBEnv();
GRBModel model = GRBModel(*env);
model.set(GRB_StringAttr_ModelName, "assignment");

// Assignment variables: x[w][s] == 1 if worker w is assigned
// to shift s. Since an assignment model always produces integer
// solutions, we use continuous variables and solve as an LP.
x = new GRBVar*[nWorkers];
for (int w = 0; w < nWorkers; ++w)
{
  x[w] = model.addVars(nShifts);
  xCt++;
  model.update();
  for (int s = 0; s < nShifts; ++s)
  {
    ostringstream vname;
    vname << Workers[w] << "." << Shifts[s];
    x[w][s].set(GRB_DoubleAttr_UB, availability[w][s]);
    x[w][s].set(GRB_DoubleAttr_Obj, pay[w]);
    x[w][s].set(GRB_StringAttr_VarName, vname.str());
  }
}

// The objective is to minimize the total pay costs
model.set(GRB_IntAttr_ModelSense, 1);

// Update model to integrate new variables
model.update();

// Constraint: assign exactly shiftRequirements[s] workers
// to each shift s
for (int s = 0; s < nShifts; ++s)
{
  GRBLinExpr lhs = 0;
  for (int w = 0; w < nWorkers; ++w)
  {
    lhs += x[w][s];
  }
  model.addConstr(lhs == shiftRequirements[s], Shifts[s]);
}

// Optimize
model.optimize();
int status = model.get(GRB_IntAttr_Status);
if (status == GRB_UNBOUNDED)
{
  cout << "The model cannot be solved "
  << "because it is unbounded" << endl;
  return 1;
}
if (status == GRB_OPTIMAL)
{
  cout << "The optimal objective is " <<
  model.get(GRB_DoubleAttr_ObjVal) << endl;
  return 0;
```

```
    }
    if ((status != GRB_INF_OR_UNBD) && (status != GRB_INFEASIBLE))
    {
      cout << "Optimization was stopped with status " << status << endl;
      return 1;
    }

    // do IIS
    cout << "The model is infeasible; computing IIS" << endl;
    model.computeIIS();
    cout << "\nThe following constraint(s) "
    << "cannot be satisfied:" << endl;
    c = model.getConstrs();
    for (int i = 0; i < model.get(GRB_IntAttr_NumConstrs); ++i)
    {
      if (c[i].get(GRB_IntAttr_IISConstr) == 1)
      {
        cout << c[i].get(GRB_StringAttr_ConstrName) << endl;
      }
    }

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Exception during optimization" << endl;
  }

  delete[] c;
  for (int i = 0; i < xCt; ++i) {
    delete[] x[i];
  }
  delete[] x;
  delete env;
  return 0;
}
```

页面：workforce2_c++.cpp

workforce2_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, use IIS iteratively
to
   find all conflicting constraints. */

#include "gurobi_c++.h"
#include <sstream>
#include <deque>
using namespace std;

int
main(int argc,
     char *argv[])
{
  GRBEnv* env = 0;
  GRBConstr* c = 0;
  GRBVar** x = 0;
  int xCt = 0;
  try
  {

    // Sample data
    const int nShifts = 14;
    const int nWorkers = 7;

    // Sets of days and workers
    string Shifts[] =
      { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
        "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12", "Sat13",
        "Sun14" };
    string Workers[] =
      { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred", "Gu" };

    // Number of workers required for each shift
    double shiftRequirements[] =
      { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

    // Amount each worker is paid to work one shift
    double pay[] = { 10, 12, 10, 8, 8, 9, 11 };

    // Worker availability: 0 if the worker is unavailable for a shift
    double availability[][nShifts] =
      { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
        { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
        { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
        { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };
```

```
// Model
env = new GRBEnv();
GRBModel model = GRBModel(*env);
model.set(GRB_StringAttr_ModelName, "assignment");

// Assignment variables: x[w][s] == 1 if worker w is assigned
// to shift s. Since an assignment model always produces integer
// solutions, we use continuous variables and solve as an LP.
x = new GRBVar*[nWorkers];
for (int w = 0; w < nWorkers; ++w)
{
  x[w] = model.addVars(nShifts);
  xCt++;
  model.update();
  for (int s = 0; s < nShifts; ++s)
  {
    ostringstream vname;
    vname << Workers[w] << "." << Shifts[s];
    x[w][s].set(GRB_DoubleAttr_UB, availability[w][s]);
    x[w][s].set(GRB_DoubleAttr_Obj, pay[w]);
    x[w][s].set(GRB_StringAttr_VarName, vname.str());
  }
}

// The objective is to minimize the total pay costs
model.set(GRB_IntAttr_ModelSense, 1);

// Update model to integrate new variables
model.update();

// Constraint: assign exactly shiftRequirements[s] workers
// to each shift s
for (int s = 0; s < nShifts; ++s)
{
  GRBLinExpr lhs = 0;
  for (int w = 0; w < nWorkers; ++w)
  {
    lhs += x[w][s];
  }
  model.addConstr(lhs == shiftRequirements[s], Shifts[s]);
}

// Optimize
model.optimize();
int status = model.get(GRB_IntAttr_Status);
if (status == GRB_UNBOUNDED)
{
  cout << "The model cannot be solved "
  << "because it is unbounded" << endl;
  return 1;
}
if (status == GRB_OPTIMAL)
{
  cout << "The optimal objective is " <<
  model.get(GRB_DoubleAttr_ObjVal) << endl;
  return 0;
```

```
    }
    if ((status != GRB_INF_OR_UNBD) && (status != GRB_INFEASIBLE))
    {
      cout << "Optimization was stopped with status " << status << endl;
      return 1;
    }

    // do IIS
    cout << "The model is infeasible; computing IIS" << endl;
    deque<string> removed;

    // Loop until we reduce to a model that can be solved
    while (1)
    {
      model.computeIIS();
      cout << "\nThe following constraint cannot be satisfied:" << endl;
      c = model.getConstrs();
      for (int i = 0; i < model.get(GRB_IntAttr_NumConstrs); ++i)
      {
        if (c[i].get(GRB_IntAttr_IISConstr) == 1)
        {
          cout << c[i].get(GRB_StringAttr_ConstrName) << endl;
          // Remove a single constraint from the model
          removed.push_back(c[i].get(GRB_StringAttr_ConstrName));
          model.remove(c[i]);
          break;
        }
      }
      delete[] c;
      c = 0;

      cout << endl;
      model.optimize();
      status = model.get(GRB_IntAttr_Status);

      if (status == GRB_UNBOUNDED)
      {
        cout << "The model cannot be solved because it is unbounded" <<
endl;
        return 0;
      }
      if (status == GRB_OPTIMAL)
      {
        break;
      }
      if ((status != GRB_INF_OR_UNBD) && (status != GRB_INFEASIBLE))
      {
        cout << "Optimization was stopped with status " << status <<
endl;
        return 1;
      }
    }
    cout << "\nThe following constraints were removed "
    << "to get a feasible LP:" << endl;

    for (deque<string>::iterator s = removed.begin();
         s != removed.end();
         ++s)
    {
      cout << *s << " ";
    }
    cout << endl;

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Exception during optimization" << endl;
  }

  delete[] c;
  for (int i = 0; i < xCt; ++i) {
    delete[] x[i];
  }
  delete[] x;
  delete env;
  return 0;
}
```

页面：workforce3_c++.cpp

workforce3_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, add slack variables
   to determine which constraints cannot be satisfied, and how much
   they need to be relaxed. */

#include "gurobi_c++.h"
#include <sstream>
#include <deque>
using namespace std;

int
main(int argc,
     char *argv[])
{
  GRBEnv* env = 0;
  GRBConstr* c = 0;
  GRBVar** x = 0;
  int xCt = 0;
  try
  {

    // Sample data
    const int nShifts = 14;
    const int nWorkers = 7;

    // Sets of days and workers
    string Shifts[] =
      { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
        "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12", "Sat13",
        "Sun14" };
    string Workers[] =
      { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred", "Gu" };

    // Number of workers required for each shift
    double shiftRequirements[] =
      { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

    // Amount each worker is paid to work one shift
    double pay[] = { 10, 12, 10, 8, 8, 9, 11 };

    // Worker availability: 0 if the worker is unavailable for a shift
    double availability[][nShifts] =
      { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
        { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
        { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
        { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };
```

```
// Model
env = new GRBEnv();
GRBModel model = GRBModel(*env);
model.set(GRB_StringAttr_ModelName, "assignment");

// Assignment variables: x[w][s] == 1 if worker w is assigned
// to shift s. Since an assignment model always produces integer
// solutions, we use continuous variables and solve as an LP.
x = new GRBVar*[nWorkers];
for (int w = 0; w < nWorkers; ++w)
{
  x[w] = model.addVars(nShifts);
  xCt++;
  model.update();
  for (int s = 0; s < nShifts; ++s)
  {
    ostringstream vname;
    vname << Workers[w] << "." << Shifts[s];
    x[w][s].set(GRB_DoubleAttr_UB, availability[w][s]);
    x[w][s].set(GRB_DoubleAttr_Obj, pay[w]);
    x[w][s].set(GRB_StringAttr_VarName, vname.str());
  }
}

// The objective is to minimize the total pay costs
model.set(GRB_IntAttr_ModelSense, 1);

// Update model to integrate new variables
model.update();

// Constraint: assign exactly shiftRequirements[s] workers
// to each shift s
for (int s = 0; s < nShifts; ++s)
{
  GRBLinExpr lhs = 0;
  for (int w = 0; w < nWorkers; ++w)
  {
    lhs += x[w][s];
  }
  model.addConstr(lhs == shiftRequirements[s], Shifts[s]);
}

// Optimize
model.optimize();
int status = model.get(GRB_IntAttr_Status);
if (status == GRB_UNBOUNDED)
{
  cout << "The model cannot be solved "
  << "because it is unbounded" << endl;
  return 1;
}
if (status == GRB_OPTIMAL)
{
  cout << "The optimal objective is " <<
  model.get(GRB_DoubleAttr_ObjVal) << endl;
  return 0;
```

```
    }
    if ((status != GRB_INF_OR_UNBD) && (status != GRB_INFEASIBLE))
    {
      cout << "Optimization was stopped with status " << status << endl;
      return 1;
    }

    // Add slack variables to make the model feasible
    cout << "The model is infeasible; adding slack variables" << endl;

    // Set original objective coefficients to zero
    model.setObjective(GRBLinExpr(0.0));

    // Add a new slack variable to each shift constraint so that the
shifts
    // can be satisfied
    deque<GRBVar> slacks;
    c = model.getConstrs();
    for (int i = 0; i < model.get(GRB_IntAttr_NumConstrs); ++i)
    {
      GRBColumn col;
      col.addTerm(1.0, c[i]);
      GRBVar newvar =
        model.addVar(0, GRB_INFINITY, 1.0, GRB_CONTINUOUS, col,
                     c[i].get(GRB_StringAttr_ConstrName) + "Slack");
      slacks.push_back(newvar);
    }

    // Solve the model with slacks
    model.optimize();
    status = model.get(GRB_IntAttr_Status);

    if ((status == GRB_INF_OR_UNBD) || (status == GRB_INFEASIBLE) ||
        (status == GRB_UNBOUNDED))
    {
      cout << "The model with slacks cannot be solved " <<
      "because it is infeasible or unbounded" << endl;
      return 1;
    }
    if (status != GRB_OPTIMAL)
    {
      cout << "Optimization was stopped with status " << status << endl;
      return 1;
    }

    cout << "\nSlack values:" << endl;
    for (deque<GRBVar>::iterator sv = slacks.begin();
         sv != slacks.end();
         ++sv)
    {
      if (sv->get(GRB_DoubleAttr_X) > 1e-6)
      {
        cout << sv->get(GRB_StringAttr_VarName) << " = " <<
        sv->get(GRB_DoubleAttr_X) << endl;
      }
    }
```

```
  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Exception during optimization" << endl;
  }

  delete[] c;
  for (int i = 0; i < xCt; ++i) {
    delete[] x[i];
  }
  delete[] x;
  delete env;
  return 0;
}
```

页面：workforce4_c++.cpp

workforce4_c++.cpp

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. We use Pareto optimization to solve the model:
   first, we minimize the linear sum of the slacks. Then, we constrain
   the sum of the slacks, and we minimize a quadratic objective that
   tries to balance the workload among the workers. */

#include "gurobi_c++.h"
#include <sstream>
using namespace std;

int solveAndPrint(GRBModel& model, GRBVar& totSlack,
                  int nWorkers, string* Workers,
                  GRBVar* totShifts) throw(GRBException);

int
main(int argc,
     char *argv[])
{
  GRBEnv* env = 0;
  GRBConstr* c = 0;
  GRBVar* v = 0;
  GRBVar** x = 0;
  GRBVar* slacks = 0;
  GRBVar* totShifts = 0;
  GRBVar* diffShifts = 0;
  int xCt = 0;
  try
  {

    // Sample data
    const int nShifts = 14;
    const int nWorkers = 7;

    // Sets of days and workers
    string Shifts[] =
      { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
        "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12", "Sat13",
        "Sun14" };
    string Workers[] =
      { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred", "Gu" };

    // Number of workers required for each shift
    double shiftRequirements[] =
      { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

    // Worker availability: 0 if the worker is unavailable for a shift
    double availability[][nShifts] =
      { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
        { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
        { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
        { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

    // Model
    env = new GRBEnv();
    GRBModel model = GRBModel(*env);
    model.set(GRB_StringAttr_ModelName, "assignment");

    // Assignment variables: x[w][s] == 1 if worker w is assigned
    // to shift s. This is no longer a pure assignment model, so we
must
    // use binary variables.
    x = new GRBVar*[nWorkers];
    for (int w = 0; w < nWorkers; ++w)
    {
      x[w] = model.addVars(nShifts);
      xCt++;
      model.update();
      for (int s = 0; s < nShifts; ++s)
      {
        ostringstream vname;
        vname << Workers[w] << "." << Shifts[s];
        x[w][s].set(GRB_DoubleAttr_UB, availability[w][s]);
        x[w][s].set(GRB_CharAttr_VType, GRB_BINARY);
        x[w][s].set(GRB_StringAttr_VarName, vname.str());
      }
    }

    // Slack variables for each shift constraint so that the shifts can
    // be satisfied
    slacks = model.addVars(nShifts);
    model.update();
    for (int s = 0; s < nShifts; ++s)
    {
      ostringstream vname;
      vname << Shifts[s] << "Slack";
      slacks[s].set(GRB_StringAttr_VarName, vname.str());
    }

    // Variable to represent the total slack
    GRBVar totSlack = model.addVar(0, GRB_INFINITY, 0, GRB_CONTINUOUS,
                                   "totSlack");

    // Variables to count the total shifts worked by each worker
    totShifts = model.addVars(nWorkers);
    model.update();
    for (int w = 0; w < nWorkers; ++w)
    {
      ostringstream vname;
      vname << Workers[w] << "TotShifts";
      totShifts[w].set(GRB_StringAttr_VarName, vname.str());
    }

    // Update model to integrate new variables
    model.update();

    GRBLinExpr lhs;

    // Constraint: assign exactly shiftRequirements[s] workers
    // to each shift s
    for (int s = 0; s < nShifts; ++s)
    {
      lhs = 0;
      lhs += slacks[s];
      for (int w = 0; w < nWorkers; ++w)
      {
        lhs += x[w][s];
      }
      model.addConstr(lhs == shiftRequirements[s], Shifts[s]);
    }

    // Constraint: set totSlack equal to the total slack
    lhs = 0;
    for (int s = 0; s < nShifts; ++s)
    {
      lhs += slacks[s];
    }
    model.addConstr(lhs == totSlack, "totSlack");

    // Constraint: compute the total number of shifts for each worker
    for (int w = 0; w < nWorkers; ++w) {
      lhs = 0;
      for (int s = 0; s < nShifts; ++s) {
        lhs += x[w][s];
      }
      ostringstream vname;
      vname << "totShifts" << Workers[w];
      model.addConstr(lhs == totShifts[w], vname.str());
    }

    // Objective: minimize the total slack
    GRBLinExpr obj = 0;
    obj += totSlack;
    model.setObjective(obj);

    // Optimize
    int status = solveAndPrint(model, totSlack, nWorkers, Workers,
totShifts);
    if (status != GRB_OPTIMAL)
    {
      return 1;
    }

    // Constrain the slack by setting its upper and lower bounds
    totSlack.set(GRB_DoubleAttr_UB, totSlack.get(GRB_DoubleAttr_X));
    totSlack.set(GRB_DoubleAttr_LB, totSlack.get(GRB_DoubleAttr_X));

    // Variable to count the average number of shifts worked
    GRBVar avgShifts =
      model.addVar(0, GRB_INFINITY, 0, GRB_CONTINUOUS, "avgShifts");
    // Variables to count the difference from average for each worker;
    // note that these variables can take negative values.
    diffShifts = model.addVars(nWorkers);
    model.update();
    for (int w = 0; w < nWorkers; ++w) {
      ostringstream vname;
      vname << Workers[w] << "Diff";
      diffShifts[w].set(GRB_StringAttr_VarName, vname.str());
      diffShifts[w].set(GRB_DoubleAttr_LB, -GRB_INFINITY);
    }

    // Update model to integrate new variables
    model.update();

    // Constraint: compute the average number of shifts worked
    lhs = 0;
    for (int w = 0; w < nWorkers; ++w) {
      lhs += totShifts[w];
    }
    model.addConstr(lhs == nWorkers * avgShifts, "avgShifts");

    // Constraint: compute the difference from the average number of
shifts
    for (int w = 0; w < nWorkers; ++w) {
      lhs = 0;
      lhs += totShifts[w];
      lhs -= avgShifts;
      ostringstream vname;
      vname << Workers[w] << "Diff";
      model.addConstr(lhs == diffShifts[w], vname.str());
    }

    // Objective: minimize the sum of the square of the difference from
the
    // average number of shifts worked
    GRBQuadExpr qobj;
    for (int w = 0; w < nWorkers; ++w) {
      qobj += diffShifts[w] * diffShifts[w];
    }
    model.setObjective(qobj);

    // Optimize
    status = solveAndPrint(model, totSlack, nWorkers, Workers,
totShifts);
    if (status != GRB_OPTIMAL)
    {
      return 1;
    }

  }
  catch (GRBException e)
  {
    cout << "Error code = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
  }
  catch (...)
  {
    cout << "Exception during optimization" << endl;
  }

  delete[] c;
  delete[] v;
  for (int i = 0; i < xCt; ++i) {
    delete[] x[i];
  }
  delete[] x;
  delete[] slacks;
  delete[] totShifts;
  delete[] diffShifts;
  delete env;
  return 0;
}

int solveAndPrint(GRBModel& model, GRBVar& totSlack,
                  int nWorkers, string* Workers,
                  GRBVar* totShifts) throw(GRBException)
{
  model.optimize();
  int status = model.get(GRB_IntAttr_Status);

  if ((status == GRB_INF_OR_UNBD) || (status == GRB_INFEASIBLE) ||
      (status == GRB_UNBOUNDED))
  {
    cout << "The model cannot be solved " <<
    "because it is infeasible or unbounded" << endl;
    return status;
  }
  if (status != GRB_OPTIMAL)
  {
    cout << "Optimization was stopped with status " << status << endl;
    return status;
  }

  // Print total slack and the number of shifts worked for each worker
  cout << endl << "Total slack required: " <<
    totSlack.get(GRB_DoubleAttr_X) << endl;
  for (int w = 0; w < nWorkers; ++w) {
    cout << Workers[w] << " worked " <<
    totShifts[w].get(GRB_DoubleAttr_X) << " shifts" << endl;
  }
  cout << endl;
  return status;
}
```

页面：Java Examples

Java 示例

本节包含了所有 Gurobi Java 的示例源代码。相同的源代码也可以在 Gurobi 发布包的 *examples/java* 目录中找到。

小节

页面：Callback.java

Callback.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP or a MIP from a file, sets a callback
   to monitor the optimization progress and to output some progress
   information to the screen and to a log file. If it is a MIP and 10%
   gap is reached, then it aborts */

import gurobi.*;

public class Callback extends GRBCallback {
  private GRBVar[] vars;
  private int      lastmsg;

  public Callback(GRBVar[] xvars) {
    vars = xvars;
    lastmsg = -100;
  }

  protected void callback() {
    try {
      if (where == GRB.CB_MESSAGE) {
        String st = getStringInfo(GRB.CB_MSG_STRING);
        if (st != null) System.out.print(st);
      } else if (where == GRB.CB_PRESOLVE) {
        int cdels = getIntInfo(GRB.CB_PRE_COLDEL);
        int rdels = getIntInfo(GRB.CB_PRE_ROWDEL);
        System.out.println(cdels+" columns and "+rdels+" rows are
removed");
      } else if (where == GRB.CB_SIMPLEX) {
        int itcnt = (int) getDoubleInfo(GRB.CB_SPX_ITRCNT);
        if (itcnt%100 == 0) {
          double obj  = getDoubleInfo(GRB.CB_SPX_OBJVAL);
          double pinf = getDoubleInfo(GRB.CB_SPX_PRIMINF);
          double dinf = getDoubleInfo(GRB.CB_SPX_DUALINF);
          int  ispert = getIntInfo(GRB.CB_SPX_ISPERT);
          char ch;
          if (ispert == 0)      ch = ' ';
          else if (ispert == 1) ch = 'S';
          else                  ch = 'P';
          System.out.println(itcnt+"  "+ obj + ch + "  "+pinf + "  " +
dinf);
        }
      } else if (where == GRB.CB_MIP) {
        int nodecnt = (int) getDoubleInfo(GRB.CB_MIP_NODCNT);
        if (nodecnt - lastmsg >= 100) {
          lastmsg = nodecnt;
          double objbst = getDoubleInfo(GRB.CB_MIP_OBJBST);
          double objbnd = getDoubleInfo(GRB.CB_MIP_OBJBND);
          if (Math.abs(objbst - objbnd) < 0.1 * (1.0 +
Math.abs(objbst)))
            abort();
          int actnodes = (int) getDoubleInfo(GRB.CB_MIP_NODLFT);
          int itcnt    = (int) getDoubleInfo(GRB.CB_MIP_ITRCNT);
          int solcnt   = getIntInfo(GRB.CB_MIP_SOLCNT);
          int cutcnt   = getIntInfo(GRB.CB_MIP_CUTCNT);
          System.out.println(nodecnt + " " +  actnodes + " " +  itcnt +
" "
            +  objbst + " " +  objbnd + " " +  solcnt + " " +  cutcnt);
        }
      } else if (where == GRB.CB_MIPSOL) {
        double obj     = getDoubleInfo(GRB.CB_MIPSOL_OBJ);
        int    nodecnt = (int) getDoubleInfo(GRB.CB_MIPSOL_NODCNT);
        double[] x = getSolution(vars);
        System.out.println("**** New solution at node " + nodecnt + ",
obj "
                           + obj + ", x[0] = " + x[0] + "****");
      }
    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
      e.printStackTrace();
    }
  }

  public static void main(String[] args) {

    if (args.length < 1) {
      System.out.println("Usage: java Callback filename");
      System.exit(1);
    }

    try {
      GRBEnv    env   = new GRBEnv();
      GRBModel  model = new GRBModel(env, args[0]);

      model.getEnv().set(GRB.IntParam.OutputFlag, 0);

      GRBVar[] vars   = model.getVars();

      model.setCallback(new Callback(vars));
      model.optimize();

      double[] x      = model.get(GRB.DoubleAttr.X, vars);
      String[] vnames = model.get(GRB.StringAttr.VarName, vars);

      for (int j = 0; j < vars.length; j++) {
        if (x[j] != 0.0) System.out.println(vnames[j] + " " + x[j]);
      }

      for (int j = 0; j < vars.length; j++) {
        if (x[j] != 0.0) System.out.println(vnames[j] + " " + x[j]);
      }

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
      e.printStackTrace();
    }
  }
}
```

页面：Diet.java

Diet.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Solve the classic diet model, showing how to add constraints
   to an existing model. */

import gurobi.*;

public class Diet {

  public static void main(String[] args) {
    try {

      // Nutrition guidelines, based on
      // USDA Dietary Guidelines for Americans, 2005
      // http://www.health.gov/DietaryGuidelines/dga2005/
      String Categories[] =
          new String[] { "calories", "protein", "fat", "sodium" };
      int nCategories = Categories.length;
      double minNutrition[] = new double[] { 1800, 91, 0, 0 };
      double maxNutrition[] = new double[] { 2200, GRB.INFINITY, 65,
1779 };

      // Set of foods
      String Foods[] =
          new String[] { "hamburger", "chicken", "hot dog", "fries",
              "macaroni", "pizza", "salad", "milk", "ice cream" };
      int nFoods = Foods.length;
      double cost[] =
          new double[] { 2.49, 2.89, 1.50, 1.89, 2.09, 1.99, 2.49, 0.89,
              1.59 };

      // Nutrition values for the foods
      double nutritionValues[][] = new double[][] {
          { 410, 24, 26, 730 },   // hamburger
          { 420, 32, 10, 1190 },  // chicken
          { 560, 20, 32, 1800 },  // hot dog
          { 380, 4, 19, 270 },    // fries
          { 320, 12, 10, 930 },   // macaroni
          { 320, 15, 12, 820 },   // pizza
          { 320, 31, 12, 1230 },  // salad
          { 100, 8, 2.5, 125 },   // milk
          { 330, 8, 10, 180 }     // ice cream
          };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.set(GRB.StringAttr.ModelName, "diet");

      // Create decision variables for the nutrition information,
      // which we limit via bounds
      GRBVar[] nutrition = new GRBVar[nCategories];
      for (int i = 0; i < nCategories; ++i) {
        nutrition[i] =
            model.addVar(minNutrition[i], maxNutrition[i], 0,
GRB.CONTINUOUS,
                         Categories[i]);
      }

      // Create decision variables for the foods to buy
      GRBVar[] buy = new GRBVar[nFoods];
      for (int j = 0; j < nFoods; ++j) {
        buy[j] =
            model.addVar(0, GRB.INFINITY, cost[j], GRB.CONTINUOUS,
Foods[j]);
      }

      // The objective is to minimize the costs
      model.set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.update();

      // Nutrition constraints
      for (int i = 0; i < nCategories; ++i) {
        GRBLinExpr ntot = new GRBLinExpr();
        for (int j = 0; j < nFoods; ++j) {
          ntot.addTerm(nutritionValues[j][i], buy[j]);
        }
        model.addConstr(ntot, GRB.EQUAL, nutrition[i], Categories[i]);
      }

      // Use barrier to solve model
      model.getEnv().set(GRB.IntParam.Method, GRB.METHOD_BARRIER);

      // Solve
      model.optimize();
      printSolution(model, buy, nutrition);

      System.out.println("\nAdding constraint: at most 6 servings of
dairy");
      GRBLinExpr lhs = new GRBLinExpr();
      lhs.addTerm(1.0, buy[7]);
      lhs.addTerm(1.0, buy[8]);
      model.addConstr(lhs, GRB.LESS_EQUAL, 6.0, "limit_dairy");

      // Solve
      model.optimize();
      printSolution(model, buy, nutrition);

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
  private static void printSolution(GRBModel model, GRBVar[] buy,
                                    GRBVar[] nutrition) throws
GRBException {
    if (model.get(GRB.IntAttr.Status) == GRB.Status.OPTIMAL) {
      System.out.println("\nCost: " + model.get(GRB.DoubleAttr.ObjVal));
      System.out.println("\nBuy:");
      for (int j = 0; j < buy.length; ++j) {
        if (buy[j].get(GRB.DoubleAttr.X) > 0.0001) {
          System.out.println(buy[j].get(GRB.StringAttr.VarName) + " " +
              buy[j].get(GRB.DoubleAttr.X));
        }
      }
      System.out.println("\nNutrition:");
      for (int i = 0; i < nutrition.length; ++i) {
        System.out.println(nutrition[i].get(GRB.StringAttr.VarName) + "
" +
            nutrition[i].get(GRB.DoubleAttr.X));
      }
    } else {
      System.out.println("No solution");
    }
  }
}
```

页面：Facility.java

Facility.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Facility location: a company currently ships its product from 5
plants
   to 4 warehouses. It is considering closing some plants to reduce
   costs. What plant(s) should the company close, in order to minimize
   transportation and fixed costs?

   Based on an example from Frontline Systems:
   http://www.solver.com/disfacility.htm
   Used with permission.
 */

import gurobi.*;

public class Facility {

  public static void main(String[] args) {
    try {

      // Warehouse demand in thousands of units
      double Demand[] = new double[] { 15, 18, 14, 20 };

      // Plant capacity in thousands of units
      double Capacity[] = new double[] { 20, 22, 17, 19, 18 };

      // Fixed costs for each plant
      double FixedCosts[] =
          new double[] { 12000, 15000, 17000, 13000, 16000 };

      // Transportation costs per thousand units
      double TransCosts[][] =
          new double[][] { { 4000, 2000, 3000, 2500, 4500 },
              { 2500, 2600, 3400, 3000, 4000 },
              { 1200, 1800, 2600, 4100, 3000 },
              { 2200, 2600, 3100, 3700, 3200 } };

      // Number of plants and warehouses
      int nPlants = Capacity.length;
      int nWarehouses = Demand.length;

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.set(GRB.StringAttr.ModelName, "facility");

      // Plant open decision variables: open[p] == 1 if plant p is open.
      GRBVar[] open = new GRBVar[nPlants];
      for (int p = 0; p < nPlants; ++p) {
        open[p] = model.addVar(0, 1, FixedCosts[p], GRB.BINARY, "Open"
+ p);
      }
      // Transportation decision variables: how much to transport from
      // a plant p to a warehouse w
      GRBVar[][] transport = new GRBVar[nWarehouses][nPlants];
      for (int w = 0; w < nWarehouses; ++w) {
        for (int p = 0; p < nPlants; ++p) {
          transport[w][p] =
              model.addVar(0, GRB.INFINITY, TransCosts[w][p],
GRB.CONTINUOUS,
                           "Trans" + p + "." + w);
        }
      }

      // The objective is to minimize the total fixed and variable
costs
      model.set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.update();

      // Production constraints
      // Note that the right-hand limit sets the production to zero if
      // the plant is closed
      for (int p = 0; p < nPlants; ++p) {
        GRBLinExpr ptot = new GRBLinExpr();
        for (int w = 0; w < nWarehouses; ++w) {
          ptot.addTerm(1.0, transport[w][p]);
        }
        GRBLinExpr limit = new GRBLinExpr();
        limit.addTerm(Capacity[p], open[p]);
        model.addConstr(ptot, GRB.LESS_EQUAL, limit, "Capacity" + p);
      }

      // Demand constraints
      for (int w = 0; w < nWarehouses; ++w) {
        GRBLinExpr dtot = new GRBLinExpr();
        for (int p = 0; p < nPlants; ++p) {
          dtot.addTerm(1.0, transport[w][p]);
        }
        model.addConstr(dtot, GRB.EQUAL, Demand[w], "Demand" + w);
      }

      // Guess at the starting point: close the plant with the highest
      // fixed costs; open all others

      // First, open all plants
      for (int p = 0; p < nPlants; ++p) {
        open[p].set(GRB.DoubleAttr.Start, 1.0);
      }

      // Now close the plant with the highest fixed cost
      System.out.println("Initial guess:");
      double maxFixed = -GRB.INFINITY;
      for (int p = 0; p < nPlants; ++p) {
        if (FixedCosts[p] > maxFixed) {
          maxFixed = FixedCosts[p];
        }
      }
      for (int p = 0; p < nPlants; ++p) {
        if (FixedCosts[p] == maxFixed) {
          open[p].set(GRB.DoubleAttr.Start, 0.0);
          System.out.println("Closing plant " + p + "\n");
          break;
        }
      }

      // Use barrier to solve root relaxation
      model.getEnv().set(GRB.IntParam.Method, GRB.METHOD_BARRIER);

      // Solve
      model.optimize();

      // Print solution
      System.out.println("\nTOTAL COSTS: " +
model.get(GRB.DoubleAttr.ObjVal));
      System.out.println("SOLUTION:");
      for (int p = 0; p < nPlants; ++p) {
        if (open[p].get(GRB.DoubleAttr.X) == 1.0) {
          System.out.println("Plant " + p + " open:");
          for (int w = 0; w < nWarehouses; ++w) {
            if (transport[w][p].get(GRB.DoubleAttr.X) > 0.0001) {
              System.out.println("  Transport " +
                  transport[w][p].get(GRB.DoubleAttr.X) +
                  " units to warehouse " + w);
            }
          }
        } else {
          System.out.println("Plant " + p + " closed!");
        }
      }

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Feasopt.java

Feasopt.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads a MIP model from a file, adds artificial
   variables to each constraint, and then minimizes the sum of the
   artificial variables.  A solution with objective zero corresponds
   to a feasible solution to the input model. */

import gurobi.*;

public class Feasopt {
  public static void main(String[] args) {

    if (args.length < 1) {
      System.out.println("Usage: java Feasopt filename");
      System.exit(1);
    }

    try {
      GRBEnv env = new GRBEnv();
      GRBModel feasmodel = new GRBModel(env, args[0]);

      // Clear objective
      feasmodel.setObjective(new GRBLinExpr());

      // Add slack variables
      GRBConstr[] c = feasmodel.getConstrs();
      for (int i = 0; i < c.length; ++i) {
        char sense = c[i].get(GRB.CharAttr.Sense);
        if (sense != '>') {
          GRBConstr[] constrs = new GRBConstr[] { c[i] };
          double[] coeffs = new double[] { -1 };
          feasmodel.addVar(0.0, GRB.INFINITY, 1.0, GRB.CONTINUOUS,
constrs,
                           coeffs, "ArtN_" +
                               c[i].get(GRB.StringAttr.ConstrName));
        }
        if (sense != '<') {
          GRBConstr[] constrs = new GRBConstr[] { c[i] };
          double[] coeffs = new double[] { 1 };
          feasmodel.addVar(0.0, GRB.INFINITY, 1.0, GRB.CONTINUOUS,
constrs,
                           coeffs, "ArtP_" +
                               c[i].get(GRB.StringAttr.ConstrName));
        }
      }
      feasmodel.update();

      // Optimize modified model
      feasmodel.write("feasopt.lp");
      feasmodel.optimize();

      // Dispose of model and environment
      feasmodel.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Fixanddive.java

Fixanddive.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Implement a simple MIP heuristic.  Relax the model,
 sort variables based on fractionality, and fix the 25% of
 the fractional variables that are closest to integer variables.
 Repeat until either the relaxation is integer feasible or
 linearly infeasible. */

import gurobi.*;
import java.util.*;

public class Fixanddive {
  public static void main(String[] args) {

    // Comparison class used to sort variable list based on relaxation
    // fractionality

    class FractionalCompare implements Comparator<GRBVar> {
      public int compare(GRBVar v1, GRBVar v2) {
        try {
          double sol1 = Math.abs(v1.get(GRB.DoubleAttr.X));
          double sol2 = Math.abs(v2.get(GRB.DoubleAttr.X));
          double frac1 = Math.abs(sol1 - Math.floor(sol1 + 0.5));
          double frac2 = Math.abs(sol2 - Math.floor(sol2 + 0.5));
          if (frac1 < frac2) {
            return -1;
          } else if (frac1 == frac2) {
            return 0;
          } else {
            return 1;
          }
        } catch (GRBException e) {
          System.out.println("Error code: " + e.getErrorCode() + ". " +
              e.getMessage());
        }
        return 0;
      }
    }

    if (args.length < 1) {
      System.out.println("Usage: java Fixanddive filename");
      System.exit(1);
    }

    try {
      // Read model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env, args[0]);

      // Collect integer variables and relax them
      ArrayList<GRBVar> intvars = new ArrayList<GRBVar>();
      for (GRBVar v : model.getVars()) {
        if (v.get(GRB.CharAttr.VType) != GRB.CONTINUOUS) {
          intvars.add(v);
          v.set(GRB.CharAttr.VType, GRB.CONTINUOUS);
        }
      }

      model.getEnv().set(GRB.IntParam.OutputFlag, 0);
      model.optimize();

      // Perform multiple iterations. In each iteration, identify the
first
      // quartile of integer variables that are closest to an integer
value
      // in the relaxation, fix them to the nearest integer, and repeat.

      for (int iter = 0; iter < 1000; ++iter) {

        // create a list of fractional variables, sorted in order of
        // increasing distance from the relaxation solution to the
nearest
        // integer value

        ArrayList<GRBVar> fractional = new ArrayList<GRBVar>();
        for (GRBVar v : intvars) {
          double sol = Math.abs(v.get(GRB.DoubleAttr.X));
          if (Math.abs(sol - Math.floor(sol + 0.5)) > 1e-5) {
            fractional.add(v);
          }
        }

        System.out.println("Iteration " + iter + ", obj " +
            model.get(GRB.DoubleAttr.ObjVal) + ", fractional " +
            fractional.size());

        if (fractional.size() == 0) {
          System.out.println("Found feasible solution - objective " +
              model.get(GRB.DoubleAttr.ObjVal));
          break;
        }

        // Fix the first quartile to the nearest integer value

        Collections.sort(fractional, new FractionalCompare());
        int nfix = Math.max(fractional.size() / 4, 1);
        for (int i = 0; i < nfix; ++i) {
          GRBVar v = fractional.get(i);
          double fixval = Math.floor(v.get(GRB.DoubleAttr.X) + 0.5);
          v.set(GRB.DoubleAttr.LB, fixval);
          v.set(GRB.DoubleAttr.UB, fixval);
          System.out.println("  Fix " + v.get(GRB.StringAttr.VarName) +
              " to " + fixval + " ( rel " + v.get(GRB.DoubleAttr.X) +
" )");
        }

        model.optimize();

        // Check optimization result
        if (model.get(GRB.IntAttr.Status) != GRB.Status.OPTIMAL) {
          System.out.println("Relaxation is infeasible");
          break;
        }
      }

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }

}
```

页面：Lp.java

Lp.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP model from a file and solves it.
   If the model is infeasible or unbounded, the example turns off
   presolve and solves the model again. If the model is infeasible,
   the example computes an Irreducible Infeasible Subsystem (IIS),
   and writes it to a file */

import gurobi.*;

public class Lp {
  public static void main(String[] args) {

    if (args.length < 1) {
      System.out.println("Usage: java Lp filename");
      System.exit(1);
    }

    try {
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env, args[0]);

      model.optimize();

      int optimstatus = model.get(GRB.IntAttr.Status);

      if (optimstatus == GRB.Status.INF_OR_UNBD) {
        model.getEnv().set(GRB.IntParam.Presolve, 0);
        model.optimize();
        optimstatus = model.get(GRB.IntAttr.Status);
      }

      if (optimstatus == GRB.Status.OPTIMAL) {
        double objval = model.get(GRB.DoubleAttr.ObjVal);
        System.out.println("Optimal objective: " + objval);
      } else if (optimstatus == GRB.Status.INFEASIBLE) {
        System.out.println("Model is infeasible");

        // Compute and write out IIS
        model.computeIIS();
        model.write("model.ilp");
      } else if (optimstatus == GRB.Status.UNBOUNDED) {
        System.out.println("Model is unbounded");
      } else {
        System.out.println("Optimization was stopped with status = "
                           + optimstatus);
      }

      // Dispose of model and environment
      model.dispose();
      env.dispose();
    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Lpmethod.java

Lpmethod.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Solve a model with different values of the Method parameter;
   show which value gives the shortest solve time. */

import gurobi.*;

public class Lpmethod {

  public static void main(String[] args) {

    if (args.length < 1) {
      System.out.println("Usage: java Lpmethod filename");
      System.exit(1);
    }

    try {
      // Read model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env, args[0]);
      GRBEnv menv = model.getEnv();

      // Solve the model with different values of Method
      int bestMethod = -1;
      double bestTime = menv.get(GRB.DoubleParam.TimeLimit);
      for (int i = 0; i <= 2; ++i) {
        model.reset();
        menv.set(GRB.IntParam.Method, i);
        model.optimize();
        if (model.get(GRB.IntAttr.Status) == GRB.Status.OPTIMAL) {
          bestTime = model.get(GRB.DoubleAttr.Runtime);
          bestMethod = i;
          // Reduce the TimeLimit parameter to save time
          // with other methods
          menv.set(GRB.DoubleParam.TimeLimit, bestTime);
        }
      }

      // Report which method was fastest
      if (bestMethod == -1) {
        System.out.println("Unable to solve this model");
      } else {
        System.out.println("Solved in " + bestTime
            + " seconds with Method: " + bestMethod);
      }

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". "
          + e.getMessage());
    }
  }
}
```

页面：Lpmod.java

Lpmod.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP model from a file and solves it.
   If the model can be solved, then it finds the smallest positive
variable,
   sets its upper bound to zero, and resolves the model two ways:
   first with an advanced start, then without an advanced start
   (i.e. 'from scratch'). */

import gurobi.*;

public class Lpmod {
  public static void main(String[] args) {

    if (args.length < 1) {
      System.out.println("Usage: java Lpmod filename");
      System.exit(1);
    }

    try {
      // Read model and determine whether it is an LP
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env, args[0]);
      if (model.get(GRB.IntAttr.IsMIP) != 0) {
        System.out.println("The model is not a linear program");
        System.exit(1);
      }

      model.optimize();

      int status = model.get(GRB.IntAttr.Status);

      if (status == GRB.Status.INF_OR_UNBD ||
          status == GRB.Status.INFEASIBLE  ||
          status == GRB.Status.UNBOUNDED     ) {
        System.out.println("The model cannot be solved because it is "
            + "infeasible or unbounded");
        System.exit(1);
      }

      if (status != GRB.Status.OPTIMAL) {
        System.out.println("Optimization was stopped with status " +
status);
        System.exit(0);
      }

      // Find the smallest variable value
      double minVal = GRB.INFINITY;
      GRBVar minVar = null;
      for (GRBVar v : model.getVars()) {
        double sol = v.get(GRB.DoubleAttr.X);
        if ((sol > 0.0001) && (sol < minVal) &&
            (v.get(GRB.DoubleAttr.LB) == 0.0)) {
          minVal = sol;
          minVar = v;
        }
      }

      System.out.println("\n*** Setting " +
          minVar.get(GRB.StringAttr.VarName) + " from " + minVal +
          " to zero ***\n");
      minVar.set(GRB.DoubleAttr.UB, 0.0);

      // Solve from this starting point
      model.optimize();

      // Save iteration & time info
      double warmCount = model.get(GRB.DoubleAttr.IterCount);
      double warmTime = model.get(GRB.DoubleAttr.Runtime);

      // Reset the model and resolve
      System.out.println("\n*** Resetting and solving "
          + "without an advanced start ***\n");
      model.reset();
      model.optimize();

      double coldCount = model.get(GRB.DoubleAttr.IterCount);
      double coldTime = model.get(GRB.DoubleAttr.Runtime);

      System.out.println("\n*** Warm start: " + warmCount + " 
iterations, " +
          warmTime + " seconds");
      System.out.println("*** Cold start: " + coldCount + " iterations, 
" +
          coldTime + " seconds");

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Mip1.java

Mip1.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example formulates and solves the following simple MIP model:

     maximize    x +   y + 2 z
     subject to  x + 2 y + 3 z <= 4
                 x +   y       >= 1
     x, y, z binary
*/

import gurobi.*;

public class Mip1 {
  public static void main(String[] args) {
    try {
      GRBEnv    env   = new GRBEnv("mip1.log");
      GRBModel  model = new GRBModel(env);

      // Create variables

      GRBVar x = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, "x");
      GRBVar y = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, "y");
      GRBVar z = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, "z");

      // Integrate new variables

      model.update();

      // Set objective: maximize x + y + 2 z

      GRBLinExpr expr = new GRBLinExpr();
      expr.addTerm(1.0, x); expr.addTerm(1.0, y); expr.addTerm(2.0, z);
      model.setObjective(expr, GRB.MAXIMIZE);

      // Add constraint: x + 2 y + 3 z <= 4

      expr = new GRBLinExpr();
      expr.addTerm(1.0, x); expr.addTerm(2.0, y); expr.addTerm(3.0, z);
      model.addConstr(expr, GRB.LESS_EQUAL, 4.0, "c0");

      // Add constraint: x + y >= 1

      expr = new GRBLinExpr();
      expr.addTerm(1.0, x); expr.addTerm(1.0, y);
      model.addConstr(expr, GRB.GREATER_EQUAL, 1.0, "c1");

      // Optimize model

      model.optimize();

      System.out.println(x.get(GRB.StringAttr.VarName)
                         + " " +x.get(GRB.DoubleAttr.X));
      System.out.println(y.get(GRB.StringAttr.VarName)
                         + " " +y.get(GRB.DoubleAttr.X));
      System.out.println(z.get(GRB.StringAttr.VarName)
                         + " " +z.get(GRB.DoubleAttr.X));

      System.out.println("Obj: " + model.get(GRB.DoubleAttr.ObjVal));

      // Dispose of model and environment

      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
                         e.getMessage());
    }
  }
}
```

页面：Mip2.java

Mip2.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads a MIP model from a file, solves it and
   prints the objective values from all feasible solutions
   generated while solving the MIP. Then it creates the fixed
   model and solves that model. */

import gurobi.*;

public class Mip2 {
  public static void main(String[] args) {

    if (args.length < 1) {
      System.out.println("Usage: java Mip2 filename");
      System.exit(1);
    }

    try {
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env, args[0]);
      if (model.get(GRB.IntAttr.IsMIP) == 0) {
        System.out.println("Model is not a MIP");
        System.exit(1);
      }

      model.optimize();

      int optimstatus = model.get(GRB.IntAttr.Status);
      double objval = 0;
      if (optimstatus == GRB.Status.OPTIMAL) {
        objval = model.get(GRB.DoubleAttr.ObjVal);
        System.out.println("Optimal objective: " + objval);
      } else if (optimstatus == GRB.Status.INF_OR_UNBD) {
        System.out.println("Model is infeasible or unbounded");
        return;
      } else if (optimstatus == GRB.Status.INFEASIBLE) {
        System.out.println("Model is infeasible");
        return;
      } else if (optimstatus == GRB.Status.UNBOUNDED) {
        System.out.println("Model is unbounded");
        return;
      } else {
        System.out.println("Optimization was stopped with status = "
            + optimstatus);
        return;
      }

      /* Iterate over the solutions and compute the objectives */
      GRBVar[] vars = model.getVars();
      model.getEnv().set(GRB.IntParam.OutputFlag, 0);

      System.out.println();
      for (int k = 0; k < model.get(GRB.IntAttr.SolCount); ++k) {
        model.getEnv().set(GRB.IntParam.SolutionNumber, k);
        double objn = 0.0;

        for (int j = 0; j < vars.length; j++) {
          objn += vars[j].get(GRB.DoubleAttr.Obj)
              * vars[j].get(GRB.DoubleAttr.Xn);
        }

        System.out.println("Solution " + k + " has objective: " + objn);
      }
      System.out.println();
      model.getEnv().set(GRB.IntParam.OutputFlag, 1);

      /* Create a fixed model, turn off presolve and solve */

      GRBModel fixed = model.fixedModel();

      fixed.getEnv().set(GRB.IntParam.Presolve, 0);

      fixed.optimize();

      int foptimstatus = fixed.get(GRB.IntAttr.Status);

      if (foptimstatus != GRB.Status.OPTIMAL) {
        System.err.println("Error: fixed model isn't optimal");
        return;
      }

      double fobjval = fixed.get(GRB.DoubleAttr.ObjVal);

      if (Math.abs(fobjval - objval) > 1.0e-6 * (1.0 +
Math.abs(objval))) {
        System.err.println("Error: objective values are different");
        return;
      }

      GRBVar[] fvars  = fixed.getVars();
      double[] x      = fixed.get(GRB.DoubleAttr.X, fvars);
      String[] vnames = fixed.get(GRB.StringAttr.VarName, fvars);

      for (int j = 0; j < fvars.length; j++) {
        if (x[j] != 0.0) {
          System.out.println(vnames[j] + " " + x[j]);
        }
      }

      // Dispose of models and environment
      fixed.dispose();
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". "
          + e.getMessage());
    }
  }
```

```
}
```

页面：Params.java

Params.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Use parameters that are associated with a model.

   A MIP is solved for 5 seconds with different sets of parameters.
   The one with the smallest MIP gap is selected, and the optimization
   is resumed until the optimal solution is found.
*/

import gurobi.*;

public class Params {

  // Simple function to determine the MIP gap
  private static double gap(GRBModel model) throws GRBException {
    if ((model.get(GRB.IntAttr.SolCount) == 0) ||
        (Math.abs(model.get(GRB.DoubleAttr.ObjVal)) < 1e-6)) {
      return GRB.INFINITY;
    }
    return Math.abs(model.get(GRB.DoubleAttr.ObjBound) -
        model.get(GRB.DoubleAttr.ObjVal)) /
        Math.abs(model.get(GRB.DoubleAttr.ObjVal));
  }

  public static void main(String[] args) {

    if (args.length < 1) {
      System.out.println("Usage: java Params filename");
      System.exit(1);
    }

    try {
      // Read model and verify that it is a MIP
      GRBEnv env = new GRBEnv();
      GRBModel base = new GRBModel(env, args[0]);
      if (base.get(GRB.IntAttr.IsMIP) == 0) {
        System.out.println("The model is not an integer program");
        System.exit(1);
      }

      // Set a 5 second time limit
      base.getEnv().set(GRB.DoubleParam.TimeLimit, 5);

      // Now solve the model with different values of MIPFocus
      double bestGap = GRB.INFINITY;
      GRBModel bestModel = null;
      for (int i = 0; i <= 3; ++i) {
        GRBModel m = new GRBModel(base);
        m.getEnv().set(GRB.IntParam.MIPFocus, i);
        m.optimize();
        if (bestModel == null || bestGap > gap(m)) {
          bestModel = m;
          bestGap = gap(bestModel);
        }
      }

      // Finally, reset the time limit and continue to solve the
      // best model to optimality
      bestModel.getEnv().set(GRB.DoubleParam.TimeLimit, GRB.INFINITY);
      bestModel.optimize();
      System.out.println("Solved with MIPFocus: " +
          bestModel.getEnv().get(GRB.IntParam.MIPFocus));

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Qp1.java

Qp1.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example formulates and solves the following simple QP model:

     minimize    x^2 + x*y + y^2 + y*z + z^2
     subject to  x + 2 y + 3 z >= 4
                 x +   y       >= 1

   It solves it once as a continuous model, and once as an integer
model.
*/

import gurobi.*;

public class Qp1 {
  public static void main(String[] args) {
    try {
      GRBEnv    env   = new GRBEnv("qp1.log");
      GRBModel  model = new GRBModel(env);

      // Create variables

      GRBVar x = model.addVar(0.0, 1.0, 0.0, GRB.CONTINUOUS, "x");
      GRBVar y = model.addVar(0.0, 1.0, 0.0, GRB.CONTINUOUS, "y");
      GRBVar z = model.addVar(0.0, 1.0, 0.0, GRB.CONTINUOUS, "z");

      // Integrate new variables

      model.update();

      // Set objective

      GRBQuadExpr obj = new GRBQuadExpr();
      obj.addTerm(1.0, x, x);
      obj.addTerm(1.0, x, y);
      obj.addTerm(1.0, y, y);
      obj.addTerm(1.0, y, z);
      obj.addTerm(1.0, z, z);
      model.setObjective(obj);

      // Add constraint: x + 2 y + 3 z >= 4

      GRBLinExpr expr = new GRBLinExpr();
      expr.addTerm(1.0, x); expr.addTerm(2.0, y); expr.addTerm(3.0, z);
      model.addConstr(expr, GRB.GREATER_EQUAL, 4.0, "c0");

      // Add constraint: x + y >= 1

      expr = new GRBLinExpr();
      expr.addTerm(1.0, x); expr.addTerm(1.0, y);
      model.addConstr(expr, GRB.GREATER_EQUAL, 1.0, "c1");
      // Optimize model

      model.optimize();

      System.out.println(x.get(GRB.StringAttr.VarName)
                         + " " +x.get(GRB.DoubleAttr.X));
      System.out.println(y.get(GRB.StringAttr.VarName)
                         + " " +y.get(GRB.DoubleAttr.X));
      System.out.println(z.get(GRB.StringAttr.VarName)
                         + " " +z.get(GRB.DoubleAttr.X));

      System.out.println("Obj: " + model.get(GRB.DoubleAttr.ObjVal) + "
" +
                         obj.getValue());
      System.out.println();


      // Change variable types to integer

      x.set(GRB.CharAttr.VType, GRB.INTEGER);
      y.set(GRB.CharAttr.VType, GRB.INTEGER);
      z.set(GRB.CharAttr.VType, GRB.INTEGER);

      // Optimize again

      model.optimize();

      System.out.println(x.get(GRB.StringAttr.VarName)
                         + " " +x.get(GRB.DoubleAttr.X));
      System.out.println(y.get(GRB.StringAttr.VarName)
                         + " " +y.get(GRB.DoubleAttr.X));
      System.out.println(z.get(GRB.StringAttr.VarName)
                         + " " +z.get(GRB.DoubleAttr.X));

      System.out.println("Obj: " + model.get(GRB.DoubleAttr.ObjVal) + "
" +
                         obj.getValue());

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Sensitivity.java

Sensitivity.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Simple MIP sensitivity analysis example.
 For each integer variable, fix it to its lower and upper bound
 and check the impact on the objective. */

import gurobi.*;

public class Sensitivity {

  public static void main(String[] args) {

    if (args.length < 1) {
      System.out.println("Usage: java Sensitivity filename");
      System.exit(1);
    }

    try {
      // Read model
      GRBEnv env = new GRBEnv();
      GRBModel a = new GRBModel(env, args[0]);
      a.optimize();
      a.getEnv().set(GRB.IntParam.OutputFlag, 0);

      // Extract variables from model
      GRBVar[] avars = a.getVars();

      for (int i = 0; i < avars.length; ++i) {
        GRBVar v = avars[i];
        if (v.get(GRB.CharAttr.VType) == GRB.BINARY) {

          // Create clone and fix variable
          GRBModel b = new GRBModel(a);
          GRBVar bv = b.getVars()[i];
          if (v.get(GRB.DoubleAttr.X) - v.get(GRB.DoubleAttr.LB) < 0.5)
{
            bv.set(GRB.DoubleAttr.LB, bv.get(GRB.DoubleAttr.UB));
          } else {
            bv.set(GRB.DoubleAttr.UB, bv.get(GRB.DoubleAttr.LB));
          }

          b.optimize();

          if (b.get(GRB.IntAttr.Status) == GRB.Status.OPTIMAL) {
            double objchg =
                b.get(GRB.DoubleAttr.ObjVal) -
a.get(GRB.DoubleAttr.ObjVal);
            if (objchg < 0) {
              objchg = 0;
            }
            System.out.println("Objective sensitivity for variable " +
                v.get(GRB.StringAttr.VarName) + " is " + objchg);
```

```
     } else {
      System.out.println("Objective sensitivity for variable " +
        v.get(GRB.StringAttr.VarName) + " is infinite");
     }

     // Dispose of model
     b.dispose();
    }
   }

   // Dispose of model and environment
   a.dispose();
   env.dispose();

  } catch (GRBException e) {
   System.out.println("Error code: " + e.getErrorCode() + ". " +
     e.getMessage());
  }
 }
}
```

页面：Sos.java

Sos.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example creates a very simple Special Ordered Set (SOS) model.
   The model consists of 3 continuous variables, no linear constraints,
   and a pair of SOS constraints of type 1. */

import gurobi.*;

public class Sos {
  public static void main(String[] args) {
    try {
      GRBEnv env = new GRBEnv();

      GRBModel model = new GRBModel(env);

      // Create variables

      double ub[]    = {1, 1, 2};
      double obj[]   = {-2, -1, -1};
      String names[] = {"x0", "x1", "x2"};

      GRBVar[] x = model.addVars(null, ub, obj, null, names);

      // Integrate new variables

      model.update();

      // Add first SOS1: x0=0 or x1=0

      GRBVar sosv1[]  = {x[0], x[1]};
      double soswt1[] = {1, 2};

      model.addSOS(sosv1, soswt1, GRB.SOS_TYPE1);

      // Add second SOS1: x0=0 or x2=0

      GRBVar sosv2[]  = {x[0], x[2]};
      double soswt2[] = {1, 2};

      model.addSOS(sosv2, soswt2, GRB.SOS_TYPE1);

      // Optimize model

      model.optimize();

      for (int i = 0; i < 3; i++)
        System.out.println(x[i].get(GRB.StringAttr.VarName) + " "
                           + x[i].get(GRB.DoubleAttr.X));

      // Dispose of model and environment
      model.dispose();
      env.dispose();
    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Sudoku.java

Sudoku.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */
/*
  Sudoku example.

  The Sudoku board is a 9x9 grid, which is further divided into a 3x3
grid
  of 3x3 grids.  Each cell in the grid must take a value from 0 to 9.
  No two grid cells in the same row, column, or 3x3 subgrid may take
the
  same value.

  In the MIP formulation, binary variables x[i,j,v] indicate whether
  cell <i,j> takes value 'v'.  The constraints are as follows:
    1. Each cell must take exactly one value (sum_v x[i,j,v] = 1)
    2. Each value is used exactly once per row (sum_i x[i,j,v] = 1)
    3. Each value is used exactly once per column (sum_j x[i,j,v] = 1)
    4. Each value is used exactly once per 3x3 subgrid (sum_grid
x[i,j,v] = 1)
*/

import gurobi.*;
import java.io.*;

public class Sudoku {
  public static void main(String[] args) {
    int n = 9;
    int s = 3;

    if (args.length < 1) {
      System.out.println("Usage: java Sudoku filename");
      System.exit(1);
    }

    try {
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);

      // Create 3-D array of model variables

      GRBVar[][][] vars = new GRBVar[n][n][n];

      for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
          for (int v = 0; v < n; v++) {
            String st = "G_" + String.valueOf(i) + "_" +
String.valueOf(j)
                                 + "_" + String.valueOf(v);
            vars[i][j][v] = model.addVar(0.0, 1.0, 0.0, GRB.BINARY, st);
          }
        }
      }
      // Integrate variables into model

      model.update();

      // Add constraints

      GRBLinExpr expr;

      // Each cell must take one value

      for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
          expr = new GRBLinExpr();
          expr.addTerms(null, vars[i][j]);
          String st = "V_" + String.valueOf(i) + "_" +
String.valueOf(j);
          model.addConstr(expr, GRB.EQUAL, 1.0, st);
        }
      }

      // Each value appears once per row

      for (int i = 0; i < n; i++) {
        for (int v = 0; v < n; v++) {
          expr = new GRBLinExpr();
          for (int j = 0; j < n; j++)
            expr.addTerm(1.0, vars[i][j][v]);
          String st = "R_" + String.valueOf(i) + "_" +
String.valueOf(v);
          model.addConstr(expr, GRB.EQUAL, 1.0, st);
        }
      }

      // Each value appears once per column

      for (int j = 0; j < n; j++) {
        for (int v = 0; v < n; v++) {
          expr = new GRBLinExpr();
          for (int i = 0; i < n; i++)
            expr.addTerm(1.0, vars[i][j][v]);
          String st = "C_" + String.valueOf(j) + "_" +
String.valueOf(v);
          model.addConstr(expr, GRB.EQUAL, 1.0, st);
        }
      }

      // Each value appears once per sub-grid

      for (int v = 0; v < n; v++) {
        for (int i0 = 0; i0 < s; i0++) {
          for (int j0 = 0; j0 < s; j0++) {
            expr = new GRBLinExpr();
            for (int i1 = 0; i1 < s; i1++) {
              for (int j1 = 0; j1 < s; j1++) {
                expr.addTerm(1.0, vars[i0*s+i1][j0*s+j1][v]);
              }
            }
            String st = "Sub_" + String.valueOf(v) + "_" +
String.valueOf(i0)
                               + "_" + String.valueOf(j0);
            model.addConstr(expr, GRB.EQUAL, 1.0, st);
          }
        }
      }

      // Update model

      model.update();

      // Fix variables associated with pre-specified cells

      File file = new File(args[0]);
      FileInputStream fis = new FileInputStream(file);
      byte[] input = new byte[n];

      for (int i = 0; i < n; i++) {
        fis.read(input);
        for (int j = 0; j < n; j++) {
          int val = (int) input[j] - 48 - 1; // 0-based

          if (val >= 0)
            vars[i][j][val].set(GRB.DoubleAttr.LB, 1.0);
        }
        // read the endline byte
        fis.read();
      }

      // Optimize model

      model.optimize();

      // Write model to file
      model.write("sudoku.lp");

      double[][][] x = model.get(GRB.DoubleAttr.X, vars);

      System.out.println();
      for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
          for (int v = 0; v < n; v++) {
            if (x[i][j][v] > 0.5) {
              System.out.print(v+1);
            }
          }
        }
        System.out.println();
      }

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    } catch (IOException e) {
      System.out.println("IO Error");
    }
  }
}
```

页面：Workforce1_java

Workforce1.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, use IIS to find a
set of
   conflicting constraints. Note that there may be additional conflicts
   besides what is reported via IIS. */

import gurobi.*;

public class Workforce1 {

  public static void main(String[] args) {
    try {

      // Sample data
      // Sets of days and workers
      String Shifts[] =
          new String[] { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
              "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12",
"Sat13",
              "Sun14" };
      String Workers[] =
          new String[] { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred",
"Gu" };

      int nShifts = Shifts.length;
      int nWorkers = Workers.length;

      // Number of workers required for each shift
      double shiftRequirements[] =
          new double[] { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

      // Amount each worker is paid to work one shift
      double pay[] = new double[] { 10, 12, 10, 8, 8, 9, 11 };

      // Worker availability: 0 if the worker is unavailable for a
shift
      double availability[][] =
          new double[][] { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
              { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
              { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
              { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
              { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.set(GRB.StringAttr.ModelName, "assignment");
      // Assignment variables: x[w][s] == 1 if worker w is assigned
      // to shift s. Since an assignment model always produces integer
      // solutions, we use continuous variables and solve as an LP.
      GRBVar[][] x = new GRBVar[nWorkers][nShifts];
      for (int w = 0; w < nWorkers; ++w) {
        for (int s = 0; s < nShifts; ++s) {
          x[w][s] =
              model.addVar(0, availability[w][s], pay[w],
GRB.CONTINUOUS,
                           Workers[w] + "." + Shifts[s]);
        }
      }

      // The objective is to minimize the total pay costs
      model.set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.update();

      // Constraint: assign exactly shiftRequirements[s] workers
      // to each shift s
      for (int s = 0; s < nShifts; ++s) {
        GRBLinExpr lhs = new GRBLinExpr();
        for (int w = 0; w < nWorkers; ++w) {
          lhs.addTerm(1.0, x[w][s]);
        }
        model.addConstr(lhs, GRB.EQUAL, shiftRequirements[s],
Shifts[s]);
      }

      // Optimize
      model.optimize();
      int status = model.get(GRB.IntAttr.Status);
      if (status == GRB.Status.UNBOUNDED) {
        System.out.println("The model cannot be solved "
            + "because it is unbounded");
        return;
      }
      if (status == GRB.Status.OPTIMAL) {
        System.out.println("The optimal objective is " +
            model.get(GRB.DoubleAttr.ObjVal));
        return;
      }
      if (status != GRB.Status.INF_OR_UNBD &&
          status != GRB.Status.INFEASIBLE    ){
        System.out.println("Optimization was stopped with status " +
status);
        return;
      }

      // Compute IIS
      System.out.println("The model is infeasible; computing IIS");
      model.computeIIS();
      System.out.println("\nThe following constraint(s) "
          + "cannot be satisfied:");
      for (GRBConstr c : model.getConstrs()) {
        if (c.get(GRB.IntAttr.IISConstr) == 1) {
          System.out.println(c.get(GRB.StringAttr.ConstrName));
        }
      }

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Workforce2.java

Workforce2.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, use IIS iteratively
to
   find all conflicting constraints. */

import gurobi.*;
import java.util.*;

public class Workforce2 {

  public static void main(String[] args) {
    try {

      // Sample data
      // Sets of days and workers
      String Shifts[] =
          new String[] { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
              "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12",
"Sat13",
              "Sun14" };
      String Workers[] =
          new String[] { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred",
"Gu" };

      int nShifts = Shifts.length;
      int nWorkers = Workers.length;

      // Number of workers required for each shift
      double shiftRequirements[] =
          new double[] { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

      // Amount each worker is paid to work one shift
      double pay[] = new double[] { 10, 12, 10, 8, 8, 9, 11 };

      // Worker availability: 0 if the worker is unavailable for a
shift
      double availability[][] =
          new double[][] { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
              { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
              { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
              { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
              { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.set(GRB.StringAttr.ModelName, "assignment");
      // Assignment variables: x[w][s] == 1 if worker w is assigned
      // to shift s. Since an assignment model always produces integer
      // solutions, we use continuous variables and solve as an LP.
      GRBVar[][] x = new GRBVar[nWorkers][nShifts];
      for (int w = 0; w < nWorkers; ++w) {
        for (int s = 0; s < nShifts; ++s) {
          x[w][s] =
              model.addVar(0, availability[w][s], pay[w],
GRB.CONTINUOUS,
                           Workers[w] + "." + Shifts[s]);
        }
      }

      // The objective is to minimize the total pay costs
      model.set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.update();

      // Constraint: assign exactly shiftRequirements[s] workers
      // to each shift s
      for (int s = 0; s < nShifts; ++s) {
        GRBLinExpr lhs = new GRBLinExpr();
        for (int w = 0; w < nWorkers; ++w) {
          lhs.addTerm(1.0, x[w][s]);
        }
        model.addConstr(lhs, GRB.EQUAL, shiftRequirements[s],
Shifts[s]);
      }

      // Optimize
      model.optimize();
      int status = model.get(GRB.IntAttr.Status);
      if (status == GRB.Status.UNBOUNDED) {
        System.out.println("The model cannot be solved "
            + "because it is unbounded");
        return;
      }
      if (status == GRB.Status.OPTIMAL) {
        System.out.println("The optimal objective is " +
            model.get(GRB.DoubleAttr.ObjVal));
        return;
      }
      if (status != GRB.Status.INF_OR_UNBD &&
          status != GRB.Status.INFEASIBLE    ) {
        System.out.println("Optimization was stopped with status " +
status);
        return;
      }

      // Do IIS
      System.out.println("The model is infeasible; computing IIS");
      LinkedList<String> removed = new LinkedList<String>();

      // Loop until we reduce to a model that can be solved
      while (true) {
        model.computeIIS();
        System.out.println("\nThe following constraint cannot be 
satisfied:");
        for (GRBConstr c : model.getConstrs()) {
          if (c.get(GRB.IntAttr.IISConstr) == 1) {
            System.out.println(c.get(GRB.StringAttr.ConstrName));
            // Remove a single constraint from the model
            removed.add(c.get(GRB.StringAttr.ConstrName));
            model.remove(c);
            break;
          }
        }

        System.out.println();
        model.optimize();
        status = model.get(GRB.IntAttr.Status);

        if (status == GRB.Status.UNBOUNDED) {
          System.out.println("The model cannot be solved "
              + "because it is unbounded");
          return;
        }
        if (status == GRB.Status.OPTIMAL) {
          break;
        }
        if (status != GRB.Status.INF_OR_UNBD &&
            status != GRB.Status.INFEASIBLE    ) {
          System.out.println("Optimization was stopped with status " +
              status);
          return;
        }
      }

      System.out.println("\nThe following constraints were removed "
          + "to get a feasible LP:");
      for (String s : removed) {
        System.out.print(s + " ");
      }
      System.out.println();

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Workforce3.java

Workforce3.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, add slack variables
   to determine which constraints cannot be satisfied, and how much
   they need to be relaxed. */

import gurobi.*;
import java.util.*;

public class Workforce3 {

  public static void main(String[] args) {
    try {

      // Sample data
      // Sets of days and workers
      String Shifts[] =
          new String[] { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
              "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12",
"Sat13",
              "Sun14" };
      String Workers[] =
          new String[] { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred",
"Gu" };

      int nShifts = Shifts.length;
      int nWorkers = Workers.length;

      // Number of workers required for each shift
      double shiftRequirements[] =
          new double[] { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

      // Amount each worker is paid to work one shift
      double pay[] = new double[] { 10, 12, 10, 8, 8, 9, 11 };

      // Worker availability: 0 if the worker is unavailable for a
shift
      double availability[][] =
          new double[][] { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
              { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
              { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
              { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
              { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.set(GRB.StringAttr.ModelName, "assignment");
      // Assignment variables: x[w][s] == 1 if worker w is assigned
      // to shift s. Since an assignment model always produces integer
      // solutions, we use continuous variables and solve as an LP.
      GRBVar[][] x = new GRBVar[nWorkers][nShifts];
      for (int w = 0; w < nWorkers; ++w) {
        for (int s = 0; s < nShifts; ++s) {
          x[w][s] =
              model.addVar(0, availability[w][s], pay[w],
GRB.CONTINUOUS,
                           Workers[w] + "." + Shifts[s]);
        }
      }

      // The objective is to minimize the total pay costs
      model.set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.update();

      // Constraint: assign exactly shiftRequirements[s] workers
      // to each shift s
      LinkedList<GRBConstr> reqCts = new LinkedList<GRBConstr>();
      for (int s = 0; s < nShifts; ++s) {
        GRBLinExpr lhs = new GRBLinExpr();
        for (int w = 0; w < nWorkers; ++w) {
          lhs.addTerm(1.0, x[w][s]);
        }
        GRBConstr newCt =
            model.addConstr(lhs, GRB.EQUAL, shiftRequirements[s],
Shifts[s]);
        reqCts.add(newCt);
      }

      // Optimize
      model.optimize();
      int status = model.get(GRB.IntAttr.Status);
      if (status == GRB.Status.UNBOUNDED) {
        System.out.println("The model cannot be solved "
            + "because it is unbounded");
        return;
      }
      if (status == GRB.Status.OPTIMAL) {
        System.out.println("The optimal objective is " +
            model.get(GRB.DoubleAttr.ObjVal));
        return;
      }
      if (status != GRB.Status.INF_OR_UNBD &&
          status != GRB.Status.INFEASIBLE    ) {
        System.out.println("Optimization was stopped with status " +
status);
        return;
      }

      // Add slack variables to make the model feasible
      System.out.println("The model is infeasible; adding slack
variables");
```

```
      // Set original objective coefficients to zero
      model.setObjective(new GRBLinExpr());

      // Add a new slack variable to each shift constraint so that the
shifts
      // can be satisfied
      LinkedList<GRBVar> slacks = new LinkedList<GRBVar>();
      for (GRBConstr c : reqCts) {
        GRBColumn col = new GRBColumn();
        col.addTerm(1.0, c);
        GRBVar newvar =
            model.addVar(0, GRB.INFINITY, 1.0, GRB.CONTINUOUS, col,
                         c.get(GRB.StringAttr.ConstrName) + "Slack");
        slacks.add(newvar);
      }

      // Solve the model with slacks
      model.optimize();
      status = model.get(GRB.IntAttr.Status);
      if (status == GRB.Status.INF_OR_UNBD ||
          status == GRB.Status.INFEASIBLE  ||
          status == GRB.Status.UNBOUNDED     ) {
        System.out.println("The model with slacks cannot be solved "
            + "because it is infeasible or unbounded");
        return;
      }
      if (status != GRB.Status.OPTIMAL) {
        System.out.println("Optimization was stopped with status " +
status);
        return;
      }

      System.out.println("\nSlack values:");
      for (GRBVar sv : slacks) {
        if (sv.get(GRB.DoubleAttr.X) > 1e-6) {
          System.out.println(sv.get(GRB.StringAttr.VarName) + " = " +
              sv.get(GRB.DoubleAttr.X));
        }
      }

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }
}
```

页面：Workforce4.java

Workforce4.java

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. We use Pareto optimization to solve the model:
   first, we minimize the linear sum of the slacks. Then, we constrain
   the sum of the slacks, and we minimize a quadratic objective that
   tries to balance the workload among the workers. */

import gurobi.*;

public class Workforce4 {

  public static void main(String[] args) {
    try {

      // Sample data
      // Sets of days and workers
      String Shifts[] =
          new String[] { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
              "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12",
"Sat13",
              "Sun14" };
      String Workers[] =
          new String[] { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred",
"Gu" };

      int nShifts = Shifts.length;
      int nWorkers = Workers.length;

      // Number of workers required for each shift
      double shiftRequirements[] =
          new double[] { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

      // Worker availability: 0 if the worker is unavailable for a
shift
      double availability[][] =
          new double[][] { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
              { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
              { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
              { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
              { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.set(GRB.StringAttr.ModelName, "assignment");

      // Assignment variables: x[w][s] == 1 if worker w is assigned
      // to shift s. This is no longer a pure assignment model, so we
must
      // use binary variables.
      GRBVar[][] x = new GRBVar[nWorkers][nShifts];
      for (int w = 0; w < nWorkers; ++w) {
        for (int s = 0; s < nShifts; ++s) {
          x[w][s] =
              model.addVar(0, availability[w][s], 0, GRB.BINARY,
                           Workers[w] + "." + Shifts[s]);
        }
      }

      // Slack variables for each shift constraint so that the shifts
can
      // be satisfied
      GRBVar[] slacks = new GRBVar[nShifts];
      for (int s = 0; s < nShifts; ++s) {
        slacks[s] =
            model.addVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS,
                         Shifts[s] + "Slack");
      }

      // Variable to represent the total slack
      GRBVar totSlack = model.addVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS,
                                     "totSlack");

      // Variables to count the total shifts worked by each worker
      GRBVar[] totShifts = new GRBVar[nWorkers];
      for (int w = 0; w < nWorkers; ++w) {
        totShifts[w] = model.addVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS,
                                    Workers[w] + "TotShifts");
      }

      // Update model to integrate new variables
      model.update();

      GRBLinExpr lhs;

      // Constraint: assign exactly shiftRequirements[s] workers
      // to each shift s, plus the slack
      for (int s = 0; s < nShifts; ++s) {
        lhs = new GRBLinExpr();
        lhs.addTerm(1.0, slacks[s]);
        for (int w = 0; w < nWorkers; ++w) {
          lhs.addTerm(1.0, x[w][s]);
        }
        model.addConstr(lhs, GRB.EQUAL, shiftRequirements[s],
Shifts[s]);
      }

      // Constraint: set totSlack equal to the total slack
      lhs = new GRBLinExpr();
      lhs.addTerm(-1.0, totSlack);
      for (int s = 0; s < nShifts; ++s) {
        lhs.addTerm(1.0, slacks[s]);
      }
      model.addConstr(lhs, GRB.EQUAL, 0, "totSlack");
      // Constraint: compute the total number of shifts for each worker
      for (int w = 0; w < nWorkers; ++w) {
        lhs = new GRBLinExpr();
        lhs.addTerm(-1.0, totShifts[w]);
        for (int s = 0; s < nShifts; ++s) {
          lhs.addTerm(1.0, x[w][s]);
        }
        model.addConstr(lhs, GRB.EQUAL, 0, "totShifts" + Workers[w]);
      }

      // Objective: minimize the total slack
      GRBLinExpr obj = new GRBLinExpr();
      obj.addTerm(1.0, totSlack);
      model.setObjective(obj);

      // Optimize
      int status =
        solveAndPrint(model, totSlack, nWorkers, Workers, totShifts);
      if (status != GRB.Status.OPTIMAL ) {
        return;
      }

      // Constrain the slack by setting its upper and lower bounds
      totSlack.set(GRB.DoubleAttr.UB, totSlack.get(GRB.DoubleAttr.X));
      totSlack.set(GRB.DoubleAttr.LB, totSlack.get(GRB.DoubleAttr.X));

      // Variable to count the average number of shifts worked
      GRBVar avgShifts =
        model.addVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS, "avgShifts");

      // Variables to count the difference from average for each worker;
      // note that these variables can take negative values.
      GRBVar[] diffShifts = new GRBVar[nWorkers];
      for (int w = 0; w < nWorkers; ++w) {
        diffShifts[w] = model.addVar(-GRB.INFINITY, GRB.INFINITY, 0,
                                     GRB.CONTINUOUS, Workers[w] +
"Diff");
      }

      // Update model to integrate new variables
      model.update();

      // Constraint: compute the average number of shifts worked
      lhs = new GRBLinExpr();
      lhs.addTerm(-nWorkers, avgShifts);
      for (int w = 0; w < nWorkers; ++w) {
        lhs.addTerm(1.0, totShifts[w]);
      }
      model.addConstr(lhs, GRB.EQUAL, 0, "avgShifts");

      // Constraint: compute the difference from the average number of
shifts
      for (int w = 0; w < nWorkers; ++w) {
        lhs = new GRBLinExpr();
        lhs.addTerm(-1, diffShifts[w]);
        lhs.addTerm(-1, avgShifts);
        lhs.addTerm( 1, totShifts[w]);
        model.addConstr(lhs, GRB.EQUAL, 0, Workers[w] + "Diff");
      }

      // Objective: minimize the sum of the square of the difference
from the
      // average number of shifts worked
      GRBQuadExpr qobj = new GRBQuadExpr();
      for (int w = 0; w < nWorkers; ++w) {
        qobj.addTerm(1.0, diffShifts[w], diffShifts[w]);
      }
      model.setObjective(qobj);

      // Optimize
      status =
        solveAndPrint(model, totSlack, nWorkers, Workers, totShifts);
      if (status != GRB.Status.OPTIMAL ) {
        return;
      }

      // Dispose of model and environment
      model.dispose();
      env.dispose();

    } catch (GRBException e) {
      System.out.println("Error code: " + e.getErrorCode() + ". " +
          e.getMessage());
    }
  }

  private static int solveAndPrint(GRBModel model, GRBVar totSlack,
                                   int nWorkers, String[] Workers,
                                   GRBVar[] totShifts) throws
GRBException {

    model.optimize();
    int status = model.get(GRB.IntAttr.Status);
    if (status == GRB.Status.INF_OR_UNBD ||
        status == GRB.Status.INFEASIBLE  ||
        status == GRB.Status.UNBOUNDED     ) {
      System.out.println("The model cannot be solved "
          + "because it is infeasible or unbounded");
      return status;
    }
    if (status != GRB.Status.OPTIMAL ) {
      System.out.println("Optimization was stopped with status " +
status);
      return status;
    }

    // Print total slack and the number of shifts worked for each
worker
    System.out.println("\nTotal slack required: " +
                       totSlack.get(GRB.DoubleAttr.X));
    for (int w = 0; w < nWorkers; ++w) {
      System.out.println(Workers[w] + " worked " +
                         totShifts[w].get(GRB.DoubleAttr.X) + "
shifts");
    }
    System.out.println("\n");
    return status;
  }

}
```

页面： C# Examples

C# 示例

本节包含了所有 Gurobi C# 的示例源代码。相同的源代码也可以在 Gurobi 发布包的 *examples/c#* 目录中找到。

小节

- callback_cs.cs
- diet_cs.cs
- facility_cs.cs
- feasopt_cs.cs
- fixanddive_cs.cs
- lp_cs.cs
- lpmethod_cs.cs
- lpmod_cs.cs
- mip1_cs.cs
- mip2_cs.cs
- params_cs.cs
- qp1_cs.cs
- sensitivity_cs.cs
- sos_cs.cs
- sudoku_cs.cs
- workforce1_cs.cs
- workforce2_cs.cs
- workforce3_cs.cs
- workforce4_cs.cs

页面：callback_cs.cs

callback_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP or a MIP from a file, sets a callback
   to monitor the optimization progress and to output some progress
   information to the screen and to a log file. If it is a MIP and 10%
   gap is reached, then it aborts. */

using System;
using Gurobi;

class callback_cs : GRBCallback
{
  private GRBVar[] vars;
  private int      lastmsg;

  public callback_cs(GRBVar[] xvars)
  {
    vars = xvars;
    lastmsg = -100;
  }

  protected override void Callback()
  {
    try {
      if (where == GRB.Callback.MESSAGE) {
        string st = GetStringInfo(GRB.Callback.MSG_STRING);
        if (st != null) Console.Write(st);
      } else if (where == GRB.Callback.PRESOLVE) {
        int cdels = GetIntInfo(GRB.Callback.PRE_COLDEL);
        int rdels = GetIntInfo(GRB.Callback.PRE_ROWDEL);
        Console.WriteLine(cdels+" columns and "+rdels+" rows are
removed");
      } else if (where == GRB.Callback.SIMPLEX) {
        int itcnt = (int) GetDoubleInfo(GRB.Callback.SPX_ITRCNT);
        if (itcnt%100 == 0) {
          double obj  = GetDoubleInfo(GRB.Callback.SPX_OBJVAL);
          double pinf = GetDoubleInfo(GRB.Callback.SPX_PRIMINF);
          double dinf = GetDoubleInfo(GRB.Callback.SPX_DUALINF);
          int  ispert = GetIntInfo(GRB.Callback.SPX_ISPERT);
          char ch;
          if (ispert == 0)      ch = ' ';
          else if (ispert == 1) ch = 'S';
          else                  ch = 'P';
          Console.WriteLine(itcnt+"  "+ obj + ch + "  "+pinf + "  " +
dinf);
        }
      } else if (where == GRB.Callback.MIP) {
        int nodecnt = (int) GetDoubleInfo(GRB.Callback.MIP_NODCNT);
        if (nodecnt - lastmsg >= 100) {
          lastmsg = nodecnt;
          double objbst = GetDoubleInfo(GRB.Callback.MIP_OBJBST);
          double objbnd = GetDoubleInfo(GRB.Callback.MIP_OBJBND);
          if (Math.Abs(objbst - objbnd) < 0.1 * (1.0 +
Math.Abs(objbst)))
            Abort();
          int actnodes = (int) GetDoubleInfo(GRB.Callback.MIP_NODLFT);
          int itcnt    = (int) GetDoubleInfo(GRB.Callback.MIP_ITRCNT);
          int solcnt   = GetIntInfo(GRB.Callback.MIP_SOLCNT);
          int cutcnt   = GetIntInfo(GRB.Callback.MIP_CUTCNT);
          Console.WriteLine(nodecnt + " " +  actnodes + " " +  itcnt +
" "
            +  objbst + " " +  objbnd + " " +  solcnt + " " +  cutcnt);
        }
      } else if (where == GRB.Callback.MIPSOL) {
        double obj     = GetDoubleInfo(GRB.Callback.MIPSOL_OBJ);
        int    nodecnt = (int)
GetDoubleInfo(GRB.Callback.MIPSOL_NODCNT);
        double[] x = GetSolution(vars);
        Console.WriteLine("**** New solution at node " + nodecnt + ",
obj "
                           + obj + ", x[0] = " + x[0] + "****");
      }
    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
      Console.WriteLine(e.StackTrace);
    }
  }

  static void Main(string[] args)
  {
    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: callback_cs filename");
      return;
    }

    try {
      GRBEnv    env   = new GRBEnv();
      GRBModel  model = new GRBModel(env, args[0]);

      GRBVar[] vars   = model.GetVars();

      model.SetCallback(new callback_cs(vars));
      model.Optimize();

      double[] x      = model.Get(GRB.DoubleAttr.X, vars);
      string[] vnames = model.Get(GRB.StringAttr.VarName, vars);

      for (int j = 0; j < vars.Length; j++) {
        if (x[j] != 0.0) Console.WriteLine(vnames[j] + " " + x[j]);
      }

      for (int j = 0; j < vars.Length; j++) {
        if (x[j] != 0.0) Console.WriteLine(vnames[j] + " " + x[j]);
      }

      // Dispose of model and env
      model.Dispose();
      env.Dispose();
    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
      Console.WriteLine(e.StackTrace);
    }
  }
}
```

页面：diet_cs.cs

diet_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Solve the classic diet model, showing how to add constraints
   to an existing model. */

using System;
using Gurobi;

class diet_cs
{
  static void Main()
  {
    try {

      // Nutrition guidelines, based on
      // USDA Dietary Guidelines for Americans, 2005
      // http://www.health.gov/DietaryGuidelines/dga2005/
      string[] Categories =
          new string[] { "calories", "protein", "fat", "sodium" };
      int nCategories = Categories.Length;
      double[] minNutrition = new double[] { 1800, 91, 0, 0 };
      double[] maxNutrition = new double[] { 2200, GRB.INFINITY, 65,
1779 };

      // Set of foods
      string[] Foods =
          new string[] { "hamburger", "chicken", "hot dog", "fries",
              "macaroni", "pizza", "salad", "milk", "ice cream" };
      int nFoods = Foods.Length;
      double[] cost =
          new double[] { 2.49, 2.89, 1.50, 1.89, 2.09, 1.99, 2.49, 0.89,
              1.59 };

      // Nutrition values for the foods
      double[,] nutritionValues = new double[,] {
          { 410, 24, 26, 730 },   // hamburger
          { 420, 32, 10, 1190 },  // chicken
          { 560, 20, 32, 1800 },  // hot dog
          { 380, 4, 19, 270 },    // fries
          { 320, 12, 10, 930 },   // macaroni
          { 320, 15, 12, 820 },   // pizza
          { 320, 31, 12, 1230 },  // salad
          { 100, 8, 2.5, 125 },   // milk
          { 330, 8, 10, 180 }     // ice cream
          };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.Set(GRB.StringAttr.ModelName, "diet");

      // Create decision variables for the nutrition information,
      // which we limit via bounds
      GRBVar[] nutrition = new GRBVar[nCategories];
      for (int i = 0; i < nCategories; ++i) {
        nutrition[i] =
            model.AddVar(minNutrition[i], maxNutrition[i], 0,
GRB.CONTINUOUS,
                         Categories[i]);
      }

      // Create decision variables for the foods to buy
      GRBVar[] buy = new GRBVar[nFoods];
      for (int j = 0; j < nFoods; ++j) {
        buy[j] =
            model.AddVar(0, GRB.INFINITY, cost[j], GRB.CONTINUOUS,
Foods[j]);
      }

      // The objective is to minimize the costs
      model.Set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.Update();

      // Nutrition constraints
      for (int i = 0; i < nCategories; ++i) {
        GRBLinExpr ntot = 0.0;
        for (int j = 0; j < nFoods; ++j)
          ntot += nutritionValues[j,i] * buy[j];
        model.AddConstr(ntot == nutrition[i], Categories[i]);
      }

      // Use barrier to solve model
      model.GetEnv().Set(GRB.IntParam.Method, GRB.METHOD_BARRIER);

      // Solve
      model.Optimize();
      PrintSolution(model, buy, nutrition);

      Console.WriteLine("\nAdding constraint: at most 6 servings of
dairy");
      model.AddConstr(buy[7] + buy[8] <= 6.0, "limit_dairy");

      // Solve
      model.Optimize();
      PrintSolution(model, buy, nutrition);

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
          e.Message);
    }
  }

  private static void PrintSolution(GRBModel model, GRBVar[] buy,
                                    GRBVar[] nutrition) {
    if (model.Get(GRB.IntAttr.Status) == GRB.Status.OPTIMAL) {
      Console.WriteLine("\nCost: " + model.Get(GRB.DoubleAttr.ObjVal));
      Console.WriteLine("\nBuy:");
      for (int j = 0; j < buy.Length; ++j) {
        if (buy[j].Get(GRB.DoubleAttr.X) > 0.0001) {
          Console.WriteLine(buy[j].Get(GRB.StringAttr.VarName) + " " +
              buy[j].Get(GRB.DoubleAttr.X));
        }
      }
      Console.WriteLine("\nNutrition:");
      for (int i = 0; i < nutrition.Length; ++i) {
        Console.WriteLine(nutrition[i].Get(GRB.StringAttr.VarName) + "
" +
            nutrition[i].Get(GRB.DoubleAttr.X));
      }
    } else {
      Console.WriteLine("No solution");
    }
  }
}
```

页面：facility_cs.cs

facility_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Facility location: a company currently ships its product from 5
plants
   to 4 warehouses. It is considering closing some plants to reduce
   costs. What plant(s) should the company close, in order to minimize
   transportation and fixed costs?

   Based on an example from Frontline Systems:
   http://www.solver.com/disfacility.htm
   Used with permission.
 */

using System;
using Gurobi;

class facility_cs
{
  static void Main()
  {
    try {

      // Warehouse demand in thousands of units
      double[] Demand = new double[] { 15, 18, 14, 20 };

      // Plant capacity in thousands of units
      double[] Capacity = new double[] { 20, 22, 17, 19, 18 };

      // Fixed costs for each plant
      double[] FixedCosts =
          new double[] { 12000, 15000, 17000, 13000, 16000 };

      // Transportation costs per thousand units
      double[,] TransCosts =
          new double[,] { { 4000, 2000, 3000, 2500, 4500 },
              { 2500, 2600, 3400, 3000, 4000 },
              { 1200, 1800, 2600, 4100, 3000 },
              { 2200, 2600, 3100, 3700, 3200 } };

      // Number of plants and warehouses
      int nPlants = Capacity.Length;
      int nWarehouses = Demand.Length;

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.Set(GRB.StringAttr.ModelName, "facility");

      // Plant open decision variables: open[p] == 1 if plant p is open.
      GRBVar[] open = new GRBVar[nPlants];
      for (int p = 0; p < nPlants; ++p) {
        open[p] = model.AddVar(0, 1, FixedCosts[p], GRB.BINARY, "Open"
+ p);
      }

      // Transportation decision variables: how much to transport from
      // a plant p to a warehouse w
      GRBVar[,] transport = new GRBVar[nWarehouses,nPlants];
      for (int w = 0; w < nWarehouses; ++w) {
        for (int p = 0; p < nPlants; ++p) {
          transport[w,p] =
              model.AddVar(0, GRB.INFINITY, TransCosts[w,p],
GRB.CONTINUOUS,
                           "Trans" + p + "." + w);
        }
      }

      // The objective is to minimize the total fixed and variable
costs
      model.Set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.Update();

      // Production constraints
      // Note that the right-hand limit sets the production to zero if
      // the plant is closed
      for (int p = 0; p < nPlants; ++p) {
        GRBLinExpr ptot = 0.0;
        for (int w = 0; w < nWarehouses; ++w)
          ptot += transport[w,p];
        model.AddConstr(ptot <= Capacity[p] * open[p], "Capacity" + p);
      }

      // Demand constraints
      for (int w = 0; w < nWarehouses; ++w) {
        GRBLinExpr dtot = 0.0;
        for (int p = 0; p < nPlants; ++p)
          dtot += transport[w,p];
        model.AddConstr(dtot == Demand[w], "Demand" + w);
      }

      // Guess at the starting point: close the plant with the highest
      // fixed costs; open all others

      // First, open all plants
      for (int p = 0; p < nPlants; ++p) {
        open[p].Set(GRB.DoubleAttr.Start, 1.0);
      }

      // Now close the plant with the highest fixed cost
      Console.WriteLine("Initial guess:");
      double maxFixed = -GRB.INFINITY;
      for (int p = 0; p < nPlants; ++p) {
        if (FixedCosts[p] > maxFixed) {
          maxFixed = FixedCosts[p];
        }
      }
      for (int p = 0; p < nPlants; ++p) {
        if (FixedCosts[p] == maxFixed) {
          open[p].Set(GRB.DoubleAttr.Start, 0.0);
          Console.WriteLine("Closing plant " + p + "\n");
          break;
        }
      }

      // Use barrier to solve root relaxation
      model.GetEnv().Set(GRB.IntParam.Method, GRB.METHOD_BARRIER);

      // Solve
      model.Optimize();

      // Print solution
      Console.WriteLine("\nTOTAL COSTS: " +
model.Get(GRB.DoubleAttr.ObjVal));
      Console.WriteLine("SOLUTION:");
      for (int p = 0; p < nPlants; ++p) {
        if (open[p].Get(GRB.DoubleAttr.X) == 1.0) {
          Console.WriteLine("Plant " + p + " open:");
          for (int w = 0; w < nWarehouses; ++w) {
            if (transport[w,p].Get(GRB.DoubleAttr.X) > 0.0001) {
              Console.WriteLine("  Transport " +
                  transport[w,p].Get(GRB.DoubleAttr.X) +
                  " units to warehouse " + w);
            }
          }
        } else {
          Console.WriteLine("Plant " + p + " closed!");
        }
      }

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
  }
}
```

页面：feasopt_cs.cs

feasopt_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads a MIP model from a file, adds artificial
   variables to each constraint, and then minimizes the sum of the
   artificial variables.  A solution with objective zero corresponds
   to a feasible solution to the input model. */

using Gurobi;
using System;

class feasopt_cs
{
  static void Main(string[] args)
  {
    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: feasopt_cs filename");
      return;
    }

    try {
      GRBEnv env = new GRBEnv();
      GRBModel feasmodel = new GRBModel(env, args[0]);

      // Clear objective
      feasmodel.SetObjective(new GRBLinExpr());

      // Add slack variables
      GRBConstr[] c = feasmodel.GetConstrs();
      for (int i = 0; i < c.Length; ++i) {
        char sense = c[i].Get(GRB.CharAttr.Sense);
        if (sense != '>') {
          GRBConstr[] constrs = new GRBConstr[] { c[i] };
          double[] coeffs = new double[] { -1 };
          feasmodel.AddVar(0.0, GRB.INFINITY, 1.0, GRB.CONTINUOUS,
constrs,
                           coeffs, "ArtN_" +
c[i].Get(GRB.StringAttr.ConstrName));
        }
        if (sense != '<') {
          GRBConstr[] constrs = new GRBConstr[] { c[i] };
          double[] coeffs = new double[] { 1 };
          feasmodel.AddVar(0.0, GRB.INFINITY, 1.0, GRB.CONTINUOUS,
constrs,
                           coeffs, "ArtP_" +
                               c[i].Get(GRB.StringAttr.ConstrName));
        }
      }
      feasmodel.Update();

      // Optimize modified model
      feasmodel.Write("feasopt.lp");
      feasmodel.Optimize();
      // Dispose of model and env
      feasmodel.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
  }
}
```

页面：fixanddive_cs.cs

fixanddive_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Implement a simple MIP heuristic.  Relax the model,
   sort variables based on fractionality, and fix the 25% of
   the fractional variables that are closest to integer variables.
   Repeat until either the relaxation is integer feasible or
   linearly infeasible. */

using System;
using System.Collections.Generic;
using Gurobi;

class fixanddive_cs
{
  // Comparison class used to sort variable list based on relaxation
  // fractionality

  class FractionalCompare : IComparer<GRBVar>
  {
    public int Compare(GRBVar v1, GRBVar v2)
    {
      try {
        double sol1 = Math.Abs(v1.Get(GRB.DoubleAttr.X));
        double sol2 = Math.Abs(v2.Get(GRB.DoubleAttr.X));
        double frac1 = Math.Abs(sol1 - Math.Floor(sol1 + 0.5));
        double frac2 = Math.Abs(sol2 - Math.Floor(sol2 + 0.5));
        if (frac1 < frac2) {
          return -1;
        } else if (frac1 > frac2) {
          return 1;
        } else {
          return 0;
        }
      } catch (GRBException e) {
        Console.WriteLine("Error code: " + e.ErrorCode + ". " +
            e.Message);
      }
      return 0;
    }
  }

  static void Main(string[] args)
  {
    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: fixanddive_cs filename");
      return;
    }

    try {
      // Read model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env, args[0]);
      // Collect integer variables and relax them
      List<GRBVar> intvars = new List<GRBVar>();
      foreach (GRBVar v in model.GetVars()) {
        if (v.Get(GRB.CharAttr.VType) != GRB.CONTINUOUS) {
          intvars.Add(v);
          v.Set(GRB.CharAttr.VType, GRB.CONTINUOUS);
        }
      }

      model.GetEnv().Set(GRB.IntParam.OutputFlag, 0);
      model.Optimize();

      // Perform multiple iterations. In each iteration, identify the
first
      // quartile of integer variables that are closest to an integer
value
      // in the relaxation, fix them to the nearest integer, and repeat.

      for (int iter = 0; iter < 1000; ++iter) {

        // create a list of fractional variables, sorted in order of
        // increasing distance from the relaxation solution to the
nearest
        // integer value

        List<GRBVar> fractional = new List<GRBVar>();
        foreach (GRBVar v in intvars) {
          double sol = Math.Abs(v.Get(GRB.DoubleAttr.X));
          if (Math.Abs(sol - Math.Floor(sol + 0.5)) > 1e-5) {
            fractional.Add(v);
          }
        }

        Console.WriteLine("Iteration " + iter + ", obj " +
            model.Get(GRB.DoubleAttr.ObjVal) + ", fractional " +
            fractional.Count);

        if (fractional.Count == 0) {
          Console.WriteLine("Found feasible solution - objective " +
              model.Get(GRB.DoubleAttr.ObjVal));
          break;
        }

        // Fix the first quartile to the nearest integer value

        fractional.Sort(new FractionalCompare());
        int nfix = Math.Max(fractional.Count / 4, 1);
        for (int i = 0; i < nfix; ++i) {
          GRBVar v = fractional[i];
          double fixval = Math.Floor(v.Get(GRB.DoubleAttr.X) + 0.5);
          v.Set(GRB.DoubleAttr.LB, fixval);
          v.Set(GRB.DoubleAttr.UB, fixval);
          Console.WriteLine("  Fix " + v.Get(GRB.StringAttr.VarName) +
              " to " + fixval + " ( rel " + v.Get(GRB.DoubleAttr.X) +
" )");
        }
        model.Optimize();

        // Check optimization result

        if (model.Get(GRB.IntAttr.Status) != GRB.Status.OPTIMAL) {
          Console.WriteLine("Relaxation is infeasible");
          break;
        }
      }

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
          e.Message);
    }
  }
}
```

页面：lp_cs.cs

lp_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP model from a file and solves it.
   If the model is infeasible or unbounded, the example turns off
   presolve and solves the model again. If the model is infeasible,
   the example computes an Irreducible Infeasible Subsystem (IIS),
   and writes it to a file. */

using System;
using Gurobi;

class lp_cs
{
  static void Main(string[] args)
  {
    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: lp_cs filename");
      return;
    }

    try {
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env, args[0]);

      model.Optimize();

      int optimstatus = model.Get(GRB.IntAttr.Status);

      if (optimstatus == GRB.Status.INF_OR_UNBD) {
        model.GetEnv().Set(GRB.IntParam.Presolve, 0);
        model.Optimize();
        optimstatus = model.Get(GRB.IntAttr.Status);
      }

      if (optimstatus == GRB.Status.OPTIMAL) {
        double objval = model.Get(GRB.DoubleAttr.ObjVal);
        Console.WriteLine("Optimal objective: " + objval);
      } else if (optimstatus == GRB.Status.INFEASIBLE) {
        Console.WriteLine("Model is infeasible");

        // compute and write out IIS

        model.ComputeIIS();
        model.Write("model.ilp");
      } else if (optimstatus == GRB.Status.UNBOUNDED) {
        Console.WriteLine("Model is unbounded");
      } else {
        Console.WriteLine("Optimization was stopped with status = "
                          + optimstatus);
      }

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
  }
}
```

页面：lpmethod_cs.cs

lpmethod_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Solve a model with different values of the Method parameter;
   show which value gives the shortest solve time. */

using System;
using Gurobi;

class lpmethod_cs
{
  static void Main(string[] args)
  {
    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: lpmethod_cs filename");
      return;
    }

    try {
      // Read model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env, args[0]);
      GRBEnv menv = model.GetEnv();

      // Solve the model with different values of Method
      int bestMethod = -1;
      double bestTime = menv.Get(GRB.DoubleParam.TimeLimit);
      for (int i = 0; i <= 2; ++i)
      {
        model.Reset();
        menv.Set(GRB.IntParam.Method, i);
        model.Optimize();
        if (model.Get(GRB.IntAttr.Status) == GRB.Status.OPTIMAL)
        {
          bestTime = model.Get(GRB.DoubleAttr.Runtime);
          bestMethod = i;
          // Reduce the TimeLimit parameter to save time
          // with other methods
          menv.Set(GRB.DoubleParam.TimeLimit, bestTime);
        }
      }

      // Report which method was fastest
      if (bestMethod == -1) {
        Console.WriteLine("Unable to solve this model");
      } else {
        Console.WriteLine("Solved in " + bestTime
          + " seconds with Method: " + bestMethod);
      }

      // Dispose of model and env
      model.Dispose();
      env.Dispose();
    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
  }
}
```

页面：lpmod_cs.cs

lpmod_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads an LP model from a file and solves it.
   If the model can be solved, then it finds the smallest positive
variable,
   sets its upper bound to zero, and resolves the model two ways:
   first with an advanced start, then without an advanced start
   (i.e. 'from scratch'). */

using System;
using Gurobi;

class lpmod_cs
{
  static void Main(string[] args)
  {
    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: lpmod_cs filename");
      return;
    }

    try {
      // Read model and determine whether it is an LP
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env, args[0]);
      if (model.Get(GRB.IntAttr.IsMIP) != 0) {
        Console.WriteLine("The model is not a linear program");
        Environment.Exit(1);
      }

      model.Optimize();

      int status = model.Get(GRB.IntAttr.Status);

      if ((status == GRB.Status.INF_OR_UNBD) ||
          (status == GRB.Status.INFEASIBLE) ||
          (status == GRB.Status.UNBOUNDED)) {
        Console.WriteLine("The model cannot be solved because it is "
            + "infeasible or unbounded");
        Environment.Exit(1);
      }

      if (status != GRB.Status.OPTIMAL) {
        Console.WriteLine("Optimization was stopped with status " +
status);
        Environment.Exit(0);
      }

      // Find the smallest variable value
      double minVal = GRB.INFINITY;
      GRBVar minVar = null;
      foreach (GRBVar v in model.GetVars()) {
        double sol = v.Get(GRB.DoubleAttr.X);
        if ((sol > 0.0001) && (sol < minVal) &&
            (v.Get(GRB.DoubleAttr.LB) == 0.0)) {
          minVal = sol;
          minVar = v;
        }
      }

      Console.WriteLine("\n*** Setting " +
          minVar.Get(GRB.StringAttr.VarName) + " from " + minVal +
          " to zero ***\n");
      minVar.Set(GRB.DoubleAttr.UB, 0.0);

      // Solve from this starting point
      model.Optimize();

      // Save iteration & time info
      double warmCount = model.Get(GRB.DoubleAttr.IterCount);
      double warmTime = model.Get(GRB.DoubleAttr.Runtime);

      // Reset the model and resolve
      Console.WriteLine("\n*** Resetting and solving "
          + "without an advanced start ***\n");
      model.Reset();
      model.Optimize();

      double coldCount = model.Get(GRB.DoubleAttr.IterCount);
      double coldTime = model.Get(GRB.DoubleAttr.Runtime);

      Console.WriteLine("\n*** Warm start: " + warmCount + " iterations,
" +
          warmTime + " seconds");
      Console.WriteLine("*** Cold start: " + coldCount + " iterations,
" +
          coldTime + " seconds");

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
          e.Message);
    }
  }
}
```

页面：mip1_cs.cs

mip1_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example formulates and solves the following simple MIP model:

     maximize    x +   y + 2 z
     subject to  x + 2 y + 3 z <= 4
                 x +   y       >= 1
     x, y, z binary
*/

using System;
using Gurobi;

class mip1_cs
{
  static void Main()
  {
    try {
      GRBEnv    env   = new GRBEnv("mip1.log");
      GRBModel  model = new GRBModel(env);

      // Create variables

      GRBVar x = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "x");
      GRBVar y = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "y");
      GRBVar z = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "z");

      // Integrate new variables

      model.Update();

      // Set objective: maximize x + y + 2 z

      model.SetObjective(x + y + 2 * z, GRB.MAXIMIZE);

      // Add constraint: x + 2 y + 3 z <= 4

      model.AddConstr(x + 2 * y + 3 * z <= 4.0, "c0");

      // Add constraint: x + y >= 1

      model.AddConstr(x + y >= 1.0, "c1");

      // Optimize model

      model.Optimize();

      Console.WriteLine(x.Get(GRB.StringAttr.VarName)
                         + " " + x.Get(GRB.DoubleAttr.X));
      Console.WriteLine(y.Get(GRB.StringAttr.VarName)
                         + " " + y.Get(GRB.DoubleAttr.X));
      Console.WriteLine(z.Get(GRB.StringAttr.VarName)
                         + " " + z.Get(GRB.DoubleAttr.X));

      Console.WriteLine("Obj: " + model.Get(GRB.DoubleAttr.ObjVal));

      // Dispose of model and env

      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
  }
}
```

页面：mip2_cs.cs

mip2_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example reads a MIP model from a file, solves it and
   prints the objective values from all feasible solutions
   generated while solving the MIP. Then it creates the fixed
   model and solves that model. */

using System;
using Gurobi;

class mip2_cs
{
  static void Main(string[] args)
  {
    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: mip2_cs filename");
      return;
    }

    try {
      GRBEnv    env   = new GRBEnv();
      GRBModel  model = new GRBModel(env, args[0]);
      if (model.Get(GRB.IntAttr.IsMIP) == 0) {
        Console.WriteLine("Model is not a MIP");
        return;
      }

      model.Optimize();

      int optimstatus = model.Get(GRB.IntAttr.Status);
      double objval = 0;
      if (optimstatus == GRB.Status.OPTIMAL) {
        objval = model.Get(GRB.DoubleAttr.ObjVal);
        Console.WriteLine("Optimal objective: " + objval);
      } else if (optimstatus == GRB.Status.INF_OR_UNBD) {
        Console.WriteLine("Model is infeasible or unbounded");
        return;
      } else if (optimstatus == GRB.Status.INFEASIBLE) {
        Console.WriteLine("Model is infeasible");
        return;
      } else if (optimstatus == GRB.Status.UNBOUNDED) {
        Console.WriteLine("Model is unbounded");
        return;
      } else {
        Console.WriteLine("Optimization was stopped with status = "
                          + optimstatus);
        return;
      }

      /* Iterate over the solutions and compute the objectives */

      GRBVar[] vars = model.GetVars();
      model.GetEnv().Set(GRB.IntParam.OutputFlag, 0);

      Console.WriteLine();
      for (int k = 0; k < model.Get(GRB.IntAttr.SolCount); ++k) {
        model.GetEnv().Set(GRB.IntParam.SolutionNumber, k);
        double objn = 0.0;

        for (int j = 0; j < vars.Length; j++) {
          objn += vars[j].Get(GRB.DoubleAttr.Obj)
            * vars[j].Get(GRB.DoubleAttr.Xn);
        }

        Console.WriteLine("Solution " + k + " has objective: " + objn);
      }
      Console.WriteLine();
      model.GetEnv().Set(GRB.IntParam.OutputFlag, 1);

      /* Create a fixed model, turn off presolve and solve */

      GRBModel fixedmodel = model.FixedModel();

      fixedmodel.GetEnv().Set(GRB.IntParam.Presolve, 0);

      fixedmodel.Optimize();

      int foptimstatus = fixedmodel.Get(GRB.IntAttr.Status);

      if (foptimstatus != GRB.Status.OPTIMAL) {
        Console.WriteLine("Error: fixed model isn't optimal");
        return;
      }

      double fobjval = fixedmodel.Get(GRB.DoubleAttr.ObjVal);

      if (Math.Abs(fobjval - objval) > 1.0e-6 * (1.0 +
Math.Abs(objval))) {
        Console.WriteLine("Error: objective values are different");
        return;
      }

      GRBVar[] fvars  = fixedmodel.GetVars();
      double[] x      = fixedmodel.Get(GRB.DoubleAttr.X, fvars);
      string[] vnames = fixedmodel.Get(GRB.StringAttr.VarName, fvars);

      for (int j = 0; j < fvars.Length; j++) {
        if (x[j] != 0.0) Console.WriteLine(vnames[j] + " " + x[j]);
      }

      // Dispose of models and env
      fixedmodel.Dispose();
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
    }
}
```

页面：params_cs.cs

params_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Use parameters that are associated with a model.

   A MIP is solved for 5 seconds with different sets of parameters.
   The one with the smallest MIP gap is selected, and the optimization
   is resumed until the optimal solution is found.
*/

using System;
using Gurobi;

class params_cs
{
  // Simple function to determine the MIP gap
  private static double Gap(GRBModel model) {
    if ((model.Get(GRB.IntAttr.SolCount) == 0) ||
        (Math.Abs(model.Get(GRB.DoubleAttr.ObjVal)) < 1e-6)) {
      return GRB.INFINITY;
    }
    return Math.Abs(model.Get(GRB.DoubleAttr.ObjBound) -
        model.Get(GRB.DoubleAttr.ObjVal)) /
        Math.Abs(model.Get(GRB.DoubleAttr.ObjVal));
  }

  static void Main(string[] args)
  {
    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: params_cs filename");
      return;
    }

    try {
      // Read model and verify that it is a MIP
      GRBEnv env = new GRBEnv();
      GRBModel basemodel = new GRBModel(env, args[0]);
      if (basemodel.Get(GRB.IntAttr.IsMIP) == 0) {
        Console.WriteLine("The model is not an integer program");
        Environment.Exit(1);
      }

      // Set a 5 second time limit
      basemodel.GetEnv().Set(GRB.DoubleParam.TimeLimit, 5);

      // Now solve the model with different values of MIPFocus
      double bestGap = GRB.INFINITY;
      GRBModel bestModel = null;
      for (int i = 0; i <= 3; ++i) {
        GRBModel m = new GRBModel(basemodel);
        m.GetEnv().Set(GRB.IntParam.MIPFocus, i);
        m.Optimize();
        if (bestModel == null || bestGap > Gap(m)) {
          bestModel = m;
          bestGap = Gap(bestModel);
        }
      }

      // Finally, reset the time limit and continue to solve the
      // best model to optimality
      bestModel.GetEnv().Set(GRB.DoubleParam.TimeLimit, GRB.INFINITY);
      bestModel.Optimize();
      Console.WriteLine("Solved with MIPFocus: " +
          bestModel.GetEnv().Get(GRB.IntParam.MIPFocus));

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
          e.Message);
    }
  }
}
```

页面：qp1_cs.cs

qp1_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example formulates and solves the following simple QP model:

     minimize    x^2 + x*y + y^2 + y*z + z^2
     subject to  x + 2 y + 3 z >= 4
                 x +   y       >= 1

   It solves it once as a continuous model, and once as an integer
model.
*/

using System;
using Gurobi;

class qp1_cs
{
  static void Main()
  {
    try {
      GRBEnv    env   = new GRBEnv("qp1.log");
      GRBModel  model = new GRBModel(env);

      // Create variables

      GRBVar x = model.AddVar(0.0, 1.0, 0.0, GRB.CONTINUOUS, "x");
      GRBVar y = model.AddVar(0.0, 1.0, 0.0, GRB.CONTINUOUS, "y");
      GRBVar z = model.AddVar(0.0, 1.0, 0.0, GRB.CONTINUOUS, "z");

      // Integrate new variables

      model.Update();

      // Set objective

      GRBQuadExpr obj = x*x + x*y + y*y + y*z + z*z;
      model.SetObjective(obj);

      // Add constraint: x + 2 y + 3 z >= 4

      model.AddConstr(x + 2 * y + 3 * z >= 4.0, "c0");

      // Add constraint: x + y >= 1

      model.AddConstr(x + y >= 1.0, "c1");

      // Optimize model

      model.Optimize();

      Console.WriteLine(x.Get(GRB.StringAttr.VarName)
                         + " " + x.Get(GRB.DoubleAttr.X));
      Console.WriteLine(y.Get(GRB.StringAttr.VarName)
                         + " " + y.Get(GRB.DoubleAttr.X));
      Console.WriteLine(z.Get(GRB.StringAttr.VarName)
                         + " " + z.Get(GRB.DoubleAttr.X));

      Console.WriteLine("Obj: " + model.Get(GRB.DoubleAttr.ObjVal) + "
" +
                        obj.Value);


      // Change variable types to integer

      x.Set(GRB.CharAttr.VType, GRB.INTEGER);
      y.Set(GRB.CharAttr.VType, GRB.INTEGER);
      z.Set(GRB.CharAttr.VType, GRB.INTEGER);

      // Optimize model

      model.Optimize();

      Console.WriteLine(x.Get(GRB.StringAttr.VarName)
                         + " " + x.Get(GRB.DoubleAttr.X));
      Console.WriteLine(y.Get(GRB.StringAttr.VarName)
                         + " " + y.Get(GRB.DoubleAttr.X));
      Console.WriteLine(z.Get(GRB.StringAttr.VarName)
                         + " " + z.Get(GRB.DoubleAttr.X));

      Console.WriteLine("Obj: " + model.Get(GRB.DoubleAttr.ObjVal) + "
" +
                        obj.Value);

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
  }
}
```

页面：sensitivity_cs.cs

sensitivity_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Simple MIP sensitivity analysis example.
   For each integer variable, fix it to its lower and upper bound
   and check the impact on the objective. */

using System;
using Gurobi;

class sensitivity_cs
{
  static void Main(string[] args)
  {
    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: sensitivity_cs filename");
      return;
    }

    try {
      // Read model
      GRBEnv env = new GRBEnv();
      GRBModel a = new GRBModel(env, args[0]);
      a.Optimize();
      a.GetEnv().Set(GRB.IntParam.OutputFlag, 0);

      // Extract variables from model
      GRBVar[] avars = a.GetVars();

      for (int i = 0; i < avars.Length; ++i) {
        GRBVar v = avars[i];
        if (v.Get(GRB.CharAttr.VType) == GRB.BINARY) {

          // Create clone and fix variable
          GRBModel b = new GRBModel(a);
          GRBVar bv = b.GetVars()[i];
          if (v.Get(GRB.DoubleAttr.X) - v.Get(GRB.DoubleAttr.LB) < 0.5)
{
            bv.Set(GRB.DoubleAttr.LB, bv.Get(GRB.DoubleAttr.UB));
          } else {
            bv.Set(GRB.DoubleAttr.UB, bv.Get(GRB.DoubleAttr.LB));
          }

          b.Optimize();

          if (b.Get(GRB.IntAttr.Status) == GRB.Status.OPTIMAL) {
            double objchg =
                b.Get(GRB.DoubleAttr.ObjVal) -
a.Get(GRB.DoubleAttr.ObjVal);
            if (objchg < 0) {
              objchg = 0;
            }
            Console.WriteLine("Objective sensitivity for variable " +
        v.Get(GRB.StringAttr.VarName) + " is " + objchg);
     } else {
      Console.WriteLine("Objective sensitivity for variable " +
        v.Get(GRB.StringAttr.VarName) + " is infinite");
     }

     // Dispose of model
     b.Dispose();
    }
   }

   // Dispose of model and env
   a.Dispose();
   env.Dispose();

  } catch (GRBException e) {
   Console.WriteLine("Error code: " + e.ErrorCode + ". " +
     e.Message);
  }
 }
}
```

页面：sos_cs.cs

sos_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* This example creates a very simple Special Ordered Set (SOS) model.
   The model consists of 3 continuous variables, no linear constraints,
   and a pair of SOS constraints of type 1. */

using System;
using Gurobi;

class sos_cs
{
  static void Main()
  {
    try {
      GRBEnv env = new GRBEnv();

      GRBModel model = new GRBModel(env);

      // Create variables

      double[] ub    = {1, 1, 2};
      double[] obj   = {-2, -1, -1};
      string[] names = {"x0", "x1", "x2"};

      GRBVar[] x = model.AddVars(null, ub, obj, null, names);

      // Integrate new variables

      model.Update();

      // Add first SOS1: x0=0 or x1=0

      GRBVar[] sosv1  = {x[0], x[1]};
      double[] soswt1 = {1, 2};

      model.AddSOS(sosv1, soswt1, GRB.SOS_TYPE1);

      // Add second SOS1: x0=0 or x2=0

      GRBVar[] sosv2  = {x[0], x[2]};
      double[] soswt2 = {1, 2};

      model.AddSOS(sosv2, soswt2, GRB.SOS_TYPE1);

      // Optimize model

      model.Optimize();

      for (int i = 0; i < 3; i++)
        Console.WriteLine(x[i].Get(GRB.StringAttr.VarName) + " "
                          + x[i].Get(GRB.DoubleAttr.X));
      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
  }
}
```

页面：sudoku_cs.cs

Sudoku_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/*
  Sudoku example.

  The Sudoku board is a 9x9 grid, which is further divided into a 3x3
grid
  of 3x3 grids.  Each cell in the grid must take a value from 0 to 9.
  No two grid cells in the same row, column, or 3x3 subgrid may take
the
  same value.

  In the MIP formulation, binary variables x[i,j,v] indicate whether
  cell <i,j> takes value 'v'.  The constraints are as follows:
    1. Each cell must take exactly one value (sum_v x[i,j,v] = 1)
    2. Each value is used exactly once per row (sum_i x[i,j,v] = 1)
    3. Each value is used exactly once per column (sum_j x[i,j,v] = 1)
    4. Each value is used exactly once per 3x3 subgrid (sum_grid
x[i,j,v] = 1)
*/

using System;
using System.IO;
using Gurobi;

class sudoku_cs
{
  static void Main(string[] args)
  {
    int n = 9;
    int s = 3;

    if (args.Length < 1) {
      Console.Out.WriteLine("Usage: sudoku_cs filename");
      return;
    }

    try {
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);

      // Create 3-D array of model variables

      GRBVar[,,] vars = new GRBVar[n,n,n];

      for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
          for (int v = 0; v < n; v++) {
            string st = "G_" + i.ToString() + "_" + j.ToString()
                              + "_" + v.ToString();
            vars[i,j,v] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, st);
          }
        }
      }

      // Integrate variables into model

      model.Update();

      // Add constraints

      GRBLinExpr expr;

      // Each cell must take one value

      for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
          expr = 0.0;
          for (int v = 0; v < n; v++)
            expr += vars[i,j,v];
          string st = "V_" + i.ToString() + "_" + j.ToString();
          model.AddConstr(expr == 1.0, st);
        }
      }

      // Each value appears once per row

      for (int i = 0; i < n; i++) {
        for (int v = 0; v < n; v++) {
          expr = 0.0;
          for (int j = 0; j < n; j++)
            expr += vars[i,j,v];
          string st = "R_" + i.ToString() + "_" + v.ToString();
          model.AddConstr(expr == 1.0, st);
        }
      }

      // Each value appears once per column

      for (int j = 0; j < n; j++) {
        for (int v = 0; v < n; v++) {
          expr = 0.0;
          for (int i = 0; i < n; i++)
            expr += vars[i,j,v];
          string st = "C_" + j.ToString() + "_" + v.ToString();
          model.AddConstr(expr == 1.0, st);
        }
      }

      // Each value appears once per sub-grid

      for (int v = 0; v < n; v++) {
        for (int i0 = 0; i0 < s; i0++) {
          for (int j0 = 0; j0 < s; j0++) {
            expr = 0.0;
            for (int i1 = 0; i1 < s; i1++) {
              for (int j1 = 0; j1 < s; j1++) {
                expr += vars[i0*s+i1,j0*s+j1,v];
              }
```

```
            }
            string st = "Sub_" + v.ToString() + "_" + i0.ToString()
                               + "_" + j0.ToString();
            model.AddConstr(expr == 1.0, st);
          }
        }
      }

      // Update model

      model.Update();

      // Fix variables associated with pre-specified cells

      StreamReader sr = File.OpenText(args[0]);

      for (int i = 0; i < n; i++) {
        string input = sr.ReadLine();
        for (int j = 0; j < n; j++) {
          int val = (int) input[j] - 48 - 1; // 0-based

          if (val >= 0)
            vars[i,j,val].Set(GRB.DoubleAttr.LB, 1.0);
        }
      }

      // Optimize model

      model.Optimize();

      // Write model to file
      model.Write("sudoku.lp");

      double[,,] x = model.Get(GRB.DoubleAttr.X, vars);

      Console.WriteLine();
      for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
          for (int v = 0; v < n; v++) {
            if (x[i,j,v] > 0.5) {
              Console.Write(v+1);
            }
          }
        }
        Console.WriteLine();
      }

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
e.Message);
    }
  }
}
```

页面：workforce1_cs.cs

workforce1_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, use IIS to find a
set of
   conflicting constraints. Note that there may be additional conflicts
   besides what is reported via IIS. */

using System;
using Gurobi;

class workforce1_cs
{
  static void Main()
  {
    try {

      // Sample data
      // Sets of days and workers
      string[] Shifts =
          new string[] { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
              "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12",
"Sat13",
              "Sun14" };
      string[] Workers =
          new string[] { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred",
"Gu" };

      int nShifts = Shifts.Length;
      int nWorkers = Workers.Length;

      // Number of workers required for each shift
      double[] shiftRequirements =
          new double[] { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

      // Amount each worker is paid to work one shift
      double[] pay = new double[] { 10, 12, 10, 8, 8, 9, 11 };

      // Worker availability: 0 if the worker is unavailable for a
shift
      double[,] availability =
          new double[,] { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
              { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
              { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
              { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
              { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.Set(GRB.StringAttr.ModelName, "assignment");

      // Assignment variables: x[w][s] == 1 if worker w is assigned
      // to shift s. Since an assignment model always produces integer
      // solutions, we use continuous variables and solve as an LP.
      GRBVar[,] x = new GRBVar[nWorkers,nShifts];
      for (int w = 0; w < nWorkers; ++w) {
        for (int s = 0; s < nShifts; ++s) {
          x[w,s] =
              model.AddVar(0, availability[w,s], pay[w], GRB.CONTINUOUS,
                           Workers[w] + "." + Shifts[s]);
        }
      }

      // The objective is to minimize the total pay costs
      model.Set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.Update();

      // Constraint: assign exactly shiftRequirements[s] workers
      // to each shift s
      for (int s = 0; s < nShifts; ++s) {
        GRBLinExpr lhs = 0.0;
        for (int w = 0; w < nWorkers; ++w)
          lhs += x[w, s];
        model.AddConstr(lhs == shiftRequirements[s], Shifts[s]);
      }

      // Optimize
      model.Optimize();
      int status = model.Get(GRB.IntAttr.Status);
      if (status == GRB.Status.UNBOUNDED) {
        Console.WriteLine("The model cannot be solved "
            + "because it is unbounded");
        return;
      }
      if (status == GRB.Status.OPTIMAL) {
        Console.WriteLine("The optimal objective is " +
            model.Get(GRB.DoubleAttr.ObjVal));
        return;
      }
      if ((status != GRB.Status.INF_OR_UNBD) &&
          (status != GRB.Status.INFEASIBLE)) {
        Console.WriteLine("Optimization was stopped with status " +
status);
        return;
      }

      // Do IIS
      Console.WriteLine("The model is infeasible; computing IIS");
      model.ComputeIIS();
      Console.WriteLine("\nThe following constraint(s) "
          + "cannot be satisfied:");
      foreach (GRBConstr c in model.GetConstrs()) {
        if (c.Get(GRB.IntAttr.IISConstr) == 1) {
          Console.WriteLine(c.Get(GRB.StringAttr.ConstrName));
        }
      }

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
          e.Message);
    }
  }
}
```

页面：workforce2_cs.cs

workforce2_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, use IIS iteratively
to
   find all conflicting constraints. */

using System;
using System.Collections.Generic;
using Gurobi;

class workforce2_cs
{
  static void Main()
  {
    try {

      // Sample data
      // Sets of days and workers
      string[] Shifts =
          new string[] { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
              "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12",
"Sat13",
              "Sun14" };
      string[] Workers =
          new string[] { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred",
"Gu" };

      int nShifts = Shifts.Length;
      int nWorkers = Workers.Length;

      // Number of workers required for each shift
      double[] shiftRequirements =
          new double[] { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

      // Amount each worker is paid to work one shift
      double[] pay = new double[] { 10, 12, 10, 8, 8, 9, 11 };

      // Worker availability: 0 if the worker is unavailable for a
shift
      double[,] availability =
          new double[,] { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
              { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
              { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
              { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
              { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.Set(GRB.StringAttr.ModelName, "assignment");

      // Assignment variables: x[w][s] == 1 if worker w is assigned
      // to shift s. Since an assignment model always produces integer
      // solutions, we use continuous variables and solve as an LP.
      GRBVar[,] x = new GRBVar[nWorkers,nShifts];
      for (int w = 0; w < nWorkers; ++w) {
        for (int s = 0; s < nShifts; ++s) {
          x[w,s] =
              model.AddVar(0, availability[w,s], pay[w], GRB.CONTINUOUS,
                           Workers[w] + "." + Shifts[s]);
        }
      }

      // The objective is to minimize the total pay costs
      model.Set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.Update();

      // Constraint: assign exactly shiftRequirements[s] workers
      // to each shift s
      for (int s = 0; s < nShifts; ++s) {
        GRBLinExpr lhs = 0.0;
        for (int w = 0; w < nWorkers; ++w)
          lhs += x[w, s];
        model.AddConstr(lhs == shiftRequirements[s], Shifts[s]);
      }

      // Optimize
      model.Optimize();
      int status = model.Get(GRB.IntAttr.Status);
      if (status == GRB.Status.UNBOUNDED) {
        Console.WriteLine("The model cannot be solved "
            + "because it is unbounded");
        return;
      }
      if (status == GRB.Status.OPTIMAL) {
        Console.WriteLine("The optimal objective is " +
            model.Get(GRB.DoubleAttr.ObjVal));
        return;
      }
      if ((status != GRB.Status.INF_OR_UNBD) &&
          (status != GRB.Status.INFEASIBLE)) {
        Console.WriteLine("Optimization was stopped with status " +
status);
        return;
      }

      // Do IIS
      Console.WriteLine("The model is infeasible; computing IIS");
      LinkedList<string> removed = new LinkedList<string>();

      // Loop until we reduce to a model that can be solved
      while (true) {
        model.ComputeIIS();
        Console.WriteLine("\nThe following constraint cannot be
satisfied:");
        foreach (GRBConstr c in model.GetConstrs()) {
          if (c.Get(GRB.IntAttr.IISConstr) == 1) {
            Console.WriteLine(c.Get(GRB.StringAttr.ConstrName));
            // Remove a single constraint from the model
            removed.AddFirst(c.Get(GRB.StringAttr.ConstrName));
            model.Remove(c);
            break;
          }
        }

        Console.WriteLine();
        model.Optimize();
        status = model.Get(GRB.IntAttr.Status);

        if (status == GRB.Status.UNBOUNDED) {
          Console.WriteLine("The model cannot be solved "
              + "because it is unbounded");
          return;
        }
        if (status == GRB.Status.OPTIMAL) {
          break;
        }
        if ((status != GRB.Status.INF_OR_UNBD) &&
            (status != GRB.Status.INFEASIBLE)) {
          Console.WriteLine("Optimization was stopped with status " +
              status);
          return;
        }
      }

      Console.WriteLine("\nThe following constraints were removed "
          + "to get a feasible LP:");
      foreach (string s in removed) {
        Console.Write(s + " ");
      }
      Console.WriteLine();

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
          e.Message);
    }
  }
}
```

页面：workforce3_cs.cs

workforce3_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. If the problem cannot be solved, add slack variables
   to determine which constraints cannot be satisfied, and how much
   they need to be relaxed. */

using System;
using System.Collections.Generic;
using Gurobi;

class workforce3_cs
{
  static void Main()
  {
    try {

      // Sample data
      // Sets of days and workers
      string[] Shifts =
          new string[] { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
              "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12",
"Sat13",
              "Sun14" };
      string[] Workers =
          new string[] { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred",
"Gu" };

      int nShifts = Shifts.Length;
      int nWorkers = Workers.Length;

      // Number of workers required for each shift
      double[] shiftRequirements =
          new double[] { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

      // Amount each worker is paid to work one shift
      double[] pay = new double[] { 10, 12, 10, 8, 8, 9, 11 };

      // Worker availability: 0 if the worker is unavailable for a
shift
      double[,] availability =
          new double[,] { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
              { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
              { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
              { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
              { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.Set(GRB.StringAttr.ModelName, "assignment");

      // Assignment variables: x[w][s] == 1 if worker w is assigned
      // to shift s. Since an assignment model always produces integer
      // solutions, we use continuous variables and solve as an LP.
      GRBVar[,] x = new GRBVar[nWorkers,nShifts];
      for (int w = 0; w < nWorkers; ++w) {
        for (int s = 0; s < nShifts; ++s) {
          x[w,s] =
              model.AddVar(0, availability[w,s], pay[w], GRB.CONTINUOUS,
                           Workers[w] + "." + Shifts[s]);
        }
      }

      // The objective is to minimize the total pay costs
      model.Set(GRB.IntAttr.ModelSense, 1);

      // Update model to integrate new variables
      model.Update();

      // Constraint: assign exactly shiftRequirements[s] workers
      // to each shift s
      LinkedList<GRBConstr> reqCts = new LinkedList<GRBConstr>();
      for (int s = 0; s < nShifts; ++s) {
        GRBLinExpr lhs = 0.0;
        for (int w = 0; w < nWorkers; ++w)
          lhs += x[w,s];
        GRBConstr newCt =
            model.AddConstr(lhs == shiftRequirements[s], Shifts[s]);
        reqCts.AddFirst(newCt);
      }

      // Optimize
      model.Optimize();
      int status = model.Get(GRB.IntAttr.Status);
      if (status == GRB.Status.UNBOUNDED) {
        Console.WriteLine("The model cannot be solved "
            + "because it is unbounded");
        return;
      }
      if (status == GRB.Status.OPTIMAL) {
        Console.WriteLine("The optimal objective is " +
            model.Get(GRB.DoubleAttr.ObjVal));
        return;
      }
      if ((status != GRB.Status.INF_OR_UNBD) &&
          (status != GRB.Status.INFEASIBLE)) {
        Console.WriteLine("Optimization was stopped with status " +
status);
        return;
      }

      // Add slack variables to make the model feasible
      Console.WriteLine("The model is infeasible; adding slack
variables");
```

```
      // Set original objective coefficients to zero
      model.SetObjective(new GRBLinExpr());

      // Add a new slack variable to each shift constraint so that the
shifts
      // can be satisfied
      LinkedList<GRBVar> slacks = new LinkedList<GRBVar>();
      foreach (GRBConstr c in reqCts) {
        GRBColumn col = new GRBColumn();
        col.AddTerm(1.0, c);
        GRBVar newvar =
            model.AddVar(0, GRB.INFINITY, 1.0, GRB.CONTINUOUS, col,
                         c.Get(GRB.StringAttr.ConstrName) + "Slack");
        slacks.AddFirst(newvar);
      }

      // Solve the model with slacks
      model.Optimize();
      status = model.Get(GRB.IntAttr.Status);
      if ((status == GRB.Status.INF_OR_UNBD) ||
          (status == GRB.Status.INFEASIBLE) ||
          (status == GRB.Status.UNBOUNDED)) {
        Console.WriteLine("The model with slacks cannot be solved "
            + "because it is infeasible or unbounded");
        return;
      }
      if (status != GRB.Status.OPTIMAL) {
        Console.WriteLine("Optimization was stopped with status " +
status);
        return;
      }

      Console.WriteLine("\nSlack values:");
      foreach (GRBVar sv in slacks) {
        if (sv.Get(GRB.DoubleAttr.X) > 1e-6) {
          Console.WriteLine(sv.Get(GRB.StringAttr.VarName) + " = " +
              sv.Get(GRB.DoubleAttr.X));
        }
      }

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
          e.Message);
    }
  }
}
```

页面：workforce4_cs.cs

workforce4_cs.cs

```
/* Copyright 2011, Gurobi Optimization, Inc. */

/* Assign workers to shifts; each worker may or may not be available on
a
   particular day. We use Pareto optimization to solve the model:
   first, we minimize the linear sum of the slacks. Then, we constrain
   the sum of the slacks, and we minimize a quadratic objective that
   tries to balance the workload among the workers. */

using System;
using Gurobi;

class workforce4_cs
{
  static void Main()
  {
    try {

      // Sample data
      // Sets of days and workers
      string[] Shifts =
          new string[] { "Mon1", "Tue2", "Wed3", "Thu4", "Fri5", "Sat6",
              "Sun7", "Mon8", "Tue9", "Wed10", "Thu11", "Fri12",
"Sat13",
              "Sun14" };
      string[] Workers =
          new string[] { "Amy", "Bob", "Cathy", "Dan", "Ed", "Fred",
"Gu" };

      int nShifts = Shifts.Length;
      int nWorkers = Workers.Length;

      // Number of workers required for each shift
      double[] shiftRequirements =
          new double[] { 3, 2, 4, 4, 5, 6, 5, 2, 2, 3, 4, 6, 7, 5 };

      // Worker availability: 0 if the worker is unavailable for a
shift
      double[,] availability =
          new double[,] { { 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0 },
              { 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
              { 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1 },
              { 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1 },
              { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1 },
              { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

      // Model
      GRBEnv env = new GRBEnv();
      GRBModel model = new GRBModel(env);
      model.Set(GRB.StringAttr.ModelName, "assignment");
      // Assignment variables: x[w][s] == 1 if worker w is assigned
      // to shift s. This is no longer a pure assignment model, so we
must
      // use binary variables.
      GRBVar[,] x = new GRBVar[nWorkers, nShifts];
      for (int w = 0; w < nWorkers; ++w) {
        for (int s = 0; s < nShifts; ++s) {
          x[w,s] =
              model.AddVar(0, availability[w,s], 0, GRB.BINARY,
                           Workers[w] + "." + Shifts[s]);
        }
      }

      // Slack variables for each shift constraint so that the shifts
can
      // be satisfied
      GRBVar[] slacks = new GRBVar[nShifts];
      for (int s = 0; s < nShifts; ++s) {
        slacks[s] =
          model.AddVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS,
                       Shifts[s] + "Slack");
      }

      // Variable to represent the total slack
      GRBVar totSlack = model.AddVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS,
                                     "totSlack");

      // Variables to count the total shifts worked by each worker
      GRBVar[] totShifts = new GRBVar[nWorkers];
      for (int w = 0; w < nWorkers; ++w) {
        totShifts[w] = model.AddVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS,
                                    Workers[w] + "TotShifts");
      }

      // Update model to integrate new variables
      model.Update();

      GRBLinExpr lhs;

      // Constraint: assign exactly shiftRequirements[s] workers
      // to each shift s, plus the slack
      for (int s = 0; s < nShifts; ++s) {
        lhs = new GRBLinExpr();
        lhs += slacks[s];
        for (int w = 0; w < nWorkers; ++w) {
          lhs += x[w, s];
        }
        model.AddConstr(lhs == shiftRequirements[s], Shifts[s]);
      }

      // Constraint: set totSlack equal to the total slack
      lhs = new GRBLinExpr();
      for (int s = 0; s < nShifts; ++s) {
        lhs += slacks[s];
      }
      model.AddConstr(lhs == totSlack, "totSlack");
      // Constraint: compute the total number of shifts for each worker
      for (int w = 0; w < nWorkers; ++w) {
        lhs = new GRBLinExpr();
        for (int s = 0; s < nShifts; ++s) {
          lhs += x[w, s];
        }
        model.AddConstr(lhs == totShifts[w], "totShifts" + Workers[w]);
      }

      // Objective: minimize the total slack
      GRBLinExpr obj = new GRBLinExpr();
      obj += totSlack;
      model.SetObjective(obj);

      // Optimize
      int status = solveAndPrint(model, totSlack, nWorkers, Workers,
totShifts);
      if (status != GRB.Status.OPTIMAL) {
        return;
      }

      // Constrain the slack by setting its upper and lower bounds
      totSlack.Set(GRB.DoubleAttr.UB, totSlack.Get(GRB.DoubleAttr.X));
      totSlack.Set(GRB.DoubleAttr.LB, totSlack.Get(GRB.DoubleAttr.X));

      // Variable to count the average number of shifts worked
      GRBVar avgShifts =
        model.AddVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS, "avgShifts");

      // Variables to count the difference from average for each worker;
      // note that these variables can take negative values.
      GRBVar[] diffShifts = new GRBVar[nWorkers];
      for (int w = 0; w < nWorkers; ++w) {
        diffShifts[w] = model.AddVar(-GRB.INFINITY, GRB.INFINITY, 0,
                                     GRB.CONTINUOUS, Workers[w] +
"Diff");
      }

      // Update model to integrate new variables
      model.Update();

      // Constraint: compute the average number of shifts worked
      lhs = new GRBLinExpr();
      for (int w = 0; w < nWorkers; ++w) {
        lhs.AddTerm(1.0, totShifts[w]);
      }
      model.AddConstr(lhs == nWorkers * avgShifts, "avgShifts");

      // Constraint: compute the difference from the average number of
shifts
      for (int w = 0; w < nWorkers; ++w) {
        lhs = new GRBLinExpr();
        lhs += totShifts[w];
        lhs -= avgShifts;
        model.AddConstr(lhs == diffShifts[w], Workers[w] + "Diff");
      }
      // Objective: minimize the sum of the square of the difference
from the
      // average number of shifts worked
      GRBQuadExpr qobj = new GRBQuadExpr();
      for (int w = 0; w < nWorkers; ++w) {
        qobj += diffShifts[w] * diffShifts[w];
      }
      model.SetObjective(qobj);

      // Optimize
      status = solveAndPrint(model, totSlack, nWorkers, Workers,
totShifts);
      if (status != GRB.Status.OPTIMAL) {
        return;
      }

      // Dispose of model and env
      model.Dispose();
      env.Dispose();

    } catch (GRBException e) {
      Console.WriteLine("Error code: " + e.ErrorCode + ". " +
          e.Message);
    }
  }

  private static int solveAndPrint(GRBModel model, GRBVar totSlack,
                                   int nWorkers, String[] Workers,
                                   GRBVar[] totShifts)
  {
    model.Optimize();
    int status = model.Get(GRB.IntAttr.Status);
    if ((status == GRB.Status.INF_OR_UNBD) ||
        (status == GRB.Status.INFEASIBLE) ||
        (status == GRB.Status.UNBOUNDED)) {
      Console.WriteLine("The model cannot be solved "
          + "because it is infeasible or unbounded");
      return status;
    }
    if (status != GRB.Status.OPTIMAL) {
      Console.WriteLine("Optimization was stopped with status " +
status);
      return status;
    }

    // Print total slack and the number of shifts worked for each
worker
    Console.WriteLine("\nTotal slack required: " +
                      totSlack.Get(GRB.DoubleAttr.X));
    for (int w = 0; w < nWorkers; ++w) {
      Console.WriteLine(Workers[w] + " worked " +
                        totShifts[w].Get(GRB.DoubleAttr.X) + " shifts");
    }
    Console.WriteLine("\n");
    return status;
  }
}
```

页面：Visual Basic Examples

Visual Basic 示例

本节包含了所有 Gurobi Visual Basic 的示例源代码。相同的源代码也可以在 Gurobi 发布包的 *examples/vb* 目录中找到。

小节

页面：callback_vb.vb

callback_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' This example reads an LP or a MIP from a file, sets a callback
' to monitor the optimization progress and to output some progress
' information to the screen and to a log file. If it is a MIP and 10%
' gap is reached, then it aborts.

Imports System
Imports Gurobi

Class callback_vb
    Inherits GRBCallback
    Private vars As GRBVar()
    Private lastmsg As Integer

    Public Sub New(ByVal xvars As GRBVar())
        vars = xvars
        lastmsg = -100
    End Sub

    Protected Overloads Overrides Sub Callback()
        Try
            If where = GRB.Callback.PRESOLVE Then
                Dim cdels As Integer =
GetIntInfo(GRB.Callback.PRE_COLDEL)
                Dim rdels As Integer =
GetIntInfo(GRB.Callback.PRE_ROWDEL)
                Console.WriteLine(cdels & " columns and " & rdels & "
rows are removed")
            ElseIf where = GRB.Callback.SIMPLEX Then
                Dim itcnt As Integer =
CInt(GetDoubleInfo(GRB.Callback.SPX_ITRCNT))
                If itcnt Mod 100 = 0 Then
                    Dim obj As Double =
GetDoubleInfo(GRB.Callback.SPX_OBJVAL)
                    Dim pinf As Double =
GetDoubleInfo(GRB.Callback.SPX_PRIMINF)
                    Dim dinf As Double =
GetDoubleInfo(GRB.Callback.SPX_DUALINF)
                    Dim ispert As Integer =
GetIntInfo(GRB.Callback.SPX_ISPERT)
                    Dim ch As Char
                    If ispert = 0 Then
                        ch = " "c
                    ElseIf ispert = 1 Then
                        ch = "S"c
                    Else
                        ch = "P"c
                    End If
                    Console.WriteLine(itcnt & "  " & obj & ch & "  " &
pinf & "  " & dinf)
                End If
            ElseIf where = GRB.Callback.MIP Then
                Dim nodecnt As Integer =
CInt(GetDoubleInfo(GRB.Callback.MIP_NODCNT))
                If nodecnt - lastmsg >= 100 Then
                    lastmsg = nodecnt
                    Dim objbst As Double =
GetDoubleInfo(GRB.Callback.MIP_OBJBST)
                    Dim objbnd As Double =
GetDoubleInfo(GRB.Callback.MIP_OBJBND)
                    If Math.Abs(objbst - objbnd) < 0.1 * (1.0R +
Math.Abs(objbst)) Then
                        Abort()
                    End If
                    Dim actnodes As Integer =
CInt(GetDoubleInfo(GRB.Callback.MIP_NODLFT))
                    Dim itcnt As Integer =
CInt(GetDoubleInfo(GRB.Callback.MIP_ITRCNT))
                    Dim solcnt As Integer =
GetIntInfo(GRB.Callback.MIP_SOLCNT)
                    Dim cutcnt As Integer =
GetIntInfo(GRB.Callback.MIP_CUTCNT)
                    Console.WriteLine(nodecnt & " " & actnodes & " " &
itcnt & " " & _
                                      objbst & " " & objbnd & " " &
solcnt & " " & cutcnt)
                End If
            ElseIf where = GRB.Callback.MIPSOL Then
                Dim obj As Double =
GetDoubleInfo(GRB.Callback.MIPSOL_OBJ)
                Dim nodecnt As Integer =
CInt(GetDoubleInfo(GRB.Callback.MIPSOL_NODCNT))
                Dim x As Double() = GetSolution(vars)
                Console.WriteLine("**** New solution at node " &
nodecnt & ", obj " & _
                                  obj & ", x(0) = " & x(0) & "****")
            End If
        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
            Console.WriteLine(e.StackTrace)
        End Try
    End Sub

    Shared Sub Main(ByVal args As String())

        If args.Length < 1 Then
            Console.WriteLine("Usage: callback_vb filename")
            Return
        End If

        Try
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env, args(0))

            Dim vars As GRBVar() = model.GetVars()

            model.SetCallback(New callback_vb(vars))
            model.Optimize()

            Dim x As Double() = model.Get(GRB.DoubleAttr.X, vars)
            Dim vnames As String() = model.Get(GRB.StringAttr.VarName,
vars)

            For j As Integer = 0 To vars.Length - 1
                If x(j) <> 0.0R Then
                    Console.WriteLine(vnames(j) & " " & x(j))
                End If
            Next

            For j As Integer = 0 To vars.Length - 1
                If x(j) <> 0.0R Then
                    Console.WriteLine(vnames(j) & " " & x(j))
                End If
            Next

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
            Console.WriteLine(e.StackTrace)
        End Try
    End Sub
End Class
```

页面：diet_vb.vb

diet_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.

' Solve the classic diet model, showing how to add constraints
' to an existing model.

Imports System
Imports Gurobi

Class diet_vb
    Shared Sub Main()
        Try

            ' Nutrition guidelines, based on
            ' USDA Dietary Guidelines for Americans, 2005
            ' http://www.health.gov/DietaryGuidelines/dga2005/
            Dim Categories As String() = New String() {"calories",
"protein", "fat", "sodium"}
            Dim nCategories As Integer = Categories.Length
            Dim minNutrition As Double() = New Double() {1800, 91, 0, 0}
            Dim maxNutrition As Double() = New Double() {2200,
GRB.INFINITY, 65, 1779}

            ' Set of foods
            Dim Foods As String() = New String() {"hamburger",
"chicken", "hot dog", "fries", _
                                                  "macaroni", "pizza",
"salad", "milk", _
                                                  "ice cream"}
            Dim nFoods As Integer = Foods.Length
            Dim cost As Double() = New Double() {2.49, 2.89, 1.5R, 1.89,
2.09, 1.99, _
                                                 2.49, 0.89, 1.59}

            ' Nutrition values for the foods
            ' hamburger
            ' chicken
            ' hot dog
            ' fries
            ' macaroni
            ' pizza
            ' salad
            ' milk
            ' ice cream
            Dim nutritionValues As Double(,) = New Double(,) {{410, 24,
26, 730}, _
                                                              {420, 32,
10, 1190}, _
                                                              {560, 20,
32, 1800}, _
                                                              {380, 4,
19, 270}, _
```

```
                                                              {320, 12,
10, 930}, _
                                                              {320, 15,
12, 820}, _
                                                              {320, 31,
12, 1230}, _
                                                              {100, 8,
2.5, 125}, _
                                                              {330, 8,
10, 180}}

            ' Model
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env)
            model.Set(GRB.StringAttr.ModelName, "diet")

            ' Create decision variables for the nutrition information,
            ' which we limit via bounds
            Dim nutrition As GRBVar() = New GRBVar(nCategories - 1) {}
            For i As Integer = 0 To nCategories - 1
                nutrition(i) = model.AddVar(minNutrition(i),
maxNutrition(i), 0, _
                                            GRB.CONTINUOUS,
Categories(i))
            Next

            ' Create decision variables for the foods to buy
            Dim buy As GRBVar() = New GRBVar(nFoods - 1) {}
            For j As Integer = 0 To nFoods - 1
                buy(j) = model.AddVar(0, GRB.INFINITY, cost(j),
GRB.CONTINUOUS, Foods(j))
            Next

            ' The objective is to minimize the costs
            model.Set(GRB.IntAttr.ModelSense, 1)

            ' Update model to integrate new variables
            model.Update()

            ' Nutrition constraints
            For i As Integer = 0 To nCategories - 1
                Dim ntot As GRBLinExpr = 0
                For j As Integer = 0 To nFoods - 1
                    ntot += nutritionValues(j, i) * buy(j)
                Next
                model.AddConstr(ntot = nutrition(i), Categories(i))
            Next

            ' Use barrier to solve model
            model.GetEnv().Set(GRB.IntParam.Method, GRB.METHOD_BARRIER)

            ' Solve
            model.Optimize()
            PrintSolution(model, buy, nutrition)

            Console.WriteLine(vbLf & "Adding constraint: at most 6
servings of dairy")
```

```
            model.AddConstr(buy(7) + buy(8) <= 6, "limit_dairy")

            ' Solve
            model.Optimize()

            PrintSolution(model, buy, nutrition)

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub

    Private Shared Sub PrintSolution(ByVal model As GRBModel, ByVal buy
As GRBVar(), ByVal nutrition As GRBVar())
        If model.Get(GRB.IntAttr.Status) = GRB.Status.OPTIMAL Then
            Console.WriteLine(vbLf & "Cost: " &
model.Get(GRB.DoubleAttr.ObjVal))
            Console.WriteLine(vbLf & "Buy:")
            For j As Integer = 0 To buy.Length - 1
                If buy(j).Get(GRB.DoubleAttr.X) > 0.0001 Then
                    Console.WriteLine(buy(j).Get(GRB.StringAttr.VarName)
& " " & _
                                      buy(j).Get(GRB.DoubleAttr.X))
                End If
            Next
            Console.WriteLine(vbLf & "Nutrition:")
            For i As Integer = 0 To nutrition.Length - 1

Console.WriteLine(nutrition(i).Get(GRB.StringAttr.VarName) & " " & _
                                  nutrition(i).Get(GRB.DoubleAttr.X))
            Next
        Else
            Console.WriteLine("No solution")
        End If
    End Sub
End Class
```

页面：facility_vb.vb

facility_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Facility location: a company currently ships its product from 5
plants
' to 4 warehouses. It is considering closing some plants to reduce
' costs. What plant(s) should the company close, in order to minimize
' transportation and fixed costs?
'
' Based on an example from Frontline Systems:
' http://www.solver.com/disfacility.htm
' Used with permission.

Imports System
Imports Gurobi

Class facility_vb
    Shared Sub Main()
        Try

            ' Warehouse demand in thousands of units
            Dim Demand As Double() = New Double() {15, 18, 14, 20}

            ' Plant capacity in thousands of units
            Dim Capacity As Double() = New Double() {20, 22, 17, 19, 18}

            ' Fixed costs for each plant
            Dim FixedCosts As Double() = New Double() {12000, 15000,
17000, 13000, 16000}

            ' Transportation costs per thousand units
            Dim TransCosts As Double(,) = New Double(,) {{4000, 2000,
3000, 2500, 4500}, _
                            {2500, 2600,
3400, 3000, 4000}, _
                            {1200, 1800,
2600, 4100, 3000}, _
                            {2200, 2600,
3100, 3700, 3200}}

            ' Number of plants and warehouses
            Dim nPlants As Integer = Capacity.Length
            Dim nWarehouses As Integer = Demand.Length

            ' Model
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env)
            model.Set(GRB.StringAttr.ModelName, "facility")

            ' Plant open decision variables: open(p) == 1 if plant p is
open.
            Dim open As GRBVar() = New GRBVar(nPlants - 1) {}
            For p As Integer = 0 To nPlants - 1
```

```
                open(p) = model.AddVar(0, 1, FixedCosts(p), GRB.BINARY,
"Open" & p)
            Next

            ' Transportation decision variables: how much to transport
from
            ' a plant p to a warehouse w
            Dim transport As GRBVar(,) = New GRBVar(nWarehouses - 1,
nPlants - 1) {}
            For w As Integer = 0 To nWarehouses - 1
                For p As Integer = 0 To nPlants - 1
                    transport(w, p) = model.AddVar(0, GRB.INFINITY,
TransCosts(w, p), _
                                                   GRB.CONTINUOUS,
"Trans" & p & "." & w)
                Next
            Next

            ' The objective is to minimize the total fixed and variable
costs
            model.Set(GRB.IntAttr.ModelSense, 1)

            ' Update model to integrate new variables
            model.Update()

            ' Production constraints
            ' Note that the right-hand limit sets the production to
zero if
            ' the plant is closed
            For p As Integer = 0 To nPlants - 1
                Dim ptot As GRBLinExpr = 0
                For w As Integer = 0 To nWarehouses - 1
                    ptot += transport(w, p)
                Next
                model.AddConstr(ptot <= Capacity(p) * open(p),
"Capacity" & p)
            Next

            ' Demand constraints
            For w As Integer = 0 To nWarehouses - 1
                Dim dtot As GRBLinExpr = 0
                For p As Integer = 0 To nPlants - 1
                    dtot += transport(w, p)
                Next
                model.AddConstr(dtot = Demand(w), "Demand" & w)
            Next

            ' Guess at the starting point: close the plant with the
highest
            ' fixed costs; open all others

            ' First, open all plants
            For p As Integer = 0 To nPlants - 1
                open(p).Set(GRB.DoubleAttr.Start, 1.0)
            Next

            ' Now close the plant with the highest fixed cost
            Console.WriteLine("Initial guess:")
            Dim maxFixed As Double = -GRB.INFINITY
            For p As Integer = 0 To nPlants - 1
                If FixedCosts(p) > maxFixed Then
                    maxFixed = FixedCosts(p)
                End If
            Next
            For p As Integer = 0 To nPlants - 1
                If FixedCosts(p) = maxFixed Then
                    open(p).Set(GRB.DoubleAttr.Start, 0.0)
                    Console.WriteLine("Closing plant " & p & vbLf)
                    Exit For
                End If
            Next

            ' Use barrier to solve root relaxation
            model.GetEnv().Set(GRB.IntParam.Method, GRB.METHOD_BARRIER)

            ' Solve
            model.Optimize()

            ' Print solution
            Console.WriteLine(vbLf & "TOTAL COSTS: " &
model.Get(GRB.DoubleAttr.ObjVal))
            Console.WriteLine("SOLUTION:")
            For p As Integer = 0 To nPlants - 1
                If open(p).Get(GRB.DoubleAttr.X) = 1.0 Then
                    Console.WriteLine("Plant " & p & " open:")
                    For w As Integer = 0 To nWarehouses - 1
                        If transport(w, p).Get(GRB.DoubleAttr.X) >
0.0001 Then
                            Console.WriteLine("  Transport " &
transport(w, p).Get(GRB.DoubleAttr.X) & _
                                                  " units to warehouse " &
w)
                        End If
                    Next
                Else
                    Console.WriteLine("Plant " & p & " closed!")
                End If

            Next

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：feasopt_vb.vb

feasopt_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' This example reads a MIP model from a file, adds artificial
' variables to each constraint, and then minimizes the sum of the
' artificial variables.  A solution with objective zero corresponds
' to a feasible solution to the input model.

Imports Gurobi
Imports System

Class feasopt_vb
    Shared Sub Main(ByVal args As String())

        If args.Length < 1 Then
            Console.WriteLine("Usage: feasopt_vb filename")
            Return
        End If

        Try
            Dim env As New GRBEnv()
            Dim feasmodel As New GRBModel(env, args(0))

            ' Clear objective
            feasmodel.SetObjective(New GRBLinExpr())

            ' Add slack variables
            Dim c As GRBConstr() = feasmodel.GetConstrs()
            For i As Integer = 0 To c.Length - 1
                Dim sense As Char = c(i).Get(GRB.CharAttr.Sense)
                If sense <> ">"c Then
                    Dim constrs As GRBConstr() = New GRBConstr() {c(i)}
                    Dim coeffs As Double() = New Double() {-1}
                    feasmodel.AddVar(0.0, GRB.INFINITY, 1.0,
GRB.CONTINUOUS, constrs, coeffs, _
                     "ArtN_" & c(i).Get(GRB.StringAttr.ConstrName))
                End If
                If sense <> "<"c Then
                    Dim constrs As GRBConstr() = New GRBConstr() {c(i)}
                    Dim coeffs As Double() = New Double() {1}
                    feasmodel.AddVar(0.0, GRB.INFINITY, 1.0,
GRB.CONTINUOUS, constrs, coeffs, _
                     "ArtP_" & c(i).Get(GRB.StringAttr.ConstrName))
                End If
            Next
            feasmodel.Update()

            ' Optimize modified model
            feasmodel.Write("feasopt.lp")
            feasmodel.Optimize()

            ' Dispose of model and env
            feasmodel.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：fixanddive_vb.vb

fixanddive_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Implement a simple MIP heuristic.  Relax the model,
' sort variables based on fractionality, and fix the 25% of
' the fractional variables that are closest to integer variables.
' Repeat until either the relaxation is integer feasible or
' linearly infeasible.

Imports System
Imports System.Collections.Generic
Imports Gurobi

Class fixanddive_vb
    ' Comparison class used to sort variable list based on relaxation
    ' fractionality

    Private Class FractionalCompare : Implements IComparer(Of GRBVar)
        Public Function Compare(ByVal v1 As GRBVar, ByVal v2 As GRBVar)
As Integer _
                             Implements IComparer(Of
Gurobi.GRBVar).Compare
            Try
                Dim sol1 As Double = Math.Abs(v1.Get(GRB.DoubleAttr.X))
                Dim sol2 As Double = Math.Abs(v2.Get(GRB.DoubleAttr.X))
                Dim frac1 As Double = Math.Abs(sol1 - Math.Floor(sol1 +
0.5))
                Dim frac2 As Double = Math.Abs(sol2 - Math.Floor(sol2 +
0.5))
                If frac1 < frac2 Then
                    Return -1
                ElseIf frac1 > frac2 Then
                    Return 1
                Else
                    Return 0
                End If
            Catch e As GRBException
                Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
            End Try
            Return 0
        End Function
    End Class

    Shared Sub Main(ByVal args As String())

        If args.Length < 1 Then
            Console.WriteLine("Usage: fixanddive_vb filename")
            Return
        End If

        Try
            ' Read model
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env, args(0))

            ' Collect integer variables and relax them
            Dim intvars As New List(Of GRBVar)()
            For Each v As GRBVar In model.GetVars()
                If v.Get(GRB.CharAttr.VType) <> GRB.CONTINUOUS Then
                    intvars.Add(v)
                    v.Set(GRB.CharAttr.VType, GRB.CONTINUOUS)
                End If
            Next

            model.GetEnv().Set(GRB.IntParam.OutputFlag, 0)
            model.Optimize()

            ' Perform multiple iterations. In each iteration, identify
the first
            ' quartile of integer variables that are closest to an
integer value
            ' in the relaxation, fix them to the nearest integer, and
repeat.

            For iter As Integer = 0 To 999

                ' create a list of fractional variables, sorted in
order of
                ' increasing distance from the relaxation solution to
the nearest
                ' integer value

                Dim fractional As New List(Of GRBVar)()
                For Each v As GRBVar In intvars
                    Dim sol As Double =
Math.Abs(v.Get(GRB.DoubleAttr.X))
                    If Math.Abs(sol - Math.Floor(sol + 0.5)) > 0.00001
Then
                        fractional.Add(v)
                    End If
                Next

                Console.WriteLine("Iteration " & iter & ", obj " &
model.Get(GRB.DoubleAttr.ObjVal) & _
                                  ", fractional " & fractional.Count)

                If fractional.Count = 0 Then
                    Console.WriteLine("Found feasible solution -
objective " & _
                                      model.Get(GRB.DoubleAttr.ObjVal))
                    Exit For
                End If

                ' Fix the first quartile to the nearest integer value

                fractional.Sort(New FractionalCompare())
                Dim nfix As Integer = Math.Max(fractional.Count / 4, 1)
                For i As Integer = 0 To nfix - 1
                    Dim v As GRBVar = fractional(i)
                    Dim fixval As Double =
Math.Floor(v.Get(GRB.DoubleAttr.X) + 0.5)
                    v.Set(GRB.DoubleAttr.LB, fixval)
                    v.Set(GRB.DoubleAttr.UB, fixval)
                    Console.WriteLine("  Fix " &
v.Get(GRB.StringAttr.VarName) & " to " & _
                                      fixval & " ( rel " &
v.Get(GRB.DoubleAttr.X) & " )")
                Next

                model.Optimize()

                ' Check optimization result

                If model.Get(GRB.IntAttr.Status) <> GRB.Status.OPTIMAL
Then
                    Console.WriteLine("Relaxation is infeasible")
                    Exit For
                End If
            Next

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " +
e.Message)
        End Try
    End Sub
End Class
```

页面：lp_vb.vb

lp_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' This example reads an LP model from a file and solves it.
' If the model is infeasible or unbounded, the example turns off
' presolve and solves the model again. If the model is infeasible,
' the example computes an Irreducible Infeasible Subsystem (IIS),
' and writes it to a file.

Imports System
Imports Gurobi

Class lp_vb
    Shared Sub Main(ByVal args As String())

        If args.Length < 1 Then
            Console.WriteLine("Usage: lp_vb filename")
            Return
        End If

        Try
            Dim env As GRBEnv = New GRBEnv("lp1.log")
            Dim model As GRBModel = New GRBModel(env, args(0))

            model.Optimize()

            Dim optimstatus As Integer = model.Get(GRB.IntAttr.Status)

            If optimstatus = GRB.Status.INF_OR_UNBD Then
                model.GetEnv().Set(GRB.IntParam.Presolve, 0)
                model.Optimize()
                optimstatus = model.Get(GRB.IntAttr.Status)
            End If

            If optimstatus = GRB.Status.OPTIMAL Then
                Dim objval As Double = model.Get(GRB.DoubleAttr.ObjVal)
                Console.WriteLine("Optimal objective: " & objval)
            ElseIf optimstatus = GRB.Status.INFEASIBLE Then
                Console.WriteLine("Model is infeasible")
                model.ComputeIIS()
                model.Write("model.ilp")
            ElseIf optimstatus = GRB.Status.UNBOUNDED Then
                Console.WriteLine("Model is unbounded")
            Else
                Console.WriteLine("Optimization was stopped with status
= " & optimstatus)
            End If

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
```

```
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：lpmethod_vb.vb

lpmethod_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Solve a model with different values of the Method parameter;
' show which value gives the shortest solve time.

Imports System
Imports Gurobi

Class lpmethod_vb

    Shared Sub Main(ByVal args As String())

        If args.Length < 1 Then
            Console.WriteLine("Usage: lpmethod_vb filename")
            Return
        End If

        Try
            ' Read model and verify that it is a MIP
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env, args(0))
            Dim menv As GRBEnv = model.GetEnv()

            ' Solve the model with different values of Method
            Dim bestMethod As Integer = -1
            Dim bestTime As Double = menv.get(GRB.DoubleParam.TimeLimit)
            For i As Integer = 0 To 2
                model.Reset()
                menv.Set(GRB.IntParam.Method, i)
                model.Optimize()
                If model.Get(GRB.IntAttr.Status) = GRB.Status.OPTIMAL
Then
                    bestTime = model.Get(GRB.DoubleAttr.Runtime)
                    bestMethod = i
                    ' Reduce the TimeLimit parameter to save time
                    ' with other methods
                    menv.Set(GRB.DoubleParam.TimeLimit, bestTime)
                End If
            Next

            ' Report which method was fastest
            If bestMethod = -1 Then
                Console.WriteLine("Unable to solve this model")
            Else
                Console.WriteLine("Solved in " & bestTime & _
                                  " seconds with Method: " & bestMethod)
            End If

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()
        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：lpmod_vb.vb

lpmod_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' This example reads an LP model from a file and solves it.
' If the model can be solved, then it finds the smallest positive
variable,
' sets its upper bound to zero, and resolves the model two ways:
' first with an advanced start, then without an advanced start
' (i.e. from scratch).

Imports System
Imports Gurobi

Class lpmod_vb
    Shared Sub Main(ByVal args As String())

        If args.Length < 1 Then
            Console.WriteLine("Usage: lpmod_vb filename")
            Return
        End If

        Try
            ' Read model and determine whether it is an LP
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env, args(0))
            If model.Get(GRB.IntAttr.IsMIP) <> 0 Then
                Console.WriteLine("The model is not a linear program")
                Environment.Exit(1)
            End If

            model.Optimize()

            Dim status As Integer = model.Get(GRB.IntAttr.Status)

            If (status = GRB.Status.INF_OR_UNBD) OrElse _
               (status = GRB.Status.INFEASIBLE) OrElse _
               (status = GRB.Status.UNBOUNDED) Then
                Console.WriteLine("The model cannot be solved because
it is " & _
                                  "infeasible or unbounded")
                Environment.Exit(1)
            End If

            If status <> GRB.Status.OPTIMAL Then
                Console.WriteLine("Optimization was stopped with status
" & status)
                Environment.Exit(0)
            End If

            ' Find the smallest variable value
            Dim minVal As Double = GRB.INFINITY
            Dim minVar As GRBVar = Nothing
            For Each v As GRBVar In model.GetVars()
```

```
                Dim sol As Double = v.Get(GRB.DoubleAttr.X)
                If (sol > 0.0001) AndAlso _
                   (sol < minVal) AndAlso _
                   (v.Get(GRB.DoubleAttr.LB) = 0.0) Then
                    minVal = sol
                    minVar = v
                End If
            Next

            Console.WriteLine(vbLf & "*** Setting " &
minVar.Get(GRB.StringAttr.VarName) & _
                              " from " & minVal & " to zero ***" & vbLf)
            minVar.Set(GRB.DoubleAttr.UB, 0)

            ' Solve from this starting point
            model.Optimize()

            ' Save iteration & time info
            Dim warmCount As Double =
model.Get(GRB.DoubleAttr.IterCount)
            Dim warmTime As Double = model.Get(GRB.DoubleAttr.Runtime)

            ' Reset the model and resolve
            Console.WriteLine(vbLf & "*** Resetting and solving " &
"without an advanced start ***" & vbLf)
            model.Reset()
            model.Optimize()

            Dim coldCount As Double =
model.Get(GRB.DoubleAttr.IterCount)
            Dim coldTime As Double = model.Get(GRB.DoubleAttr.Runtime)

            Console.WriteLine(vbLf & "*** Warm start: " & warmCount & "
iterations, " & warmTime & " seconds")

            Console.WriteLine("*** Cold start: " & coldCount & "
iterations, " & coldTime & " seconds")

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：mip1_vb.vb

mip1_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' This example formulates and solves the following simple MIP model:
'
'    maximize    x +   y + 2 z
'    subject to  x + 2 y + 3 z <= 4
'                x +   y       >= 1
'    x, y, z binary

Imports System
Imports Gurobi

Class mip1_vb
    Shared Sub Main()
        Try
            Dim env As GRBEnv = New GRBEnv("mip1.log")
            Dim model As GRBModel = New GRBModel(env)

            ' Create variables

            Dim x As GRBVar = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY,
"x")
            Dim y As GRBVar = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY,
"y")
            Dim z As GRBVar = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY,
"z")

            ' Integrate new variables

            model.Update()

            ' Set objective: maximize x + y + 2 z

            model.SetObjective(x + y + 2 * z, GRB.MAXIMIZE)

            ' Add constraint: x + 2 y + 3 z <= 4

            model.AddConstr(x + 2 * y + 3 * z <= 4.0, "c0")

            ' Add constraint: x + y >= 1

            model.AddConstr(x + 2 * y + 3 * z <= 4.0, "c1")

            ' Optimize model

            model.Optimize()

            Console.WriteLine(x.Get(GRB.StringAttr.VarName) & " " &
x.Get(GRB.DoubleAttr.X))
            Console.WriteLine(y.Get(GRB.StringAttr.VarName) & " " &
y.Get(GRB.DoubleAttr.X))
```

```
            Console.WriteLine(z.Get(GRB.StringAttr.VarName) & " " &
z.Get(GRB.DoubleAttr.X))

            Console.WriteLine("Obj: " &
model.Get(GRB.DoubleAttr.ObjVal))

            ' Dispose of model and env

            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：mip2_vb.vb

mip2_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' This example reads a MIP model from a file, solves it and
' prints the objective values from all feasible solutions
' generated while solving the MIP. Then it creates the fixed
' model and solves that model.


Imports System
Imports Gurobi

Class mip2_vb
    Shared Sub Main(ByVal args As String())

        If args.Length < 1 Then
            Console.WriteLine("Usage: mip2_vb filename")
            Return
        End If

        Try
            Dim env As GRBEnv = New GRBEnv("lp1.log")
            Dim model As GRBModel = New GRBModel(env, args(0))

            If model.Get(GRB.IntAttr.IsMIP) = 0 Then
                Console.WriteLine("Model is not a MIP")
                Return
            End If

            model.Optimize()

            Dim optimstatus As Integer = model.Get(GRB.IntAttr.Status)

            If optimstatus = GRB.Status.INF_OR_UNBD Then
                model.GetEnv().Set(GRB.IntParam.Presolve, 0)
                model.Optimize()
                optimstatus = model.Get(GRB.IntAttr.Status)
            End If

            Dim objval As Double

            If optimstatus = GRB.Status.OPTIMAL Then
                objval = model.Get(GRB.DoubleAttr.ObjVal)
                Console.WriteLine("Optimal objective: " & objval)
            ElseIf optimstatus = GRB.Status.INFEASIBLE Then
                Console.WriteLine("Model is infeasible")
                model.ComputeIIS()
                model.Write("model.ilp")
                Return
            ElseIf optimstatus = GRB.Status.UNBOUNDED Then
                Console.WriteLine("Model is unbounded")
                Return
            Else
```

```
                Console.WriteLine("Optimization was stopped with status
= " & optimstatus)
                Return
            End If

            ' Iterate over the solutions and compute the objectives
            Dim vars() As GRBVar = model.GetVars()
            model.GetEnv().Set(GRB.IntParam.OutputFlag, 0)

            Console.WriteLine()
            For k As Integer = 0 To model.Get(GRB.IntAttr.SolCount) - 1
                model.GetEnv().Set(GRB.IntParam.SolutionNumber, k)
                Dim objn As Double = 0.0

                For j As Integer = 0 To vars.Length - 1
                    objn += vars(j).Get(GRB.DoubleAttr.Obj) * _
                        vars(j).Get(GRB.DoubleAttr.Xn)
                Next

                Console.WriteLine("Solution " & k & " has objective: "
& objn)
            Next
            Console.WriteLine()
            model.GetEnv().Set(GRB.IntParam.OutputFlag, 1)

            ' Solve fixed model
            Dim fixedmodel As GRBModel = model.FixedModel()
            fixedmodel.GetEnv().Set(GRB.IntParam.Presolve, 0)
            fixedmodel.Optimize()

            Dim foptimstatus As Integer =
fixedmodel.Get(GRB.IntAttr.Status)
            If foptimstatus <> GRB.Status.OPTIMAL Then
                Console.WriteLine("Error: fixed model isn't optimal")
                Return
            End If

            Dim fobjval As Double =
fixedmodel.Get(GRB.DoubleAttr.ObjVal)

            If Math.Abs(fobjval - objval) > 0.000001 * (1.0 +
Math.Abs(objval)) Then
            End If

            Dim fvars() As GRBVar = fixedmodel.GetVars()
            Dim x() As Double = fixedmodel.Get(GRB.DoubleAttr.X, fvars)
            Dim vnames() As String =
fixedmodel.Get(GRB.StringAttr.VarName, fvars)

            For j As Integer = 0 To fvars.Length - 1
                If x(j) <> 0 Then
                    Console.WriteLine(vnames(j) & " " & x(j))
                End If
            Next

            ' Dispose of models and env
            fixedmodel.Dispose()
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：params_vb.vb

params_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Use parameters that are associated with a model.

' A MIP is solved for 5 seconds with different sets of parameters.
' The one with the smallest MIP gap is selected, and the optimization
' is resumed until the optimal solution is found.

Imports System
Imports Gurobi

Class params_vb
    ' Simple function to determine the MIP gap
    Private Shared Function Gap(ByVal model As GRBModel) As Double
        If (model.Get(GRB.IntAttr.SolCount) = 0) OrElse _
           (Math.Abs(model.Get(GRB.DoubleAttr.ObjVal)) < 0.000001) Then
            Return GRB.INFINITY
        End If
        Return Math.Abs(model.Get(GRB.DoubleAttr.ObjBound) -
model.Get(GRB.DoubleAttr.ObjVal)) / _
                   Math.Abs(model.Get(GRB.DoubleAttr.ObjVal))
    End Function

    Shared Sub Main(ByVal args As String())

        If args.Length < 1 Then
            Console.WriteLine("Usage: params_vb filename")
            Return
        End If

        Try
            ' Read model and verify that it is a MIP
            Dim env As New GRBEnv()
            Dim basemodel As New GRBModel(env, args(0))
            If basemodel.Get(GRB.IntAttr.IsMIP) = 0 Then
                Console.WriteLine("The model is not an integer program")
                Environment.Exit(1)
            End If

            ' Set a 5 second time limit
            basemodel.GetEnv().Set(GRB.DoubleParam.TimeLimit, 5)

            ' Now solve the model with different values of MIPFocus
            Dim bestGap As Double = GRB.INFINITY
            Dim bestModel As GRBModel = Nothing
            For i As Integer = 0 To 3
                Dim m As New GRBModel(basemodel)
                m.GetEnv().Set(GRB.IntParam.MIPFocus, i)
                m.Optimize()
                If bestModel Is Nothing OrElse bestGap > Gap(m) Then
                    bestModel = m
                    bestGap = Gap(bestModel)
                End If
            Next

            ' Finally, reset the time limit and continue to solve the
            ' best model to optimality
            bestModel.GetEnv().Set(GRB.DoubleParam.TimeLimit,
GRB.INFINITY)
            bestModel.Optimize()

            Console.WriteLine("Solved with MIPFocus: " & _

bestModel.GetEnv().Get(GRB.IntParam.MIPFocus))
        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：qp1_vb.vb

qp1_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.

' This example formulates and solves the following simple QP model:
'
'     minimize    x^2 + x*y + y^2 + y*z + z^2
'     subject to  x + 2 y + 3 z >= 4
'                 x +   y       >= 1
'
'   It solves it once as a continuous model, and once as an integer
model.
'

Imports Gurobi

Class qp1_vb
    Shared Sub Main()
        Try
            Dim env As New GRBEnv("qp1.log")
            Dim model As New GRBModel(env)

            ' Create variables

            Dim x As GRBVar = model.AddVar(0.0, 1.0, 0.0,
GRB.CONTINUOUS, "x")
            Dim y As GRBVar = model.AddVar(0.0, 1.0, 0.0,
GRB.CONTINUOUS, "y")
            Dim z As GRBVar = model.AddVar(0.0, 1.0, 0.0,
GRB.CONTINUOUS, "z")

            ' Integrate new variables

            model.Update()

            ' Set objective

            Dim obj As New GRBQuadExpr()
            obj = x*x + x*y + y*y + y*z + z*z
            model.SetObjective(obj)

            ' Add constraint: x + 2 y + 3 z >= 4

            model.AddConstr(x + 2 * y + 3 * z >= 4.0, "c0")

            ' Add constraint: x + y >= 1

            model.AddConstr(x + y >= 1.0, "c1")

            ' Optimize model

            model.Optimize()
            Console.WriteLine(x.Get(GRB.StringAttr.VarName) & " " &
x.Get(GRB.DoubleAttr.X))
            Console.WriteLine(y.Get(GRB.StringAttr.VarName) & " " &
y.Get(GRB.DoubleAttr.X))
            Console.WriteLine(z.Get(GRB.StringAttr.VarName) & " " &
z.Get(GRB.DoubleAttr.X))

            Console.WriteLine("Obj: " & model.Get(GRB.DoubleAttr.ObjVal)
& " " & obj.Value)


            ' Change variable types to integer

            x.Set(GRB.CharAttr.VType, GRB.INTEGER)
            y.Set(GRB.CharAttr.VType, GRB.INTEGER)
            z.Set(GRB.CharAttr.VType, GRB.INTEGER)

            ' Optimize model

            model.Optimize()

            Console.WriteLine(x.Get(GRB.StringAttr.VarName) & " " &
x.Get(GRB.DoubleAttr.X))
            Console.WriteLine(y.Get(GRB.StringAttr.VarName) & " " &
y.Get(GRB.DoubleAttr.X))
            Console.WriteLine(z.Get(GRB.StringAttr.VarName) & " " &
z.Get(GRB.DoubleAttr.X))

            Console.WriteLine("Obj: " & model.Get(GRB.DoubleAttr.ObjVal)
& " " & obj.Value)

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine(("Error code: " & e.ErrorCode & ". ") &
e.Message)
        End Try
    End Sub
End Class
```

页面：sensitivity_vb.vb

sensitivity_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Simple MIP sensitivity analysis example.
' For each integer variable, fix it to its lower and upper bound
' and check the impact on the objective.

Imports System
Imports Gurobi

Class sensitivity_vb
    Shared Sub Main(ByVal args As String())

        If args.Length < 1 Then
            Console.WriteLine("Usage: sensitivity_vb filename")
            Return
        End If

        Try
            ' Read model
            Dim env As New GRBEnv()
            Dim a As New GRBModel(env, args(0))
            a.Optimize()
            a.GetEnv().Set(GRB.IntParam.OutputFlag, 0)

            ' Extract variables from model
            Dim avars As GRBVar() = a.GetVars()

            For i As Integer = 0 To avars.Length - 1
                Dim v As GRBVar = avars(i)
                If v.Get(GRB.CharAttr.VType) = GRB.BINARY Then

                    ' Create clone and fix variable
                    Dim b As New GRBModel(a)
                    Dim bv As GRBVar = b.GetVars()(i)
                    If v.Get(GRB.DoubleAttr.X) -
v.Get(GRB.DoubleAttr.LB) < 0.5 Then
                        bv.Set(GRB.DoubleAttr.LB,
bv.Get(GRB.DoubleAttr.UB))
                    Else
                        bv.Set(GRB.DoubleAttr.UB,
bv.Get(GRB.DoubleAttr.LB))
                    End If

                    b.Optimize()

                    If b.Get(GRB.IntAttr.Status) = GRB.Status.OPTIMAL
Then
                        Dim objchg As Double =
b.Get(GRB.DoubleAttr.ObjVal) - _

a.Get(GRB.DoubleAttr.ObjVal)
                        If objchg < 0 Then
```

```
                            objchg = 0
                        End If
                        Console.WriteLine("Objective sensitivity for
variable " & _
                                          v.Get(GRB.StringAttr.VarName)
& " is " & objchg)
                    Else
                        Console.WriteLine("Objective sensitivity for
variable " & _
                                          v.Get(GRB.StringAttr.VarName)
& " is infinite")
                    End If

                    ' Dispose of model
                    b.Dispose()
                End If
            Next

            ' Dispose of model and env
            a.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " +
e.Message)
        End Try
    End Sub
End Class
```

页面：sos_vb.vb

sos_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' This example creates a very simple Special Ordered Set (SOS) model.
' The model consists of 3 continuous variables, no linear constraints,
' and a pair of SOS constraints of type 1.

Imports System
Imports Gurobi

Class sos_vb
    Shared Sub Main()
        Try
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env)

            ' Create variables

            Dim ub As Double() = {1, 1, 2}
            Dim obj As Double() = {-2, -1, -1}
            Dim names As String() = {"x0", "x1", "x2"}

            Dim x As GRBVar() = model.AddVars(Nothing, ub, obj, Nothing,
names)

            ' Integrate new variables

            model.Update()

            ' Add first SOS1: x0=0 or x1=0

            Dim sosv1 As GRBVar() = {x(0), x(1)}
            Dim soswt1 As Double() = {1, 2}

            model.AddSOS(sosv1, soswt1, GRB.SOS_TYPE1)

            ' Add second SOS1: x0=0 or x2=0

            Dim sosv2 As GRBVar() = {x(0), x(2)}
            Dim soswt2 As Double() = {1, 2}

            model.AddSOS(sosv2, soswt2, GRB.SOS_TYPE1)

            ' Optimize model

            model.Optimize()

            For i As Integer = 0 To 2
                Console.WriteLine(x(i).Get(GRB.StringAttr.VarName) & "
" & x(i).Get(GRB.DoubleAttr.X))
            Next

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：sudoku_vb.vb

sudoku_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Sudoku example.
'
' The Sudoku board is a 9x9 grid, which is further divided into a 3x3
grid
' of 3x3 grids.  Each cell in the grid must take a value from 0 to 9.
' No two grid cells in the same row, column, or 3x3 subgrid may take
the
' same value.

' In the MIP formulation, binary variables x(i,j,v) indicate whether
' cell <i,j> takes value 'v'.  The constraints are as follows:
'   1. Each cell must take exactly one value (sum_v x(i,j,v) = 1)
'   2. Each value is used exactly once per row (sum_i x(i,j,v) = 1)
'   3. Each value is used exactly once per column (sum_j x(i,j,v) = 1)
'   4. Each value is used exactly once per 3x3 subgrid (sum_grid
x(i,j,v) = 1)

Imports System
Imports System.IO
Imports Gurobi

Class sudoku_vb
    Shared Sub Main(ByVal args as String())
        Dim n As Integer = 9
        Dim s As Integer = 3

        If args.Length < 1 Then
            Console.WriteLine("Usage: sudoku_vb filename")
            Return
        End If

        Try
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env)

            ' Create 3-D array of model variables

            Dim vars As GRBVar(,,) = New GRBVar(n - 1, n - 1, n - 1) {}

            For i As Integer = 0 To n - 1
                For j As Integer = 0 To n - 1
                    For v As Integer = 0 To n - 1
                        Dim st As String = "G_" & i & "_" & j & "_" & v
                        vars(i, j, v) = model.AddVar(0.0, 1.0, 0.0,
GRB.BINARY, st)
                    Next
                Next
            Next

            ' Integrate variables into model
            model.Update()

            ' Add constraints

            Dim expr As GRBLinExpr

            ' Each cell must take one value

            For i As Integer = 0 To n - 1
                For j As Integer = 0 To n - 1
                    expr = 0
                    For v As Integer = 0 To n - 1
                        expr += vars(i, j, v)
                    Next
                    Dim st As String = "V_" & i & "_" & j
                    model.AddConstr(expr = 1, st)
                Next
            Next

            ' Each value appears once per row

            For i As Integer = 0 To n - 1
                For v As Integer = 0 To n - 1
                    expr = 0
                    For j As Integer = 0 To n - 1
                        expr += vars(i, j, v)
                    Next
                    Dim st As String = "R_" & i & "_" & v
                    model.AddConstr(expr = 1, st)
                Next
            Next

            ' Each value appears once per column

            For j As Integer = 0 To n - 1
                For v As Integer = 0 To n - 1
                    expr = 0
                    For i As Integer = 0 To n - 1
                        expr += vars(i, j, v)
                    Next
                    Dim st As String = "C_" & j & "_" & v
                    model.AddConstr(expr = 1, st)
                Next
            Next

            ' Each value appears once per sub-grid

            For v As Integer = 0 To n - 1
                For i0 As Integer = 0 To s - 1
                    For j0 As Integer = 0 To s - 1
                        expr = 0
                        For i1 As Integer = 0 To s - 1
                            For j1 As Integer = 0 To s - 1
                                expr += vars(i0 * s + i1, j0 * s + j1,
v)
                            Next
                        Next
                        Dim st As String = "Sub_" & v & "_" & i0 & "_"
& j0
                        model.AddConstr(expr = 1, st)
                    Next
                Next
            Next

            ' Update model

            model.Update()

            ' Fix variables associated with pre-specified cells

            Dim sr As StreamReader = File.OpenText(args(0))

            For i As Integer = 0 To n - 1
                Dim input As String = sr.ReadLine()
                For j As Integer = 0 To n - 1
                    Dim val As Integer =
Microsoft.VisualBasic.Asc(input(j)) - 48 - 1
                    ' 0-based
                    If val >= 0 Then
                        vars(i, j, val).Set(GRB.DoubleAttr.LB, 1.0)
                    End If
                Next
            Next

            ' Optimize model

            model.Optimize()

            ' Write model to file
            model.Write("sudoku.lp")

            Dim x As Double(,,) = model.Get(GRB.DoubleAttr.X, vars)

            Console.WriteLine()
            For i As Integer = 0 To n - 1
                For j As Integer = 0 To n - 1
                    For v As Integer = 0 To n - 1
                        If x(i, j, v) > 0.5 Then
                            Console.Write(v + 1)
                        End If
                    Next
                Next
                Console.WriteLine()
            Next

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
```

```
    End Sub
End Class
```

页面：workforce1_vb.vb

workforce1_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Assign workers to shifts; each worker may or may not be available on
a
' particular day. If the problem cannot be solved, use IIS to find a
set of
' conflicting constraints. Note that there may be additional conflicts
' besides what is reported via IIS.

Imports System
Imports Gurobi

Class workforce1_vb
    Shared Sub Main()
        Try

            ' Sample data
            ' Sets of days and workers
            Dim Shifts As String() = New String() {"Mon1", "Tue2",
"Wed3", "Thu4", "Fri5", "Sat6", _
                                                   "Sun7", "Mon8",
"Tue9", "Wed10", "Thu11", _
                                                   "Fri12", "Sat13",
"Sun14"}
            Dim Workers As String() = New String() {"Amy", "Bob",
"Cathy", "Dan", "Ed", "Fred", _
                                                    "Gu"}

            Dim nShifts As Integer = Shifts.Length
            Dim nWorkers As Integer = Workers.Length

            ' Number of workers required for each shift
            Dim shiftRequirements As Double() = New Double() {3, 2, 4,
4, 5, 6, _
                                                              5, 2, 2,
3, 4, 6, _
                                                              7, 5}

            ' Amount each worker is paid to work one shift
            Dim pay As Double() = New Double() {10, 12, 10, 8, 8, 9, 11}

            ' Worker availability: 0 if the worker is unavailable for a
shift
            Dim availability As Double(,) = New Double(,) { _
                       {0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1}, _
                       {1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0}, _
                       {0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1}, _
                       {0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1}, _
                       {1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1}, _
                       {1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1}, _
                       {1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}}
            ' Model
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env)
            model.Set(GRB.StringAttr.ModelName, "assignment")

            ' Assignment variables: x(w)(s) == 1 if worker w is
assigned
            ' to shift s. Since an assignment model always produces
integer
            ' solutions, we use continuous variables and solve as an LP.
            Dim x As GRBVar(,) = New GRBVar(nWorkers - 1, nShifts - 1)
{}
            For w As Integer = 0 To nWorkers - 1
                For s As Integer = 0 To nShifts - 1
                    x(w, s) = model.AddVar(0, availability(w, s),
pay(w), GRB.CONTINUOUS, _
                                           Workers(w) & "." & Shifts(s))
                Next
            Next

            ' The objective is to minimize the total pay costs
            model.Set(GRB.IntAttr.ModelSense, 1)

            ' Update model to integrate new variables
            model.Update()

            ' Constraint: assign exactly shiftRequirements(s) workers
            ' to each shift s
            For s As Integer = 0 To nShifts - 1
                Dim lhs As GRBLinExpr = 0
                For w As Integer = 0 To nWorkers - 1
                    lhs += x(w, s)
                Next
                model.AddConstr(lhs = shiftRequirements(s), Shifts(s))
            Next

            ' Optimize
            model.Optimize()
            Dim status As Integer = model.Get(GRB.IntAttr.Status)
            If status = GRB.Status.UNBOUNDED Then
                Console.WriteLine("The model cannot be solved " &
"because it is unbounded")
                Exit Sub
            End If
            If status = GRB.Status.OPTIMAL Then
                Console.WriteLine("The optimal objective is " &
model.Get(GRB.DoubleAttr.ObjVal))
                Exit Sub
            End If
            If (status <> GRB.Status.INF_OR_UNBD) AndAlso (status <>
GRB.Status.INFEASIBLE) Then
                Console.WriteLine("Optimization was stopped with status
" & status)
                Exit Sub
            End If

            ' Do IIS
            Console.WriteLine("The model is infeasible; computing IIS")
            model.ComputeIIS()
            Console.WriteLine(vbLf & "The following constraint(s) " &
"cannot be satisfied:")
            For Each c As GRBConstr In model.GetConstrs()
                If c.Get(GRB.IntAttr.IISConstr) = 1 Then
                    Console.WriteLine(c.Get(GRB.StringAttr.ConstrName))
                End If

            Next

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub
End Class
```

页面：workforce2_vb.vb

workforce2_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Assign workers to shifts; each worker may or may not be available on
a
' particular day. If the problem cannot be solved, use IIS iteratively
to
' find all conflicting constraints.

Imports System
Imports System.Collections.Generic
Imports Gurobi

Class workforce2_vb
    Shared Sub Main()
        Try

            ' Sample data
            ' Sets of days and workers
            Dim Shifts As String() = New String() {"Mon1", "Tue2",
"Wed3", "Thu4", "Fri5", "Sat6", _
                                                   "Sun7", "Mon8",
"Tue9", "Wed10", "Thu11", _
                                                   "Fri12", "Sat13",
"Sun14"}
            Dim Workers As String() = New String() {"Amy", "Bob",
"Cathy", "Dan", "Ed", "Fred", _
                                                    "Gu"}

            Dim nShifts As Integer = Shifts.Length
            Dim nWorkers As Integer = Workers.Length

            ' Number of workers required for each shift
            Dim shiftRequirements As Double() = New Double() {3, 2, 4,
4, 5, 6, _
                                                              5, 2, 2,
3, 4, 6, _
                                                              7, 5}

            ' Amount each worker is paid to work one shift
            Dim pay As Double() = New Double() {10, 12, 10, 8, 8, 9, 11}

            ' Worker availability: 0 if the worker is unavailable for a
shift
            Dim availability As Double(,) = New Double(,) { _
                       {0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1}, _
                       {1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0}, _
                       {0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1}, _
                       {0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1}, _
                       {1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1}, _
                       {1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1}, _
                       {1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}}
            ' Model
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env)
            model.Set(GRB.StringAttr.ModelName, "assignment")

            ' Assignment variables: x(w)(s) == 1 if worker w is
assigned
            ' to shift s. Since an assignment model always produces
integer
            ' solutions, we use continuous variables and solve as an LP.
            Dim x As GRBVar(,) = New GRBVar(nWorkers - 1, nShifts - 1)
{}
            For w As Integer = 0 To nWorkers - 1
                For s As Integer = 0 To nShifts - 1
                    x(w, s) = model.AddVar(0, availability(w, s),
pay(w), GRB.CONTINUOUS, _
                                           Workers(w) & "." & Shifts(s))
                Next
            Next

            ' The objective is to minimize the total pay costs
            model.Set(GRB.IntAttr.ModelSense, 1)

            ' Update model to integrate new variables
            model.Update()

            ' Constraint: assign exactly shiftRequirements(s) workers
            ' to each shift s
            For s As Integer = 0 To nShifts - 1
                Dim lhs As GRBLinExpr = 0
                For w As Integer = 0 To nWorkers - 1
                    lhs += x(w, s)
                Next
                model.AddConstr(lhs = shiftRequirements(s), Shifts(s))
            Next

            ' Optimize
            model.Optimize()
            Dim status As Integer = model.Get(GRB.IntAttr.Status)
            If status = GRB.Status.UNBOUNDED Then
                Console.WriteLine("The model cannot be solved " &
"because it is unbounded")
                Exit Sub
            End If
            If status = GRB.Status.OPTIMAL Then
                Console.WriteLine("The optimal objective is " &
model.Get(GRB.DoubleAttr.ObjVal))
                Exit Sub
            End If
            If (status <> GRB.Status.INF_OR_UNBD) AndAlso (status <>
GRB.Status.INFEASIBLE) Then
                Console.WriteLine("Optimization was stopped with status
" & status)
                Exit Sub
            End If

            ' Do IIS
            Console.WriteLine("The model is infeasible; computing IIS")
            Dim removed As LinkedList(Of String) = New LinkedList(Of
String)()

            ' Loop until we reduce to a model that can be solved
            While True
                model.ComputeIIS()
                Console.WriteLine(vbLf & "The following constraint
cannot be satisfied:")
                For Each c As GRBConstr In model.GetConstrs()
                    If c.Get(GRB.IntAttr.IISConstr) = 1 Then

Console.WriteLine(c.Get(GRB.StringAttr.ConstrName))
                        ' Remove a single constraint from the model

removed.AddFirst(c.Get(GRB.StringAttr.ConstrName))
                        model.Remove(c)
                        Exit For
                    End If
                Next

                Console.WriteLine()
                model.Optimize()
                status = model.Get(GRB.IntAttr.Status)

                If status = GRB.Status.UNBOUNDED Then
                    Console.WriteLine("The model cannot be solved " &
"because it is unbounded")
                    Exit Sub
                End If
                If status = GRB.Status.OPTIMAL Then
                    Exit While
                End If
                If (status <> GRB.Status.INF_OR_UNBD) AndAlso _
                   (status <> GRB.Status.INFEASIBLE) Then
                    Console.WriteLine("Optimization was stopped with
status " & status)
                    Exit Sub
                End If
            End While

            Console.WriteLine(vbLf & "The following constraints were
removed " & _
                              "to get a feasible LP:")
            For Each s As String In removed
                Console.Write(s & " ")
            Next

            Console.WriteLine()

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
```

```
        End Try
    End Sub
End Class
```

页面：workforce3_vb.vb

workforce3_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Assign workers to shifts; each worker may or may not be available on
a
' particular day. If the problem cannot be solved, add slack variables
' to determine which constraints cannot be satisfied, and how much
' they need to be relaxed.

Imports System
Imports System.Collections.Generic
Imports Gurobi

Class workforce3_vb
    Shared Sub Main()
        Try

            ' Sample data
            ' Sets of days and workers
            Dim Shifts As String() = New String() {"Mon1", "Tue2",
"Wed3", "Thu4", "Fri5", "Sat6", _
                                                   "Sun7", "Mon8",
"Tue9", "Wed10", "Thu11", _
                                                   "Fri12", "Sat13",
"Sun14"}
            Dim Workers As String() = New String() {"Amy", "Bob",
"Cathy", "Dan", "Ed", "Fred", _
                                                    "Gu"}

            Dim nShifts As Integer = Shifts.Length
            Dim nWorkers As Integer = Workers.Length

            ' Number of workers required for each shift
            Dim shiftRequirements As Double() = New Double() {3, 2, 4,
4, 5, 6, _
                                                              5, 2, 2,
3, 4, 6, _
                                                              7, 5}

            ' Amount each worker is paid to work one shift
            Dim pay As Double() = New Double() {10, 12, 10, 8, 8, 9, 11}

            ' Worker availability: 0 if the worker is unavailable for a
shift
            Dim availability As Double(,) = New Double(,) { _
                       {0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1}, _
                       {1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0}, _
                       {0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1}, _
                       {0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1}, _
                       {1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1}, _
                       {1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1}, _
                       {1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}}
            ' Model
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env)
            model.Set(GRB.StringAttr.ModelName, "assignment")

            ' Assignment variables: x(w)(s) == 1 if worker w is
assigned
            ' to shift s. Since an assignment model always produces
integer
            ' solutions, we use continuous variables and solve as an LP.
            Dim x As GRBVar(,) = New GRBVar(nWorkers - 1, nShifts - 1)
{}
            For w As Integer = 0 To nWorkers - 1
                For s As Integer = 0 To nShifts - 1
                    x(w, s) = model.AddVar(0, availability(w, s),
pay(w), GRB.CONTINUOUS, _
                                           Workers(w) & "." & Shifts(s))
                Next
            Next

            ' The objective is to minimize the total pay costs
            model.Set(GRB.IntAttr.ModelSense, 1)

            ' Update model to integrate new variables
            model.Update()

            ' Constraint: assign exactly shiftRequirements(s) workers
            ' to each shift s
            Dim reqCts As New LinkedList(Of GRBConstr)()
            For s As Integer = 0 To nShifts - 1
                Dim lhs As GRBLinExpr = 0
                For w As Integer = 0 To nWorkers - 1
                    lhs += x(w, s)
                Next
                Dim newCt As GRBConstr = model.AddConstr(lhs =
shiftRequirements(s), Shifts(s))
                reqCts.AddFirst(newCt)
            Next

            ' Optimize
            model.Optimize()
            Dim status As Integer = model.Get(GRB.IntAttr.Status)
            If status = GRB.Status.UNBOUNDED Then
                Console.WriteLine("The model cannot be solved " &
"because it is unbounded")
                Exit Sub
            End If
            If status = GRB.Status.OPTIMAL Then
                Console.WriteLine("The optimal objective is " &
model.Get(GRB.DoubleAttr.ObjVal))
                Exit Sub
            End If
            If (status <> GRB.Status.INF_OR_UNBD) AndAlso (status <>
GRB.Status.INFEASIBLE) Then
                Console.WriteLine("Optimization was stopped with status
" & status)
                Exit Sub
            End If

            ' Add slack variables to make the model feasible
            Console.WriteLine("The model is infeasible; adding slack
variables")

            ' Set original objective coefficients to zero
            model.SetObjective(New GRBLinExpr())

            ' Add a new slack variable to each shift constraint so that
the shifts
            ' can be satisfied
            Dim slacks As New LinkedList(Of GRBVar)()
            For Each c As GRBConstr In reqCts
                Dim col As New GRBColumn()
                col.AddTerm(1.0, c)
                Dim newvar As GRBVar = model.AddVar(0, GRB.INFINITY,
1.0, GRB.CONTINUOUS, col, _

c.Get(GRB.StringAttr.ConstrName) & "Slack")
                slacks.AddFirst(newvar)
            Next

            ' Solve the model with slacks
            model.Optimize()
            status = model.Get(GRB.IntAttr.Status)
            If (status = GRB.Status.INF_OR_UNBD) OrElse _
               (status = GRB.Status.INFEASIBLE) OrElse _
               (status = GRB.Status.UNBOUNDED) Then
                Console.WriteLine("The model with slacks cannot be
solved " & _
                                  "because it is infeasible or
unbounded")
                Exit Sub
            End If
            If status <> GRB.Status.OPTIMAL Then
                Console.WriteLine("Optimization was stopped with status
" & status)
                Exit Sub
            End If

            Console.WriteLine(vbLf & "Slack values:")
            For Each sv As GRBVar In slacks
                If sv.Get(GRB.DoubleAttr.X) > 0.000001 Then
                    Console.WriteLine(sv.Get(GRB.StringAttr.VarName) &
" = " & _
                                      sv.Get(GRB.DoubleAttr.X))
                End If
            Next

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
```

```
        End Try
    End Sub
End Class
```

页面：workforce4_vb.vb

workforce4_vb.vb

```
' Copyright 2011, Gurobi Optimization, Inc.
'
' Assign workers to shifts; each worker may or may not be available on
a
' particular day. We use Pareto optimization to solve the model:
' first, we minimize the linear sum of the slacks. Then, we constrain
' the sum of the slacks, and we minimize a quadratic objective that
' tries to balance the workload among the workers.

Imports System
Imports Gurobi

Class workforce4_vb
    Shared Sub Main()
        Try

            ' Sample data
            ' Sets of days and workers
            Dim Shifts As String() = New String() {"Mon1", "Tue2",
"Wed3", "Thu4", "Fri5", "Sat6", _
                                                   "Sun7", "Mon8",
"Tue9", "Wed10", "Thu11", _
                                                   "Fri12", "Sat13",
"Sun14"}
            Dim Workers As String() = New String() {"Amy", "Bob",
"Cathy", "Dan", "Ed", "Fred", _
                                                    "Gu"}

            Dim nShifts As Integer = Shifts.Length
            Dim nWorkers As Integer = Workers.Length

            ' Number of workers required for each shift
            Dim shiftRequirements As Double() = New Double() {3, 2, 4,
4, 5, 6, _
                                                              5, 2, 2,
3, 4, 6, _
                                                              7, 5}

            ' Worker availability: 0 if the worker is unavailable for a
shift
            Dim availability As Double(,) = New Double(,) { _
                       {0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1}, _
                       {1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0}, _
                       {0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1}, _
                       {0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1}, _
                       {1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1}, _
                       {1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1}, _
                       {1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}}

            ' Model
            Dim env As New GRBEnv()
            Dim model As New GRBModel(env)
            model.Set(GRB.StringAttr.ModelName, "assignment")

            ' Assignment variables: x(w)(s) == 1 if worker w is
assigned
            ' to shift s. This is no longer a pure assignment model, so
we
            ' must use binary variables.
            Dim x As GRBVar(,) = New GRBVar(nWorkers - 1, nShifts - 1)
{}
            For w As Integer = 0 To nWorkers - 1
                For s As Integer = 0 To nShifts - 1
                    x(w, s) = model.AddVar(0, availability(w, s), 0, _
                                           GRB.BINARY, _
                                           Workers(w) & "." & Shifts(s))
                Next
            Next

            ' Add a new slack variable to each shift constraint so that
the
            ' shifts can be satisfied
            Dim slacks As GRBVar() = New GRBVar(nShifts - 1) {}
            For s As Integer = 0 To nShifts - 1
                slacks(s) = _
                    model.AddVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS, _
                                 Shifts(s) & "Slack")
            Next

            ' Variable to represent the total slack
            Dim totSlack As GRBVar = model.AddVar(0, GRB.INFINITY, 0, _
                                                  GRB.CONTINUOUS,
"totSlack")

            ' Variables to count the total shifts worked by each worker
            Dim totShifts As GRBVar() = New GRBVar(nWorkers - 1) {}
            For w As Integer = 0 To nWorkers - 1
                totShifts(w) = _
                    model.AddVar(0, GRB.INFINITY, 0, GRB.CONTINUOUS, _
                                 Workers(w) & "TotShifts")
            Next

            ' Update model to integrate new variables
            model.Update()

            Dim lhs As GRBLinExpr

            ' Constraint: assign exactly shiftRequirements(s) workers
            ' to each shift s, plus the slack
            For s As Integer = 0 To nShifts - 1
                lhs = 0
                lhs += slacks(s)
                For w As Integer = 0 To nWorkers - 1
                    lhs += x(w, s)
                Next
                model.AddConstr(lhs = shiftRequirements(s), Shifts(s))
            Next

            ' Constraint: set totSlack equal to the total slack
            lhs = 0
            For s As Integer = 0 To nShifts - 1
                lhs += slacks(s)
            Next
            model.AddConstr(lhs = totSlack, "totSlack")

            ' Constraint: compute the total number of shifts for each
worker
            For w As Integer = 0 To nWorkers - 1
                lhs = 0
                For s As Integer = 0 To nShifts - 1
                    lhs += x(w, s)
                Next
                model.AddConstr(lhs = totShifts(w), "totShifts" &
Workers(w))
            Next

            ' Objective: minimize the total slack
            Dim obj As GRBLinExpr = New GRBLinExpr
            obj += totSlack
            model.SetObjective(obj)

            ' Optimize
            Dim status As Integer = _
                solveAndPrint(model, totSlack, nWorkers, Workers,
totShifts)
            If status <> GRB.Status.OPTIMAL Then
                Exit Sub
            End If

            ' Constrain the slack by setting its upper and lower bounds
            totSlack.Set(GRB.DoubleAttr.UB,
totSlack.Get(GRB.DoubleAttr.X))
            totSlack.Set(GRB.DoubleAttr.LB,
totSlack.Get(GRB.DoubleAttr.X))

            ' Variable to count the average number of shifts worked
            Dim avgShifts As GRBVar = model.AddVar(0, GRB.INFINITY, 0,
_
                                                  GRB.CONTINUOUS,
"avgShifts")

            ' Variables to count the difference from average for each
worker;
            ' note that these variables can take negative values.
            Dim diffShifts As GRBVar() = New GRBVar(nWorkers - 1) {}
            For w As Integer = 0 To nWorkers - 1
                diffShifts(w) = _
                    model.AddVar(-GRB.INFINITY, GRB.INFINITY, 0, _
                                 GRB.CONTINUOUS, Workers(w) & "Diff")
            Next

            ' Update model to integrate new variables
            model.Update()

            ' Constraint: compute the average number of shifts worked
            lhs = 0
            For w As Integer = 0 To nWorkers - 1
                lhs += totShifts(w)
            Next
            model.AddConstr(lhs = nWorkers * avgShifts, "avgShifts")

            ' Constraint: compute the difference from the average
number of shifts
            For w As Integer = 0 To nWorkers - 1
                lhs = 0
                lhs += totShifts(w)
                lhs -= avgShifts
                model.AddConstr(lhs = diffShifts(w), Workers(w) &
"Diff")
            Next

            ' Objective: minimize the sum of the square of the
difference
            ' from the average number of shifts worked
            Dim qobj As GRBQuadExpr = New GRBQuadExpr
            For w As Integer = 0 To nWorkers - 1
                qobj += diffShifts(w) * diffShifts(w)
            Next
            model.SetObjective(qobj)

            ' Optimize
            status = solveAndPrint(model, totSlack, nWorkers, Workers,
totShifts)
            If status <> GRB.Status.OPTIMAL Then
                Exit Sub
            End If

            ' Dispose of model and env
            model.Dispose()
            env.Dispose()

        Catch e As GRBException
            Console.WriteLine("Error code: " & e.ErrorCode & ". " &
e.Message)
        End Try
    End Sub

    Private Shared Function solveAndPrint(ByVal model As GRBModel,
ByVal totSlack As GRBVar, _
                                          ByVal nWorkers As Integer,
ByVal Workers As String(), _
                                          ByVal totShifts As GRBVar())
As Integer
        model.Optimize()
        Dim status As Integer = model.Get(GRB.IntAttr.Status)
        solveAndPrint = status
        If (status = GRB.Status.INF_OR_UNBD) OrElse _
           (status = GRB.Status.INFEASIBLE) OrElse _
           (status = GRB.Status.UNBOUNDED) Then
            Console.WriteLine("The model cannot be solved because " & _
                              "it is infeasible or unbounded")
            Exit Function
        End If
        If status <> GRB.Status.OPTIMAL Then
            Console.WriteLine("Optimization was stopped with status " _
                              & status)
            Exit Function
        End If

        ' Print total slack and the number of shifts worked for each
worker
        Console.WriteLine(vbLf & "Total slack required: " & _
                          totSlack.Get(GRB.DoubleAttr.X))
        For w As Integer = 0 To nWorkers - 1
            Console.WriteLine(Workers(w) & " worked " & _
                              totShifts(w).Get(GRB.DoubleAttr.X) & _
                              " shifts")
        Next

        Console.WriteLine(vbLf)
    End Function
End Class
```

页面：Python Examples

Python 示例

本节包含了所有 Gurobi Python 的示例源代码。相同的源代码也可以在 Gurobi 发布包的 *examples/python* 目录中找到。

小节

- sudoku.py
- workforce1.py
- workforce2.py
- workforce3.py
- workforce4.py

页面：callback.py

callback.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# This example reads an LP or a MIP from a file, sets a callback
# to monitor the optimization progress, and outputs progress
# information to the screen and to the log file. If the input model
# is a MIP, the callback aborts the optimization after 10000 nodes have
# been explored.


import sys
from gurobipy import *


# Define callback function

def mycallback(model, where):
    if where == GRB.callback.PRESOLVE:
        print 'Removed', model.cbGet(GRB.callback.PRE_COLDEL), 'columns
and', \
              model.cbGet(GRB.callback.PRE_ROWDEL), 'rows'
    elif where == GRB.callback.SIMPLEX:
        itcnt = model.cbGet(GRB.callback.SPX_ITRCNT)
        if itcnt % 100 == 0:
            obj  = model.cbGet(GRB.callback.SPX_OBJVAL)
            pinf = model.cbGet(GRB.callback.SPX_PRIMINF)
            dinf = model.cbGet(GRB.callback.SPX_DUALINF)
            pert = model.cbGet(GRB.callback.SPX_ISPERT)
            if pert == 0:
                ch = ' '
            elif pert == 1:
                ch = 'S'
            else:
                ch = 'P'
            print int(itcnt), obj, ch, pinf, dinf
    elif where == GRB.callback.MIP:
        nodecnt = model.cbGet(GRB.callback.MIP_NODCNT)
        if nodecnt % 100 == 0:
            objbst = model.cbGet(GRB.callback.MIP_OBJBST)
            objbnd = model.cbGet(GRB.callback.MIP_OBJBND)
            print int(nodecnt), objbst, objbnd
        if nodecnt > model._mynodelimit:
            model.terminate()
    elif where == GRB.callback.MIPSOL:
        obj     = model.cbGet(GRB.callback.MIPSOL_OBJ)
        nodecnt = int(model.cbGet(GRB.callback.MIPSOL_NODCNT))
        print '*** New solution at node', nodecnt, 'objective', obj
        print model.cbGetSolution(model.getVars())
    elif where == GRB.callback.MIPNODE:
        print '*** New node'
        if model.cbGet(GRB.callback.MIPNODE_STATUS) ==
GRB.status.OPTIMAL:
            x = model.cbGetNodeRel(model.getVars())
            model.cbSetSolution(model.getVars(), x)

if len(sys.argv) < 2:
    print 'Usage: callback.py filename'
    quit()

# Read and solve model

model = read(sys.argv[1])

# Pass data into my callback function

model._mynodelimit = 10000

model.params.heuristics = 0

model.optimize(mycallback)
```

页面：custom.py

custom.py

```
#
# Copyright 2011, Gurobi Optimization, Inc.
#
# Interactive shell customization example
#
# Define a set of customizations for the Gurobi shell.
# Type 'from custom import *' to import them into your shell.
#

from gurobipy import *


# custom read command --- change directory as appropriate

def myread(name):
    return read('/home/jones/models/' + name)


# simple termination callback

def mycallback(model, where):
    if where == GRB.callback.MIP:
        time = model.cbGet(GRB.callback.RUNTIME)
        best = model.cbGet(GRB.callback.MIP_OBJBST)
        if time > 10 and best < GRB.INFINITY:
            model.terminate()


# custom optimize() function that uses callback

def myopt(model):
    model.optimize(mycallback)
```

页面：diet.py

diet.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Solve the classic diet model, showing how to add constraints
# to an existing model.

from gurobipy import *

# Nutrition guidelines, based on
# USDA Dietary Guidelines for Americans, 2005
# http://www.health.gov/DietaryGuidelines/dga2005/

categories, minNutrition, maxNutrition = multidict({
  'calories': [1800, 2200],
  'protein':  [91, GRB.INFINITY],
  'fat':      [0, 65],
  'sodium':   [0, 1779] })

foods, cost = multidict({
  'hamburger': 2.49,
  'chicken':   2.89,
  'hot dog':   1.50,
  'fries':     1.89,
  'macaroni':  2.09,
  'pizza':     1.99,
  'salad':     2.49,
  'milk':      0.89,
  'ice cream': 1.59 })

# Nutrition values for the foods
nutritionValues = {
  ('hamburger', 'calories'): 410,
  ('hamburger', 'protein'):  24,
  ('hamburger', 'fat'):      26,
  ('hamburger', 'sodium'):   730,
  ('chicken',   'calories'): 420,
  ('chicken',   'protein'):  32,
  ('chicken',   'fat'):      10,
  ('chicken',   'sodium'):   1190,
  ('hot dog',   'calories'): 560,
  ('hot dog',   'protein'):  20,
  ('hot dog',   'fat'):      32,
  ('hot dog',   'sodium'):   1800,
  ('fries',     'calories'): 380,
  ('fries',     'protein'):  4,
  ('fries',     'fat'):      19,
  ('fries',     'sodium'):   270,
  ('macaroni',  'calories'): 320,
  ('macaroni',  'protein'):  12,
  ('macaroni',  'fat'):      10,
  ('macaroni',  'sodium'):   930,
  ('pizza',     'calories'): 320,
  ('pizza',     'protein'):  15,
  ('pizza',     'fat'):      12,
  ('pizza',     'sodium'):   820,
  ('salad',     'calories'): 320,
  ('salad',     'protein'):  31,
  ('salad',     'fat'):      12,
  ('salad',     'sodium'):   1230,
  ('milk',      'calories'): 100,
  ('milk',      'protein'):  8,
  ('milk',      'fat'):      2.5,
  ('milk',      'sodium'):   125,
  ('ice cream', 'calories'): 330,
  ('ice cream', 'protein'):  8,
  ('ice cream', 'fat'):      10,
  ('ice cream', 'sodium'):   180 }

# Model
m = Model("diet")

# Create decision variables for the nutrition information,
# which we limit via bounds
nutrition = {}
for c in categories:
    nutrition[c] = m.addVar(lb=minNutrition[c], ub=maxNutrition[c],
name=c)

# Create decision variables for the foods to buy
buy = {}
for f in foods:
    buy[f] = m.addVar(obj=cost[f], name=f)

# The objective is to minimize the costs
m.modelSense = GRB.MINIMIZE

# Update model to integrate new variables
m.update()

# Nutrition constraints
for c in categories:
    m.addConstr(
      quicksum(nutritionValues[f,c] * buy[f] for f in foods) ==
nutrition[c],
               c)


def printSolution():
    if m.status == GRB.status.OPTIMAL:
        print '\nCost:', m.objVal
        print '\nBuy:'
        for f in foods:
            if buy[f].x > 0.0001:
                print f, buy[f].x
        print '\nNutrition:'
        for c in categories:
            print c, nutrition[c].x
    else:
        print 'No solution'

# Solve
m.optimize()
printSolution()

print '\nAdding constraint: at most 6 servings of dairy'
m.addConstr(buy['milk'] + buy['ice cream'] <= 6, "limit_dairy")

# Solve
m.optimize()
printSolution()
```

页面：diet2.py

diet2.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Separate the model (dietmodel.py) from the data file (diet2.py), so
# that the model can be solved with different data files.
#
# Nutrition guidelines, based on
# USDA Dietary Guidelines for Americans, 2005
# http://www.health.gov/DietaryGuidelines/dga2005/

from gurobipy import *

categories, minNutrition, maxNutrition = multidict({
  'calories': [1800, 2200],
  'protein':  [91, GRB.INFINITY],
  'fat':      [0, 65],
  'sodium':   [0, 1779] })

foods, cost = multidict({
  'hamburger': 2.49,
  'chicken':   2.89,
  'hot dog':   1.50,
  'fries':     1.89,
  'macaroni':  2.09,
  'pizza':     1.99,
  'salad':     2.49,
  'milk':      0.89,
  'ice cream': 1.59 })

# Nutrition values for the foods
nutritionValues = {
  ('hamburger', 'calories'): 410,
  ('hamburger', 'protein'):  24,
  ('hamburger', 'fat'):      26,
  ('hamburger', 'sodium'):   730,
  ('chicken',   'calories'): 420,
  ('chicken',   'protein'):  32,
  ('chicken',   'fat'):      10,
  ('chicken',   'sodium'):   1190,
  ('hot dog',   'calories'): 560,
  ('hot dog',   'protein'):  20,
  ('hot dog',   'fat'):      32,
  ('hot dog',   'sodium'):   1800,
  ('fries',     'calories'): 380,
  ('fries',     'protein'):  4,
  ('fries',     'fat'):      19,
  ('fries',     'sodium'):   270,
  ('macaroni',  'calories'): 320,
  ('macaroni',  'protein'):  12,
  ('macaroni',  'fat'):      10,
  ('macaroni',  'sodium'):   930,
  ('pizza',     'calories'): 320,
  ('pizza',     'protein'):  15,
  ('pizza',     'fat'):      12,
  ('pizza',     'sodium'):   820,
  ('salad',     'calories'): 320,
  ('salad',     'protein'):  31,
  ('salad',     'fat'):      12,
  ('salad',     'sodium'):   1230,
  ('milk',      'calories'): 100,
  ('milk',      'protein'):  8,
  ('milk',      'fat'):      2.5,
  ('milk',      'sodium'):   125,
  ('ice cream', 'calories'): 330,
  ('ice cream', 'protein'):  8,
  ('ice cream', 'fat'):      10,
  ('ice cream', 'sodium'):   180 }

import dietmodel
dietmodel.solve(categories, minNutrition, maxNutrition,
                foods, cost, nutritionValues)
```

页面：diet3.py

diet3.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Use a SQLite database with the diet model (dietmodel.py).  The
database
# (diet.db) can be recreated using the included SQL script (diet.sql).
#
# Note that this example reads an external data file (..\data\diet.db).
# As a result, it must be run from the distribution directory.

import os
import sqlite3
from gurobipy import *

con = sqlite3.connect(os.path.join('..', 'data', 'diet.db'))
cur = con.cursor()

cur.execute('select category,minnutrition,maxnutrition from categories')
result = cur.fetchall()
categories, minNutrition, maxNutrition = multidict(
    (cat,[minv,maxv]) for cat,minv,maxv in result)

cur.execute('select food,cost from foods')
result = cur.fetchall()
foods, cost = multidict(result)

cur.execute('select food,category,value from nutrition')
result = cur.fetchall()
nutritionValues = dict(((f,c),v) for f,c,v in result)

import dietmodel
dietmodel.solve(categories, minNutrition, maxNutrition,
                foods, cost, nutritionValues)
```

页面：diet4.py

diet4.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Read diet model data from an Excel spreadsheet (diet.xls).
# Pass the imported data into the diet model (dietmodel.py).
#
# Note that this example reads an external data file (..\data\diet.xls).
# As a result, it must be run from the distribution directory.

import os
import xlrd

book = xlrd.open_workbook(os.path.join("..", "data", "diet.xls"))

sh = book.sheet_by_name("Categories")
categories = []
minNutrition = {}
maxNutrition = {}
i = 1
while True:
    try:
        c = sh.cell_value(i, 0)
        categories.append(c)
        minNutrition[c] = sh.cell_value(i,1)
        maxNutrition[c] = sh.cell_value(i,2)
        i = i + 1
    except IndexError:
        break

sh = book.sheet_by_name("Foods")
foods = []
cost = {}
i = 1
while True:
    try:
        f = sh.cell_value(i, 0)
        foods.append(f)
        cost[f] = sh.cell_value(i,1)
        i = i + 1
    except IndexError:
        break

sh = book.sheet_by_name("Nutrition")
nutritionValues = {}
for i in range(len(foods)):
    for j in range(len(categories)):
        nutritionValues[foods[i],categories[j]]  =  sh.cell_value(i+1,j+1)

import dietmodel
dietmodel.solve(categories, minNutrition, maxNutrition,
                foods, cost, nutritionValues)
```

页面：dietmodel.py

dietmodel.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Solve the classic diet model.  This file implements
# a function that formulates and solves the model,
# but it contains no model data.  The data is
# passed in by the calling program.  Run example 'diet2.py',
# 'diet3.py', or 'diet4.py' to invoke this function.

from gurobipy import *


def solve(categories, minNutrition, maxNutrition, foods, cost,
          nutritionValues):
    # Model
    m = Model("diet")

    # Create decision variables for the nutrition information,
    # which we limit via bounds
    nutrition = {}
    for c in categories:
        nutrition[c] = m.addVar(lb=minNutrition[c], ub=maxNutrition[c],
name=c)

    # Create decision variables for the foods to buy
    buy = {}
    for f in foods:
        buy[f] = m.addVar(obj=cost[f], name=f)

    # The objective is to minimize the costs
    m.modelSense = GRB.MINIMIZE

    # Update model to integrate new variables
    m.update()

    # Nutrition constraints
    for c in categories:
        m.addConstr(
          quicksum(nutritionValues[f,c] * buy[f] for f in foods) ==
                    nutrition[c], c)

    def printSolution():
        if m.status == GRB.status.OPTIMAL:
            print '\nCost:', m.objVal
            print '\nBuy:'
            for f in foods:
                if buy[f].x > 0.0001:
                    print f, buy[f].x
            print '\nNutrition:'
            for c in categories:
                print c, nutrition[c].x
        else:
            print 'No solution'

    # Solve
    m.optimize()
    printSolution()

    print '\nAdding constraint: at most 6 servings of dairy'
    m.addConstr(buy['milk'] + buy['ice cream'] <= 6, "limit_dairy")

    # Solve
    m.optimize()
    printSolution()
```

页面：facility.py

facility.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Facility location: a company currently ships its product from 5
plants
# to 4 warehouses. It is considering closing some plants to reduce
# costs. What plant(s) should the company close, in order to minimize
# transportation and fixed costs?
#
# Note that this example uses lists instead of dictionaries.  Since
# it does not work with sparse data, lists are a reasonable option.
#
# Based on an example from Frontline Systems:
#   http://www.solver.com/disfacility.htm
# Used with permission.

from gurobipy import *

# Warehouse demand in thousands of units
demand = [15, 18, 14, 20]

# Plant capacity in thousands of units
capacity = [20, 22, 17, 19, 18]

# Fixed costs for each plant
fixedCosts = [12000, 15000, 17000, 13000, 16000]

# Transportation costs per thousand units
transCosts = [[4000, 2000, 3000, 2500, 4500],
              [2500, 2600, 3400, 3000, 4000],
              [1200, 1800, 2600, 4100, 3000],
              [2200, 2600, 3100, 3700, 3200]]

# Range of plants and warehouses
plants = range(len(capacity))
warehouses = range(len(demand))

# Model
m = Model("facility")

# Plant open decision variables: open[p] == 1 if plant p is open.
open = []
for p in plants:
    open.append(m.addVar(vtype=GRB.BINARY, name="Open%d" % p))

# Set optimization objective - minimize sum of fixed costs
m.setObjective(quicksum([fixedCosts[p]*open[p] for p in plants]))

# Transportation decision variables: how much to transport from
# a plant p to a warehouse w
transport = []
```

```
for w in warehouses:
    transport.append([])
    for p in plants:
        transport[w].append(m.addVar(obj=transCosts[w][p],
                                     name="Trans%d.%d" % (p, w)))

# The objective is to minimize the total fixed and variable costs
m.modelSense = GRB.MINIMIZE

# Update model to integrate new variables
m.update()

# Production constraints
# Note that the right-hand limit sets the production to zero if the
plant
# is closed
for p in plants:
    m.addConstr(
        quicksum(transport[w][p] for w in warehouses) <= capacity[p] *
open[p],
        "Capacity%d" % p)

# Demand constraints
for w in warehouses:
    m.addConstr(quicksum(transport[w][p] for p in plants) == demand[w],
                "Demand%d" % w)

# Guess at the starting point: close the plant with the highest fixed
costs;
# open all others

# First, open all plants
for p in plants:
    open[p].start = 1.0

# Now close the plant with the highest fixed cost
print 'Initial guess:'
maxFixed = max(fixedCosts)
for p in plants:
    if fixedCosts[p] == maxFixed:
        open[p].start = 0.0
        print 'Closing plant', p
        break
print

# Use barrier to solve root relaxation
m.params.method = 2

# Solve
m.optimize()

# Print solution
print '\nTOTAL COSTS:', m.objVal
print 'SOLUTION:'
for p in plants:
    if open[p].x == 1.0:
        print 'Plant', p, 'open:'
        for w in warehouses:
            if transport[w][p].x > 0:
                print '  Transport', transport[w][p].x, 'units to
warehouse', w
    else:
        print 'Plant', p, 'closed!'
```

页面：feasopt.py

feasopt.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# This example reads a MIP model from a file, adds artificial
# variables to each constraint, and then minimizes the sum of the
# artificial variables.  A solution with objective zero corresponds
# to a feasible solution to the input model.

import sys
from gurobipy import *

if len(sys.argv) < 2:
    print 'Usage: feasopt.py filename'
    quit()

feasmodel = gurobi.read(sys.argv[1])

# clear objective

feasmodel.setObjective(0.0)

# add slack variables

for c in feasmodel.getConstrs():
    sense = c.sense
    if sense != '>':
        feasmodel.addVar(obj=1.0, name="ArtN_" + c.constrName,
                         column=Column([-1], [c]))
    if sense != '<':
        feasmodel.addVar(obj=1.0, name="ArtP_" + c.constrName,
                         column=Column([1], [c]))

feasmodel.update()

# optimize modified model

feasmodel.write('feasopt.lp')

feasmodel.optimize()
```

页面：fixanddive.py

fixanddive.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Implement a simple MIP heuristic.  Relax the model,
# sort variables based on fractionality, and fix the 25% of
# the fractional variables that are closest to integer variables.
# Repeat until either the relaxation is integer feasible or
# linearly infeasible.

import sys
from gurobipy import *


# Comparison function used to sort variable list based on relaxation
# fractionality

def compare(v1, v2):
    sol1 = v1.x
    sol2 = v2.x
    frac1 = abs(sol1-int(sol1+0.5))
    frac2 = abs(sol2-int(sol2+0.5))
    if frac1 <= frac2:
        return -1
    else:
        return +1


if len(sys.argv) < 2:
    print 'Usage: fixanddive.py filename'
    quit()

# Read model

model = gurobi.read(sys.argv[1])

# Collect integer variables and relax them
intvars = []
for v in model.getVars():
    if v.vType != GRB.CONTINUOUS:
        intvars += [v]
        v.vType = GRB.CONTINUOUS

model.params.outputFlag = 0

model.optimize()


# Perform multiple iterations.  In each iteration, identify the first
# quartile of integer variables that are closest to an integer value in
the
# relaxation, fix them to the nearest integer, and repeat.
```

```
for iter in range(1000):

# create a list of fractional variables, sorted in order of increasing
# distance from the relaxation solution to the nearest integer value

    fractional = []
    for v in intvars:
        sol = v.x
        if abs(sol - int(sol+0.5)) > 1e-5:
            fractional += [v]

    fractional.sort(compare)

    print 'Iteration', iter, ', obj ', model.objVal, ', fractional',
    print len(fractional)

    if len(fractional) == 0:
        print 'Found feasible solution - objective', model.objVal
        break


# Fix the first quartile to the nearest integer value
    nfix = max(len(fractional)/4, 1)
    for i in range(nfix):
        v = fractional[i]
        fixval = int(v.x+0.5)
        v.lb = fixval
        v.ub = fixval
        print '  Fix', v.varName, 'to', fixval, '( rel', v.x, ')'

    model.optimize()

# Check optimization result

    if model.status != GRB.status.OPTIMAL:
        print 'Relaxation is infeasible'
        break
```

页面：lp.py

lp.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# This example reads an LP model from a file and solves it.
# If the model is infeasible or unbounded, the example turns off
# presolve and solves the model again. If the model is infeasible,
# the example computes an Irreducible Infeasible Subsystem (IIS),
# and writes it to a file

import sys
from gurobipy import *

if len(sys.argv) < 2:
    print 'Usage: lp.py filename'
    quit()

# Read and solve model

model = read(sys.argv[1])
model.optimize()

if model.status == GRB.status.INF_OR_UNBD:
    # Turn presolve off to determine whether model is infeasible
    # or unbounded
    model.setParam(GRB.param.presolve, 0)
    model.optimize()

if model.status == GRB.status.OPTIMAL:
    print 'Optimal objective:', model.objVal
    model.write('model.sol')
    exit(0)
elif model.status != GRB.status.INFEASIBLE:
    print 'Optimization was stopped with status', model.status
    exit(0)


# Model is infeasible - compute an Irreducible Infeasible Subsystem
(IIS)

print
print "Model is infeasible"
model.computeIIS()
model.write("model.ilp")
print "IIS written to file 'model.ilp'"
```

页面：lpmethod.py

lpmethod.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Solve a model with different values of the Method parameter;
# show which value gives the shortest solve time.

import sys
from gurobipy import *

if len(sys.argv) < 2:
    print 'Usage: lpmethod.py filename'
    quit()

# Read model
m = read(sys.argv[1])

# Solve the model with different values of Method
bestTime = m.params.timeLimit
bestMethod = -1
for i in range(3):
    m.reset()
    m.params.method = i
    m.optimize()
    if m.status == GRB.status.OPTIMAL:
        bestTime = m.Runtime
        bestMethod = i
        # Reduce the TimeLimit parameter to save time with other
methods
        m.params.timeLimit = bestTime

# Report which method was fastest
if bestMethod == -1:
    print 'Unable to solve this model'
else:
    print 'Solved in', bestTime, 'seconds with Method:', bestMethod
```

页面：lpmod.py

lpmod.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# This example reads an LP model from a file and solves it.
# If the model can be solved, then it finds the smallest positive
variable,
# sets its upper bound to zero, and resolves the model two ways:
# first with an advanced start, then without an advanced start
# (i.e. 'from scratch').

import sys
from gurobipy import *

if len(sys.argv) < 2:
    print 'Usage: lpmod.py filename'
    quit()

# Read model and determine whether it is an LP

model = read(sys.argv[1])
if model.isMIP == 1:
    print 'The model is not a linear program'
    exit(1)

model.optimize()

status = model.status

if status == GRB.status.INF_OR_UNBD or status == GRB.status.INFEASIBLE
\
   or status == GRB.status.UNBOUNDED:
    print 'The model cannot be solved because it is infeasible or
unbounded'
    exit(1)

if status != GRB.status.OPTIMAL:
    print 'Optimization was stopped with status', status
    exit(0)

# Find the smallest variable value
minVal = GRB.INFINITY
for v in model.getVars():
    if v.x > 0.0001 and v.x < minVal and v.lb == 0.0:
        minVal = v.x
        minVar = v

print '\n*** Setting', minVar.varName, 'from', minVal, 'to zero ***\n'
minVar.ub = 0.0

# Solve from this starting point
model.optimize()
```

```
# Save iteration & time info
warmCount = model.IterCount
warmTime = model.Runtime

# Reset the model and resolve
print '\n*** Resetting and solving without an advanced start ***\n'
model.reset()
model.optimize()

coldCount = model.IterCount
coldTime = model.Runtime

print
print '*** Warm start:', warmCount, 'iterations,', warmTime, 'seconds'
print '*** Cold start:', coldCount, 'iterations,', coldTime, 'seconds'
```

页面：mip1.py

mip1.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# This example formulates and solves the following simple MIP model:
#  maximize
#        x +   y + 2 z
#  subject to
#        x + 2 y + 3 z <= 4
#        x +   y       >= 1
#  x, y, z binary

from gurobipy import *

try:

    # Create a new model
    m = Model("mip1")

    # Create variables
    x = m.addVar(vtype=GRB.BINARY, name="x")
    y = m.addVar(vtype=GRB.BINARY, name="y")
    z = m.addVar(vtype=GRB.BINARY, name="z")

    # Integrate new variables
    m.update()

    # Set objective
    m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)

    # Add constraint: x + 2 y + 3 z <= 4
    m.addConstr(x + 2 * y + 3 * z <= 4, "c0")

    # Add constraint: x + y >= 1
    m.addConstr(x + y >= 1, "c1")

    m.optimize()

    for v in m.getVars():
        print v.varName, v.x

    print 'Obj:', m.objVal

except GurobiError:
    print 'Error reported'
```

页面：mip2.py

mip2.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# This example reads a MIP model from a file, solves it and prints
# the objective values from all feasible solutions generated while
# solving the MIP. Then it creates the associated fixed model and
# solves that model.

import sys
from gurobipy import *

if len(sys.argv) < 2:
    print 'Usage: mip2.py filename'
    quit()

# Read and solve model

model = read(sys.argv[1])

if model.isMIP == 0:
    print 'Model is not a MIP'
    exit(0)

model.optimize()

if model.status == GRB.status.OPTIMAL:
    print 'Optimal objective:', model.objVal
elif model.status == GRB.status.INF_OR_UNBD:
    print 'Model is infeasible or unbounded'
    exit(0)
elif model.status == GRB.status.INFEASIBLE:
    print 'Model is infeasible'
    exit(0)
elif model.status == GRB.status.UNBOUNDED:
    print 'Model is unbounded'
    exit(0)
else:
    print 'Optimization ended with status', model.status
    exit(0)

# Iterate over the solutions and compute the objectives
model.params.outputFlag = 0
print
for k in range(model.solCount):
    model.params.solutionNumber = k
    objn = 0
    for v in model.getVars():
        objn += v.obj * v.xn
    print 'Solution', k, 'has objective:', objn
print
model.params.outputFlag = 1
```

```
fixed = model.fixed()
fixed.params.presolve = 0
fixed.optimize()

if fixed.status != GRB.status.OPTIMAL:
    print "Error: fixed model isn't optimal"
    exit(1)

diff = model.objVal - fixed.objVal

if abs(diff) > 1e-6 * (1.0 + abs(model.objVal)):
    print 'Error: objective values are different'
    exit(1)

# Print values of nonzero variables
for v in fixed.getVars():
    if v.x != 0:
        print v.varName, v.x
```

页面：netflow.py

netflow.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Solve a multi-commodity flow problem.  Two products ('Pencils' and
'Pens')
# are produced in 2 cities ('Detroit' and 'Denver') and must be sent to
# warehouses in 3 cities ('Boston', 'New York', and 'Seattle') to
# satisfy demand ('inflow[h,i]').
#
# Flows on the transportation network must respect arc capacity
constraints
# ('capacity[i,j]'). The objective is to minimize the sum of the arc
# transportation costs ('cost[i,j]').

from gurobipy import *

# Model data

commodities = ['Pencils', 'Pens']
nodes = ['Detroit', 'Denver', 'Boston', 'New York', 'Seattle']

arcs, capacity = multidict({
  ('Detroit', 'Boston'):   100,
  ('Detroit', 'New York'):  80,
  ('Detroit', 'Seattle'):  120,
  ('Denver',  'Boston'):   120,
  ('Denver',  'New York'): 120,
  ('Denver',  'Seattle'):  120 })
arcs = tuplelist(arcs)

cost = {
  ('Pencils', 'Detroit', 'Boston'):   10,
  ('Pencils', 'Detroit', 'New York'): 20,
  ('Pencils', 'Detroit', 'Seattle'):  60,
  ('Pencils', 'Denver',  'Boston'):   40,
  ('Pencils', 'Denver',  'New York'): 40,
  ('Pencils', 'Denver',  'Seattle'):  30,
  ('Pens',    'Detroit', 'Boston'):   20,
  ('Pens',    'Detroit', 'New York'): 20,
  ('Pens',    'Detroit', 'Seattle'):  80,
  ('Pens',    'Denver',  'Boston'):   60,
  ('Pens',    'Denver',  'New York'): 70,
  ('Pens',    'Denver',  'Seattle'):  30 }

inflow = {
  ('Pencils', 'Detroit'):   50,
  ('Pencils', 'Denver'):    60,
  ('Pencils', 'Boston'):   -50,
  ('Pencils', 'New York'): -50,
  ('Pencils', 'Seattle'):  -10,
  ('Pens',    'Detroit'):   60,
  ('Pens',    'Denver'):    40,
  ('Pens',    'Boston'):   -40,
  ('Pens',    'New York'): -30,
  ('Pens',    'Seattle'):  -30 }

# Create optimization model
m = Model('netflow')

# Create variables
flow = {}
for h in commodities:
    for i,j in arcs:
        flow[h,i,j] = m.addVar(ub=capacity[i,j], obj=cost[h,i,j],
                               name='flow_%s_%s_%s' % (h, i, j))
m.update()

# Arc capacity constraints
for i,j in arcs:
    m.addConstr(quicksum(flow[h,i,j] for h in commodities) <=
capacity[i,j],
                'cap_%s_%s' % (i, j))

# Flow conservation constraints
for h in commodities:
    for j in nodes:
        m.addConstr(
          quicksum(flow[h,i,j] for i,j in arcs.select('*',j)) +
              inflow[h,j] ==
          quicksum(flow[h,j,k] for j,k in arcs.select(j,'*')),
                   'node_%s_%s' % (h, j))

# Compute optimal solution
m.optimize()

# Print solution
if m.status == GRB.status.OPTIMAL:
    for h in commodities:
        print '\nOptimal flows for', h, ':'
        for i,j in arcs:
            if flow[h,i,j].x > 0:
                print i, '->', j, ':', flow[h,i,j].x
```

页面：params.py

params.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Use parameters that are associated with a model.
#
# A MIP is solved for 5 seconds with different sets of parameters.
# The one with the smallest MIP gap is selected, and the optimization
# is resumed until the optimal solution is found.

import sys
from gurobipy import *

if len(sys.argv) < 2:
    print 'Usage: params.py filename'
    quit()


# Simple function to determine the MIP gap
def gap(model):
    if model.solCount == 0 or abs(model.objVal) < 1e-6:
        return GRB.INFINITY
    return abs(model.objBound - model.objVal)/abs(model.objVal)

# Read model and verify that it is a MIP
base = read(sys.argv[1])
if base.isMIP == 0:
    print 'The model is not an integer program'
    exit(1)

# Set a 5 second time limit
base.params.timeLimit = 5

# Now solve the model with different values of MIPFocus
bestGap = GRB.INFINITY
bestModel = None
for i in range(4):
    m = base.copy()
    m.params.MIPFocus = i
    m.optimize()
    if bestModel == None or bestGap > gap(m):
        bestModel = m
        bestGap = gap(bestModel)

# Finally, reset the time limit and continue to solve the
# best model to optimality
bestModel.params.timeLimit = "default"
bestModel.optimize()
print 'Solved with MIPFocus:', bestModel.params.MIPFocus
```

页面：qp1.py

qp1.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# This example formulates and solves the following simple QP model:
#  minimize
#      x^2 + x*y + y^2 + y*z + z^2
#  subject to
#      x + 2 y + 3 z >= 4
#      x +   y       >= 1
#
# It solves it once as a continuous model, and once as an integer model.

from gurobipy import *

# Create a new model
m = Model("qp1")

# Create variables
x = m.addVar(ub=1.0, name="x")
y = m.addVar(ub=1.0, name="y")
z = m.addVar(ub=1.0, name="z")

# Integrate new variables
m.update()

# Set objective: x^2 + x*y + y^2 + y*z + z^2
obj = x*x + x*y + y*y + y*z + z*z
m.setObjective(obj)

# Add constraint: x + 2 y + 3 z <= 4
m.addConstr(x + 2 * y + 3 * z >= 4, "c0")

# Add constraint: x + y >= 1
m.addConstr(x + y >= 1, "c1")

m.optimize()

for v in m.getVars():
    print v.varName, v.x

print 'Obj:', obj.getValue()

x.vType = GRB.INTEGER
y.vType = GRB.INTEGER
z.vType = GRB.INTEGER

m.optimize()

for v in m.getVars():
    print v.varName, v.x
print 'Obj:', obj.getValue()
```

页面：sensitivity.py

sensitivity.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Simple MIP sensitivity analysis example.
# For each integer variable, fix it to its lower and upper bound
# and check the impact on the objective.

import sys
from gurobipy import *

if len(sys.argv) < 2:
    print 'Usage: sensitivity.py filename'
    quit()


a = gurobi.read(sys.argv[1])
a.optimize()
a.params.outputFlag = 0

# Extract variables from model

avars = a.getVars()

# Iterate through binary variables in model

for i in range(len(avars)):
    v = avars[i]
    if v.vType == GRB.BINARY:

# Create clone and fix variable

        b = a.copy()
        bv = b.getVars()[i]
        if v.x - v.lb < 0.5:
            bv.lb = bv.ub
        else:
            bv.ub = bv.lb

        b.optimize()

        if b.status == GRB.status.OPTIMAL:
            objchg = b.objVal - a.objVal
            if objchg < 0:
                objchg = 0
            print 'Objective sensitivity for variable', v.varName, 'is',
objchg
        else:
            print 'Objective sensitivity for variable', v.varName, \
                  'is infinite'
```

页面：sos.py

sos.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# This example creates a very simple Special Ordered Set (SOS) model.
# The model consists of 3 continuous variables, no linear constraints,
# and a pair of SOS constraints of type 1.

from gurobipy import *

try:

    # Create a new model

    model = Model("sos")

    # Create variables

    x0 = model.addVar(ub=1.0, name="x0")
    x1 = model.addVar(ub=1.0, name="x1")
    x2 = model.addVar(ub=2.0, name="x2")

    # Integrate new variables

    model.update()

    # Set objective
    model.setObjective(2 * x0 + x1 + x2, GRB.MAXIMIZE)

    # Add first SOS: x0 = 0 or x1 = 0
    model.addSOS(GRB.SOS_TYPE1, [x0, x1], [1, 2])

    # Add second SOS: x0 = 0 or x2 = 0
    model.addSOS(GRB.SOS_TYPE1, [x0, x2], [1, 2])

    model.optimize()

    for v in model.getVars():
        print v.varName, v.x

    print 'Obj:', model.objVal

except GurobiError:

    print 'Encountered an error'
```

页面：sudoku.py

sudoku.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Sudoku example.

# The Sudoku board is a 9x9 grid, which is further divided into a 3x3
grid
# of 3x3 grids.  Each cell in the grid must take a value from 0 to 9.
# No two grid cells in the same row, column, or 3x3 subgrid may take
the
# same value.
#
# In the MIP formulation, binary variables x[i,j,v] indicate whether
# cell <i,j> takes value 'v'.  The constraints are as follows:
#   1. Each cell must take exactly one value (sum_v x[i,j,v] = 1)
#   2. Each value is used exactly once per row (sum_i x[i,j,v] = 1)
#   3. Each value is used exactly once per column (sum_j x[i,j,v] = 1)
#   4. Each value is used exactly once per 3x3 subgrid (sum_grid
x[i,j,v] = 1)

import sys
import math
from gurobipy import *

if len(sys.argv) < 2:
    print 'Usage: sudoku.py filename'
    quit()

f = open(sys.argv[1])

grid = f.read().split()

n = len(grid[0])
s = int(math.sqrt(n))


# 3-D array of variables will be indexed by (i,j,v) tuples

vars = {}


# Create our 3-D array of model variables

model = Model('sudoku')

for i in range(n):
    for j in range(n):
        for v in range(n):
            vars[i,j,v] = model.addVar(vtype=GRB.BINARY,
                                       name='G_'+
str(i)+'_'+str(j)+'_'+str(v))
```

```
# Update model to integrate new variables

model.update()


# Fix variables associated with cells whose values are pre-specified

for i in range(n):
    for j in range(n):
        if grid[i][j] != '.':
            v = int(grid[i][j]) - 1
            model.addConstr(vars[i,j,v] == 1, 'Fix_' + str(i) + '_' +
str(j))

# Each cell must take one value

for i in range(n):
    for j in range(n):
        model.addConstr(quicksum([vars[i,j,v] for v in range(n)]) == 1,
                        'V_' + str(i) + '_' + str(j))

# Each value appears once per row

for i in range(n):
    for v in range(n):
        model.addConstr(quicksum([vars[i,j,v] for j in range(n)]) == 1,
                        'R_' + str(i) + '_' + str(v))

# Each value appears once per column

for j in range(n):
    for v in range(n):
        model.addConstr(quicksum([vars[i,j,v] for i in range(n)]) == 1,
                        'C_' + str(j) + '_' + str(v))

# Each value appears once per subgrid

for v in range(n):
    for i0 in range(s):
        for j0 in range(s):
            subgrid = [vars[i,j,v] for i in range(i0*s, (i0+1)*s)
                                   for j in range(j0*s, (j0+1)*s)]
            model.addConstr(quicksum(subgrid) == 1,
                            'Sub_' + str(i0) + '_' + str(j0) + '_' +
str(v))

model.optimize()

model.write('sudoku.lp')

print
print 'Solution:'
print

for i in range(n):
    sol = ''
```

```
for j in range(n):
    for v in range(n):
        if vars[i,j,v].x > 0.5:
            sol += str(v+1)
print sol
```

页面：workforce1.py

workforce1.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Assign workers to shifts; each worker may or may not be available on
a
# particular day. If the problem cannot be solved, use IIS to find a
set of
# conflicting constraints. Note that there may be additional conflicts
besides
# what is reported via IIS.

from gurobipy import *

# Number of workers required for each shift
shifts, shiftRequirements = multidict({
  "Mon1":  3,
  "Tue2":  2,
  "Wed3":  4,
  "Thu4":  4,
  "Fri5":  5,
  "Sat6":  6,
  "Sun7":  5,
  "Mon8":  2,
  "Tue9":  2,
  "Wed10": 3,
  "Thu11": 4,
  "Fri12": 6,
  "Sat13": 7,
  "Sun14": 5 })

# Amount each worker is paid to work one shift
workers, pay = multidict({
  "Amy":   10,
  "Bob":   12,
  "Cathy": 10,
  "Dan":   8,
  "Ed":    8,
  "Fred":  9,
  "Gu":    11 })

# Worker availability
availability = tuplelist([
('Amy', 'Tue2'), ('Amy', 'Wed3'), ('Amy', 'Fri5'), ('Amy', 'Sun7'),
('Amy', 'Tue9'), ('Amy', 'Wed10'), ('Amy', 'Thu11'), ('Amy', 'Fri12'),
('Amy', 'Sat13'), ('Amy', 'Sun14'), ('Bob', 'Mon1'), ('Bob', 'Tue2'),
('Bob', 'Fri5'), ('Bob', 'Sat6'), ('Bob', 'Mon8'), ('Bob', 'Thu11'),
('Bob', 'Sat13'), ('Cathy', 'Wed3'), ('Cathy', 'Thu4'), ('Cathy',
'Fri5'),
('Cathy', 'Sun7'), ('Cathy', 'Mon8'), ('Cathy', 'Tue9'), ('Cathy',
'Wed10'),
('Cathy', 'Thu11'), ('Cathy', 'Fri12'), ('Cathy', 'Sat13'),
('Cathy', 'Sun14'), ('Dan', 'Tue2'), ('Dan', 'Wed3'), ('Dan', 'Fri5'),
('Dan', 'Sat6'), ('Dan', 'Mon8'), ('Dan', 'Tue9'), ('Dan', 'Wed10'),
('Dan', 'Thu11'), ('Dan', 'Fri12'), ('Dan', 'Sat13'), ('Dan', 'Sun14'),
('Ed', 'Mon1'), ('Ed', 'Tue2'), ('Ed', 'Wed3'), ('Ed', 'Thu4'),
('Ed', 'Fri5'), ('Ed', 'Sun7'), ('Ed', 'Mon8'), ('Ed', 'Tue9'),
('Ed', 'Thu11'), ('Ed', 'Sat13'), ('Ed', 'Sun14'), ('Fred', 'Mon1'),
('Fred', 'Tue2'), ('Fred', 'Wed3'), ('Fred', 'Sat6'), ('Fred', 'Mon8'),
('Fred', 'Tue9'), ('Fred', 'Fri12'), ('Fred', 'Sat13'), ('Fred',
'Sun14'),
('Gu', 'Mon1'), ('Gu', 'Tue2'), ('Gu', 'Wed3'), ('Gu', 'Fri5'),
('Gu', 'Sat6'), ('Gu', 'Sun7'), ('Gu', 'Mon8'), ('Gu', 'Tue9'),
('Gu', 'Wed10'), ('Gu', 'Thu11'), ('Gu', 'Fri12'), ('Gu', 'Sat13'),
('Gu', 'Sun14')
])

# Model
m = Model("assignment")

# Assignment variables: x[w,s] == 1 if worker w is assigned to shift s.
# Since an assignment model always produces integer solutions, we use
# continuous variables and solve as an LP.
x = {}
for w,s in availability:
    x[w,s] = m.addVar(ub=1, obj=pay[w], name=w+"."+s)

# The objective is to minimize the total pay costs
m.modelSense = GRB.MINIMIZE

# Update model to integrate new variables
m.update()

# Constraint: assign exactly shiftRequirements[s] workers to each shift
s
reqCts = {}
for s in shifts:
    reqCts[s] = m.addConstr(
      quicksum(x[w,s] for w,s in availability.select('*', s)) ==
        shiftRequirements[s], s)

# Optimize
m.optimize()
status = m.status
if status == GRB.status.UNBOUNDED:
    print 'The model cannot be solved because it is unbounded'
    exit(0)
if status == GRB.status.OPTIMAL:
    print 'The optimal objective is', m.objVal
    exit(0)
if status != GRB.status.INF_OR_UNBD and status != GRB.status.INFEASIBLE:
    print 'Optimization was stopped with status', status
    exit(0)

# do IIS
print 'The model is infeasible; computing IIS'
m.computeIIS()
print '\nThe following constraint(s) cannot be satisfied:'
for c in m.getConstrs():
```

```
if c.IISConstr:
    print c.constrName
```

页面：workforce2.py

workforce2.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Assign workers to shifts; each worker may or may not be available on
a
# particular day. If the problem cannot be solved, use IIS iteratively
to
# find all conflicting constraints.

from gurobipy import *

# Number of workers required for each shift
shifts, shiftRequirements = multidict({
  "Mon1":  3,
  "Tue2":  2,
  "Wed3":  4,
  "Thu4":  4,
  "Fri5":  5,
  "Sat6":  6,
  "Sun7":  5,
  "Mon8":  2,
  "Tue9":  2,
  "Wed10": 3,
  "Thu11": 4,
  "Fri12": 6,
  "Sat13": 7,
  "Sun14": 5 })

# Amount each worker is paid to work one shift
workers, pay = multidict({
  "Amy":   10,
  "Bob":   12,
  "Cathy": 10,
  "Dan":   8,
  "Ed":    8,
  "Fred":  9,
  "Gu":    11 })

# Worker availability
availability = tuplelist([
('Amy', 'Tue2'), ('Amy', 'Wed3'), ('Amy', 'Fri5'), ('Amy', 'Sun7'),
('Amy', 'Tue9'), ('Amy', 'Wed10'), ('Amy', 'Thu11'), ('Amy', 'Fri12'),
('Amy', 'Sat13'), ('Amy', 'Sun14'), ('Bob', 'Mon1'), ('Bob', 'Tue2'),
('Bob', 'Fri5'), ('Bob', 'Sat6'), ('Bob', 'Mon8'), ('Bob', 'Thu11'),
('Bob', 'Sat13'), ('Cathy', 'Wed3'), ('Cathy', 'Thu4'), ('Cathy',
'Fri5'),
('Cathy', 'Sun7'), ('Cathy', 'Mon8'), ('Cathy', 'Tue9'), ('Cathy',
'Wed10'),
('Cathy', 'Thu11'), ('Cathy', 'Fri12'), ('Cathy', 'Sat13'),
('Cathy', 'Sun14'), ('Dan', 'Tue2'), ('Dan', 'Wed3'), ('Dan', 'Fri5'),
('Dan', 'Sat6'), ('Dan', 'Mon8'), ('Dan', 'Tue9'), ('Dan', 'Wed10'),
('Dan', 'Thu11'), ('Dan', 'Fri12'), ('Dan', 'Sat13'), ('Dan', 'Sun14'),
('Ed', 'Mon1'), ('Ed', 'Tue2'), ('Ed', 'Wed3'), ('Ed', 'Thu4'),
('Ed', 'Fri5'), ('Ed', 'Sun7'), ('Ed', 'Mon8'), ('Ed', 'Tue9'),
('Ed', 'Thu11'), ('Ed', 'Sat13'), ('Ed', 'Sun14'), ('Fred', 'Mon1'),
('Fred', 'Tue2'), ('Fred', 'Wed3'), ('Fred', 'Sat6'), ('Fred', 'Mon8'),
('Fred', 'Tue9'), ('Fred', 'Fri12'), ('Fred', 'Sat13'), ('Fred',
'Sun14'),
('Gu', 'Mon1'), ('Gu', 'Tue2'), ('Gu', 'Wed3'), ('Gu', 'Fri5'),
('Gu', 'Sat6'), ('Gu', 'Sun7'), ('Gu', 'Mon8'), ('Gu', 'Tue9'),
('Gu', 'Wed10'), ('Gu', 'Thu11'), ('Gu', 'Fri12'), ('Gu', 'Sat13'),
('Gu', 'Sun14')
])

# Model
m = Model("assignment")

# Assignment variables: x[w,s] == 1 if worker w is assigned to shift s.
# Since an assignment model always produces integer solutions, we use
# continuous variables and solve as an LP.
x = {}
for w,s in availability:
    x[w,s] = m.addVar(ub=1, obj=pay[w], name=w+"."+s)

# The objective is to minimize the total pay costs
m.modelSense = GRB.MINIMIZE

# Update model to integrate new variables
m.update()

# Constraint: assign exactly shiftRequirements[s] workers to each shift
s
reqCts = {}
for s in shifts:
    reqCts[s] = m.addConstr(
      quicksum(x[w,s] for w,s in availability.select('*', s)) ==
        shiftRequirements[s], s)

# Optimize
m.optimize()
status = m.status
if status == GRB.status.UNBOUNDED:
    print 'The model cannot be solved because it is unbounded'
    exit(0)
if status == GRB.status.OPTIMAL:
    print 'The optimal objective is', m.objVal
    exit(0)
if status != GRB.status.INF_OR_UNBD and status != GRB.status.INFEASIBLE:
    print 'Optimization was stopped with status', status
    exit(0)

# do IIS
print 'The model is infeasible; computing IIS'
removed = []

# Loop until we reduce to a model that can be solved
while True:
    m.computeIIS()
    print '\nThe following constraint cannot be satisfied:'
    for c in m.getConstrs():
        if c.IISConstr:
            print c.constrName
            # Remove a single constraint from the model
            removed.append(c.constrName)
            m.remove(c)
            break
    print

    m.optimize()
    status = m.status

    if status == GRB.status.UNBOUNDED:
        print 'The model cannot be solved because it is unbounded'
        exit(0)
    if status == GRB.status.OPTIMAL:
        break
    if status != GRB.status.INF_OR_UNBD and status !=
GRB.status.INFEASIBLE:
        print 'Optimization was stopped with status', status
        exit(0)

print '\nThe following constraints were removed to get a feasible LP:'
print removed
```

页面：workforce3.py

workforce3.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Assign workers to shifts; each worker may or may not be available on
a
# particular day. If the problem cannot be solved, add slack variables
# to determine which constraints cannot be satisfied, and how much
# they need to be relaxed.

from gurobipy import *

# Number of workers required for each shift
shifts, shiftRequirements = multidict({
  "Mon1":  3,
  "Tue2":  2,
  "Wed3":  4,
  "Thu4":  4,
  "Fri5":  5,
  "Sat6":  6,
  "Sun7":  5,
  "Mon8":  2,
  "Tue9":  2,
  "Wed10": 3,
  "Thu11": 4,
  "Fri12": 6,
  "Sat13": 7,
  "Sun14": 5 })

# Amount each worker is paid to work one shift
workers, pay = multidict({
  "Amy":   10,
  "Bob":   12,
  "Cathy": 10,
  "Dan":   8,
  "Ed":    8,
  "Fred":  9,
  "Gu":    11 })

# Worker availability
availability = tuplelist([
('Amy', 'Tue2'), ('Amy', 'Wed3'), ('Amy', 'Fri5'), ('Amy', 'Sun7'),
('Amy', 'Tue9'), ('Amy', 'Wed10'), ('Amy', 'Thu11'), ('Amy', 'Fri12'),
('Amy', 'Sat13'), ('Amy', 'Sun14'), ('Bob', 'Mon1'), ('Bob', 'Tue2'),
('Bob', 'Fri5'), ('Bob', 'Sat6'), ('Bob', 'Mon8'), ('Bob', 'Thu11'),
('Bob', 'Sat13'), ('Cathy', 'Wed3'), ('Cathy', 'Thu4'), ('Cathy',
'Fri5'),
('Cathy', 'Sun7'), ('Cathy', 'Mon8'), ('Cathy', 'Tue9'), ('Cathy',
'Wed10'),
('Cathy', 'Thu11'), ('Cathy', 'Fri12'), ('Cathy', 'Sat13'),
('Cathy', 'Sun14'), ('Dan', 'Tue2'), ('Dan', 'Wed3'), ('Dan', 'Fri5'),
('Dan', 'Sat6'), ('Dan', 'Mon8'), ('Dan', 'Tue9'), ('Dan', 'Wed10'),
('Dan', 'Thu11'), ('Dan', 'Fri12'), ('Dan', 'Sat13'), ('Dan', 'Sun14'),
('Ed', 'Mon1'), ('Ed', 'Tue2'), ('Ed', 'Wed3'), ('Ed', 'Thu4'),
('Ed', 'Fri5'), ('Ed', 'Sun7'), ('Ed', 'Mon8'), ('Ed', 'Tue9'),
('Ed', 'Thu11'), ('Ed', 'Sat13'), ('Ed', 'Sun14'), ('Fred', 'Mon1'),
('Fred', 'Tue2'), ('Fred', 'Wed3'), ('Fred', 'Sat6'), ('Fred', 'Mon8'),
('Fred', 'Tue9'), ('Fred', 'Fri12'), ('Fred', 'Sat13'), ('Fred',
'Sun14'),
('Gu', 'Mon1'), ('Gu', 'Tue2'), ('Gu', 'Wed3'), ('Gu', 'Fri5'),
('Gu', 'Sat6'), ('Gu', 'Sun7'), ('Gu', 'Mon8'), ('Gu', 'Tue9'),
('Gu', 'Wed10'), ('Gu', 'Thu11'), ('Gu', 'Fri12'), ('Gu', 'Sat13'),
('Gu', 'Sun14')
])

# Model
m = Model("assignment")

# Assignment variables: x[w,s] == 1 if worker w is assigned to shift s.
# Since an assignment model always produces integer solutions, we use
# continuous variables and solve as an LP.
x = {}
for w,s in availability:
    x[w,s] = m.addVar(ub=1, obj=pay[w], name=w+"."+s)

# The objective is to minimize the total pay costs
m.modelSense = GRB.MINIMIZE

# Update model to integrate new variables
m.update()

# Constraint: assign exactly shiftRequirements[s] workers to each shift
s
reqCts = {}
for s in shifts:
    reqCts[s] = m.addConstr(
      quicksum(x[w,s] for w,s in availability.select('*', s)) ==
        shiftRequirements[s], s)

# Optimize
m.optimize()
status = m.status
if status == GRB.status.UNBOUNDED:
    print 'The model cannot be solved because it is unbounded'
    exit(0)
if status == GRB.status.OPTIMAL:
    print 'The optimal objective is', m.objVal
    exit(0)
if status != GRB.status.INF_OR_UNBD and status != GRB.status.INFEASIBLE:
    print 'Optimization was stopped with status', status
    exit(0)


# Add slack variables to make the model feasible
print 'The model is infeasible; adding slack variables'

# Set original objective coefficients to zero
m.setObjective(0.0)
```

```
# Add a new slack variable to each shift constraint so that the shifts
can
# be satisfied
slacks = {}
for s,c in reqCts.items():
    slacks[s] = m.addVar(obj=1.0, name=s+"Slack", column=Column([1],
[c]))

# Solve the model with slacks
m.optimize()
status = m.status
if status == GRB.status.INF_OR_UNBD or status == GRB.status.INFEASIBLE
\
   or status == GRB.status.UNBOUNDED:
    print 'The model with slacks cannot be solved ' \
          'because it is infeasible or unbounded'
    exit(1)

if status != GRB.status.OPTIMAL:
    print 'Optimization was stopped with status', status
    exit(1)

print '\nSlack values:'
for sv in slacks.values():
    if sv.x > 1e-6:
        print sv.varName, '=', sv.x
```

页面：workforce4.py

workforce4.py

```
#!/usr/bin/python

# Copyright 2011, Gurobi Optimization, Inc.

# Assign workers to shifts; each worker may or may not be available on
a
# particular day. We use lexicographic optimization to solve the model:
# first, we minimize the linear sum of the slacks. Then, we constrain
# the sum of the slacks, and we minimize a quadratic objective that
# tries to balance the workload among the workers.

from gurobipy import *

# Number of workers required for each shift
shifts, shiftRequirements = multidict({
  "Mon1":  3,
  "Tue2":  2,
  "Wed3":  4,
  "Thu4":  4,
  "Fri5":  5,
  "Sat6":  6,
  "Sun7":  5,
  "Mon8":  2,
  "Tue9":  2,
  "Wed10": 3,
  "Thu11": 4,
  "Fri12": 6,
  "Sat13": 7,
  "Sun14": 5 })

# Amount each worker is paid to work one shift
workers, pay = multidict({
  "Amy":   10,
  "Bob":   12,
  "Cathy": 10,
  "Dan":   8,
  "Ed":    8,
  "Fred":  9,
  "Gu":    11 })

# Worker availability
availability = tuplelist([
('Amy', 'Tue2'), ('Amy', 'Wed3'), ('Amy', 'Fri5'), ('Amy', 'Sun7'),
('Amy', 'Tue9'), ('Amy', 'Wed10'), ('Amy', 'Thu11'), ('Amy', 'Fri12'),
('Amy', 'Sat13'), ('Amy', 'Sun14'), ('Bob', 'Mon1'), ('Bob', 'Tue2'),
('Bob', 'Fri5'), ('Bob', 'Sat6'), ('Bob', 'Mon8'), ('Bob', 'Thu11'),
('Bob', 'Sat13'), ('Cathy', 'Wed3'), ('Cathy', 'Thu4'), ('Cathy',
'Fri5'),
('Cathy', 'Sun7'), ('Cathy', 'Mon8'), ('Cathy', 'Tue9'), ('Cathy',
'Wed10'),
('Cathy', 'Thu11'), ('Cathy', 'Fri12'), ('Cathy', 'Sat13'),
('Cathy', 'Sun14'), ('Dan', 'Tue2'), ('Dan', 'Wed3'), ('Dan', 'Fri5'),
('Dan', 'Sat6'), ('Dan', 'Mon8'), ('Dan', 'Tue9'), ('Dan', 'Wed10'),
('Dan', 'Thu11'), ('Dan', 'Fri12'), ('Dan', 'Sat13'), ('Dan', 'Sun14'),
('Ed', 'Mon1'), ('Ed', 'Tue2'), ('Ed', 'Wed3'), ('Ed', 'Thu4'),
('Ed', 'Fri5'), ('Ed', 'Sun7'), ('Ed', 'Mon8'), ('Ed', 'Tue9'),
('Ed', 'Thu11'), ('Ed', 'Sat13'), ('Ed', 'Sun14'), ('Fred', 'Mon1'),
('Fred', 'Tue2'), ('Fred', 'Wed3'), ('Fred', 'Sat6'), ('Fred', 'Mon8'),
('Fred', 'Tue9'), ('Fred', 'Fri12'), ('Fred', 'Sat13'), ('Fred',
'Sun14'),
('Gu', 'Mon1'), ('Gu', 'Tue2'), ('Gu', 'Wed3'), ('Gu', 'Fri5'),
('Gu', 'Sat6'), ('Gu', 'Sun7'), ('Gu', 'Mon8'), ('Gu', 'Tue9'),
('Gu', 'Wed10'), ('Gu', 'Thu11'), ('Gu', 'Fri12'), ('Gu', 'Sat13'),
('Gu', 'Sun14')
])

# Model
m = Model("assignment")

# Assignment variables: x[w,s] == 1 if worker w is assigned to shift s.
# This is no longer a pure assignment model, so we must use binary
variables.
x = {}
for w,s in availability:
    x[w,s] = m.addVar(vtype=GRB.BINARY, obj=pay[w], name=w+"."+s)

# Slack variables for each shift constraint so that the shifts can
# be satisfied
slacks = {}
for s in shifts:
    slacks[s] = m.addVar(name=s+"Slack")

# Variable to represent the total slack
totSlack = m.addVar(name="totSlack")

# Variables to count the total shifts worked by each worker
totShifts = {}
for w in workers:
    totShifts[w] = m.addVar(name=w+"TotShifts")

# Update model to integrate new variables
m.update()

# Constraint: assign exactly shiftRequirements[s] workers to each shift
s,
# plus the slack
for s in shifts:
    m.addConstr(slacks[s] +
      quicksum(x[w,s] for w,s in availability.select('*', s)) ==
        shiftRequirements[s], s)

# Constraint: set totSlack equal to the total slack
m.addConstr(totSlack == quicksum(slacks[s] for s in shifts), "totSlack")

# Constraint: compute the total number of shifts for each worker
for w in workers:
    m.addConstr(totShifts[w] ==
      quicksum(x[w,s] for w,s in availability.select(w, '*')),
      "totShifts" + w)
```

```
# Objective: minimize the total slack
# Note that this replaces the previous 'pay' objective coefficients
m.setObjective(totSlack)


# Optimize
def solveAndPrint():
    m.optimize()
    status = m.status
    if status == GRB.status.INF_OR_UNBD or status ==
GRB.status.INFEASIBLE \
      or status == GRB.status.UNBOUNDED:
        print 'The model cannot be solved because it is infeasible or \
               unbounded'
        exit(1)

    if status != GRB.status.OPTIMAL:
        print 'Optimization was stopped with status', status
        exit(0)

    # Print total slack and the number of shifts worked for each worker
    print
    print 'Total slack required:', totSlack.x
    for w in workers:
        print w, "worked", totShifts[w].x, "shifts"
    print

solveAndPrint()

# Constrain the slack by setting its upper and lower bounds
totSlack.ub = totSlack.x
totSlack.lb = totSlack.x

# Variable to count the average number of shifts worked
avgShifts = m.addVar(name="avgShifts")

# Variables to count the difference from average for each worker;
# note that these variables can take negative values.
diffShifts = {}
for w in workers:
    diffShifts[w] = \
      m.addVar(lb=-GRB.INFINITY, ub=GRB.INFINITY, name=w+"Diff")

# Update model to integrate new variables
m.update()

# Constraint: compute the average number of shifts worked
m.addConstr(len(workers) * avgShifts ==
              quicksum(totShifts[w] for w in workers),
            "avgShifts")

# Constraint: compute the difference from the average number of shifts
for w in workers:
    m.addConstr(diffShifts[w] == totShifts[w] - avgShifts, w + "Diff")

# Objective: minimize the sum of the square of the difference from the
# average number of shifts worked
m.setObjective(quicksum(diffShifts[w]*diffShifts[w] for w in workers))

# Optimize
solveAndPrint()
```