Посвящается пользователям Ryoto –
прошлым, настоящим и будущим

Содержание

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Благодарности

Здесь будут фамилии тех, кто помогал изданию этой книги, прислав в издательство найденные ошибки или ссылку на подозрительные материалы.

Предисловие

В этой книге описывается инструмент математического моделирования – программа Python Optimization Modeling Objects (Pyomo). Pyomo поддерживает создание и анализ математических моделей для сложных приложений оптимизации. Обычно эта способность ассоциируется с языками алгебраического моделирования (algebraic modeling languages – AML) высокого уровня. Большинство AML реализованы как самостоятельные языки моделирования, но объекты моделирования Pyomo встроены в Python, полноценный язык программирования высокого уровня, для которого имеется обширный набор библиотек. Pyomo получил награды организации R&D100 и INFORMS Computing Society.

Моделирование – фундаментальный процесс во многих аспектах научных исследований, техники и бизнеса, а благодаря широкому распространению вычислительной техники численный анализ математических моделей стал обыденностью. Ко всему прочему основным принципом языков AML стала робастная формулировка больших моделей для сложных приложений, встречающихся на практике [37]. AML облегчили процесс формулирования моделей, упростив управление разреженными данными и добавив поддержку естественного выражения компонентов модели. Дополнительно AML типа Pyomo поддерживают написание скриптов, включающих объекты модели, что дает возможность проводить нестандартный анализ сложных задач.

Основой Pyomo является объектно ориентированное представление моделей оптимизации. Также Pyomo содержит пакеты для определения расширений и переформулирования модели. Кроме того, в состав Pyomo входят пакеты, определяющие интерфейсы с такими решателями, как CPLEX и Gurobi, и службами решения типа NEOS.

Цели этой книги

В третье издание включено переработанное описание средства моделирования Pyomo. Основная цель книги – дать общее описание Pyomo, которое позволило бы пользователям создавать и оптимизировать модели. Поэтому в книге много примеров, иллюстрирующих различные методы формулирования моделей.

Другая цель книги – проиллюстрировать богатство возможностей Pyomo, в том числе формулирование и анализ типичных моделей оптимизации, включая линейное программирование, смешанно-целочисленное линейное программирование, нелинейное программирование, смешанно-целочисленное нелинейное программирование, математическое программирова-

ние с ограничениями равновесия, ограничения и целевые функции, основанные на дифференциальных уравнениях, обобщенное дизъюнктивное программирование. Кроме того, Pyomo включает интерфейсы к различным распространенным пакетам программ оптимизации, как то CBC, CPLEX, GLPK и Gurobi. Модели Pyomo можно оптимизировать с помощью таких программ, как IPOPT, в которой используется интерфейс к библиотеке AMPL Solver Library.

Наконец, книга призвана облегчить знакомство с Pyomo даже тем пользователям, которые мало что знают о Python. Приложение A содержит краткое введение в Python, но мы были поражены тем, насколько справочники по Python полезны новым пользователям Pyomo. Хотя в Pyomo используются объекты Python, выражение моделей на Pyomo следует ясному и лаконичному синтаксису Python.

Однако в нашем обсуждении продвинутых средств моделирования Pyomo предполагается некоторое знакомство с объектно ориентированным проектированием и возможностями языка Python. Например, мы проводим различие между определением класса и его экземплярами. Мы не пытались объяснить эти средства Python в книге. Поэтому от читателя ожидается желание хотя бы немного изучить Python, если он хочет понимать и эффективно использовать продвинутые средства моделирования.

Для кого предназначена эта книга

Книга представляет собой справочное пособие для студентов, научных работников и инженеров-практиков. Структура Pyomo настолько проста, что программу можно использовать в курсах для студентов и аспирантов. Однако предполагается, что читатель знаком с основами оптимизации и математического моделирования. В книге нет глоссария, но мы рекомендуем использовать для справки глоссарий математического программирования [32].

Pyomo также является ценным инструментом для теоретиков и практиков. При разработке Pyomo много внимания было уделено способности поддерживать описание и анализ реальных приложений. А значит, производительность и надежные интерфейсы с решателями занимали не последнее место.

Кроме того, мы надеемся, что исследователи найдут в Pyomo эффективный каркас для разработки высокоуровневых средств оптимизации и анализа. Например, Pyomo лежит в основе пакета оптимизации в условиях неопределенности `mpi-sppy`, в котором используется тот факт, что объекты моделирования встроены в полнофункциональный высокоуровневый язык программирования. Это позволяет прозрачно распараллеливать подзадачи с помощью соответствующих библиотек Python. Этот механизм разработки обобщенных решателей для сложных моделей очень мощный, и мы надеемся, что его можно будет использовать в сочетании со многими другими методами оптимизации.

Благодарности

Мы ценим усилия многих людей, способствовавших выходу этого и предыдущих изданий книги. Мы благодарны Элизабет Лоев из издательства Springer, которая сопровождала книгу от идеи до последних этапов производства; ее энтузиазм заразителен. Мы также благодарим Маделинн Фарбер из Sandia National Laboratories за помощь в юридических нюансах выпуска программного обеспечения с открытым исходным кодом и публикации книги. Наконец, мы признательны Дугу Проуту за разработку логотипов Pyomo, PySP, Pyomo.DAE и Coopr.

Мы в долгу перед всеми, кто тратил свое время и силы на рецензирование книги. Не будь их, книга содержала бы много опечаток и программных ошибок. Итак, спасибо Джеку Инголлу, Зеву Фридману, Харви Гринбергу, Шону Леггу, Анжелике Вонг, Дэниэлю Уорду, Деанне Гарсиа, Эллис Озакиол и Флориану Мадеру. Отдельное спасибо Эмбер Грей-Фенне и Рэнди Бросту.

Особенно мы благодарны растущему сообществу пользователей Pyomo. Ваш интерес к Pyomo и ваш энтузиазм стали самым важным фактором, повлиявшим на наше решение написать эту книгу. Мы благодарим первых приверженцев Pyomo, которые поделились с нами подробными отзывами о структуре и полезности программы, в том числе Фернандо Бадилла, Стивена Чена, Неда Дмитрова. ЮэЮэ Фана, Эрика Хонга, Аллена Холдера, Андреса Ироуме, Дэррила Меландера, Кэрол Мейерс, Пьера Нансель-Пенарда, Мехула Рангвала, Еву Уормингхаус и Дэвида Алдерсона. Ваши отзывы продолжают оказывать важное влияние на дизайн и возможности Pyomo.

Мы также благодарны нашим друзьям из проекта COIN-OR за поддержку Pyomo. Хотя разработка Pyomo ведется в основном на GitHub, наше партнерство с COIN-OR – ключевая часть нашей стратегии, благодаря которой Pyomo остается жизнеспособным проектом с открытым исходным кодом.

Особая благодарность – нашим коллегам, создававшим пакеты для Pyomo: Франсиско Муньосу, Тимоти Эклу, Кэвину Хантеру, Патрику Стилу и Дэниэлю Уорду. Мы также признательны Тому Браунстейну, Дэйву Гэю и Нику Беневидасу за помощь в разработке модулей Python документации по Pyomo.

Авторы выражают благодарность за поддержку, оказанную при создании этой книги: Национальному научному фонду (гранты CBET#0941313 и CBET#0955205), Управлению передовых научных исследований в области вычислений при отделении по науке Министерства энергетики США, проекту ARPA-E Министерства энергетики США в рамках программы интеграции зеленых электрических сетей, Институту проектирования передовых энергетических систем (IDAES) с финансированием от отдела технических систем на основе имитационного моделирования, междисциплинарную исследовательскую программу отделения ископаемого топлива Министерства энергетики США, программу моделирования передовых сетей электроснабжения (AGM) Министерства энергетики США и Лабораторию целенаправленных исследований и разработок в Sandia National Laboratories.

И наконец, мы благодарны своим семьям и друзьям за их терпеливое отношение к нашей страсти к программам оптимизации.

Замечания и вопросы

В этой книге документирована версия Pyomo 6.0. Дополнительную информацию, включая опечатки, смотрите на сайте Pyomo: http://www.pyomo.org.

Исходный код Pyomo размещен на GitHub, а примеры, приведенные в этой книге, находятся в каталоге `pyomo/examples/doc/pyomobook`: https://github.com/Pyomo/pyomo.

На многие вопросы, касающиеся Pyomo, есть ответы на сайте Stack Overflow: https://stackoverflow.com/.

Мы приветствуем отзывы читателей. Направляйте их непосредственно авторам или на форум Pyomo: pyomo-forum@googlegroups.com.

Удачи!

Альбукерк, Нью-Мексико, США	*Майкл Бинум*
Энн Арбор, Мичиган, США	*Гэйб Хакебейл*
Альбукерк, Нью-Мексико, США	*Уильям Харт*
Альбукерк, Нью-Мексико, США	*Карл Лэрд*
Альбукерк, Нью-Мексико, США	*Бетани Николсон*
Альбукерк, Нью-Мексико, США	*Джон Сиирола*
Ливермор, Калифорния, США	*Жан-Поль Уотсон*
Дэвис, Калифорния, США	*Дэвид Л. Вудраф*

5 января 2021 г.

Глава 1

Введение

В этой главе мы познакомимся с Pyomo, инструментом для моделирования и решения задач оптимизации на основе Python, и расскажем, зачем он создавался. Моделирование – фундаментальный процесс во многих аспектах научных исследований, техники и бизнеса. Языки алгебраического моделирования типа Pyomo – это высокоуровневые языки для формулирования и решения математических задач оптимизации. Pyomo – гибкий и расширяемый каркас моделирования, который вбирает в себя и обобщает главные идеи языков алгебраического моделирования, организуя их в контексте широко распространенного языка программирования.

1.1. Языки моделирования для оптимизации

В этой книге описывается инструмент математического моделирования: программный пакет Python Optimization Modeling Objects (Pyomo). Pyomo поддерживает формулирование и анализ математических моделей для сложных задач оптимизации. Обычно эта возможность ассоциируется с коммерческими языками алгебраического моделирования (AML), такими как AIMMS [1], AMPL [2] и GAMS [22]. Pyomo реализует развитый набор средств моделирования и анализа и предоставляет доступ к этим средствам из полнофункционального высокоуровневого языка программирования Python, для которого написано множество библиотек.

Модели оптимизации определяют целевые функции для рассматриваемой системы. Модели можно использовать для исследования компромиссов между различными целевыми функциями, нахождения экстремальных состояний и худших случаев, а также идентификации основных факторов, от которых зависит поведение системы. Поэтому модели оптимизации применяются для анализа широкого круга научных, деловых и технических задач.

Благодаря широкой доступности вычислительных ресурсов численный анализ моделей оптимизации стал обыденным явлением. Для вычислительного анализа модели оптимизации необходимо описать модель и передать ее программе-решателю. В отсутствие языка спецификации таких моделей процесс написания входных файлов, выполнения решателя и получения ре-

зультатов от него оказывается утомительным и чреват ошибками. На это накладывается тот факт, что реальные приложения велики и сложны, так что отлаживать их трудно. Ко всему прочему для решателей имеется много разных форматов ввода, но лишь малая их часть стандартизирована. Таким образом, применение нескольких решателей для анализа одной модели оптимизации влечет за собой дополнительные сложности. Кроме того, верифицировать модель (т. е. проверить, что переданная решателю модель точно выражает намерения разработчика) исключительно трудно, не имея высокоуровневого языка для выражения модели.

AML – это высокоуровневые языки для описания и решения задач оптимизации [26, 37]. Они сводят к минимуму трудности, связанные с анализом моделей, благодаря высокоуровневой спецификации модели. Кроме того, AML предоставляют интерфейсы к внешним программам-решателям, которые используются для анализа задач, и позволяют пользователю взаимодействовать с результатами решателя в контексте этой спецификации.

Специализированные AML, такие как AIMMS [1], AMPL [2, 21] и GAMS [22], представляют собой языки описания модели оптимизации с интуитивно понятным и лаконичным синтаксисом для определения переменных, ограничений и целевых функций. Кроме того, они поддерживают абстрактные понятия: разреженные множества, индексы и алгебраические выражения, без которых не обойтись при описании крупномасштабных реальных задач с тысячами или миллионами ограничений и переменных. Эти AML могут представлять самые разные модели оптимизации и реализуют интерфейсы с различными решателями. В последнее время AML стали поддерживать новые скриптовые возможности, которые позволяют выражать высокоуровневые алгоритмы анализа вместе со спецификацией модели оптимизации.

Альтернативная стратегия – использовать AML, который расширяет какой-нибудь стандартный высокоуровневый язык программирования (в противоположность специализированному проприетарному языку) для формулирования моделей оптимизации, которые затем анализируются решателями, написанными на низкоуровневых языках. Такой двухъязыковой подход сочетает гибкость высокоуровневого языка для формулирования задач оптимизации и эффективность низкоуровневого для численных расчетов. Этот подход завоевывает все большую популярность в программном обеспечении для научных расчетов. Среда оптимизации TOMLAB, написанная на Matlab [57], – один из наиболее зрелых пакетов, в которых он применяется; в Pyomo также используется этот подход. Есть еще примеры расширения стандартных языков программирования типа Java и C++ путем включения AML-конструкций. Так, библиотеки моделирования FlopC++ [19], OptimJ [47] и JuMP [13] поддерживают описание моделей оптимизации с использованием объектно ориентированного проектирования на C++, Java и Julia соответственно. Эти библиотеки частично приносят в жертву интуитивно понятный математический синтаксис специализированного AML, зато позволяют воспользоваться всей гибкостью современных языков программирования высокого уровня. Их дополнительное преимущество заключается в том, что они непосредственно компонуются с высокопроизводительными библио-

теками оптимизации и решателями, что для некоторых приложений может быть существенно.

1.2. Моделирование на Pyomo

Цель Pyomo – предоставить платформу для описания моделей оптимизации, которая воплощает идеи современных AML в контексте, обеспечивающем гибкость, расширяемость, переносимость, открытость и удобство сопровождения. Pyomo – это AML, расширяющий Python путем включения объектов для моделирования оптимизации [30]. Эти объекты можно использовать для задания моделей оптимизации и их преобразования в различные форматы, понятные внешним решателям.

Теперь мы приведем несколько примеров, иллюстрирующих использование Pyomo для описания моделей оптимизации.

1.2.1. Простые примеры

Рассмотрим следующую линейную программу (ЛП):

$$\begin{aligned} &\min\ x_1 + 2x_2 \\ \text{при условиях}\ &3x_1 + 4x_2 \geqslant 1 \\ &2x_1 + 5x_2 \geqslant 2 \\ &x_1, x_2 \geqslant 0 \end{aligned}$$

Эту ЛП легко выразить на Pyomo следующим образом:

```
import pyomo.environ as pyo

model = pyo.ConcreteModel()
model.x_1 = pyo.Var(within=pyo.NonNegativeReals)
model.x_2 = pyo.Var(within=pyo.NonNegativeReals)
model.obj = pyo.Objective(expr=model.x_1 + 2*model.x_2)
model.con1 = pyo.Constraint(expr=3*model.x_1 + 4*model.x_2 >= 1)
model.con2 = pyo.Constraint(expr=2*model.x_1 + 5*model.x_2 >= 2)
```

Первая строка – стандартное предложение импорта в Python, которое инициализирует окружение Pyomo и загружает библиотеку базовых компонентов моделирования. В следующей строке конструируется объект модели и определяются его атрибуты. В этом примере описывается *конкретная модель*. Компоненты модели – объекты, являющиеся атрибутами объекта модели, а объект `ConcreteModel` инициализирует каждый компонент модели при добавлении. Переменные, ограничения и целевая функция модели определяются с помощью *компонентов модели*.

Редко бывает так, что создается единственный экземпляр некоторой модели оптимизации, подлежащей решению. Чаще имеется общая модель оп-

тимизации, а затем создается конкретный экземпляр этой модели с определенными данными. Например, следующие уравнения представляют ЛП со скалярными параметрами n и m, векторными параметрами b и c и матричным параметром a:

$$\begin{aligned} &\min \sum_{i=1}^{n} c_i x_i \\ \text{при условиях } &\sum_{i=1}^{n} a_{ji} x_i \geqslant b_j \qquad \forall j = 1 \ldots m \\ &x_i \geqslant 0 \qquad\qquad\quad \forall i = 1 \ldots n \end{aligned}$$

Эту ЛП можно выразить в виде конкретной модели на Pyomo следующим образом:

```
import pyomo.environ as pyo
import mydata

model = pyo.ConcreteModel()

model.x = pyo.Var(mydata.N, within=pyo.NonNegativeReals)

def obj_rule(model):
    return sum(mydata.c[i]*model.x[i] for i in mydata.N)
model.obj = pyo.Objective(rule=obj_rule)

def con_rule(model, m):
    return sum(mydata.a[m,i]*model.x[i] for i in mydata.N) \
               >= mydata.b[m]
model.con = pyo.Constraint(mydata.M, rule=con_rule)
```

Этот скрипт требует, чтобы при конструировании каждого компонента модели были доступны соответствующие данные. В нашем случае необходимые данные находятся в файле `mydata.py`:

```
N = [1,2]
M = [1,2]
c = {1:1, 2:2}
a = {(1,1):3, (1,2):4, (2,1):2, (2,2):5}
b = {1:1, 2:2}
```

Эту ЛП можно также рассматривать как абстрактную математическую модель, в которой неспецифицированные символические значения параметров определяются позже, на этапе инициализации модели. Например, эту ЛП можно следующим образом выразить в виде абстрактной модели на Pyomo:

```
import pyomo.environ as pyo

model = pyo.AbstractModel()

model.N = pyo.Set()
model.M = pyo.Set()
```

```
model.c = pyo.Param(model.N)
model.a = pyo.Param(model.M, model.N)
model.b = pyo.Param(model.M)
model.x = pyo.Var(model.N, within=pyo.NonNegativeReals)

def obj_rule(model):
    return sum(model.c[i]*model.x[i] for i in model.N)
model.obj = pyo.Objective(rule=obj_rule)

def con_rule(model, m):
    return sum(model.a[m,i]*model.x[i] for i in model.N) \
               >= model.b[m]
model.con = pyo.Constraint(model.M, rule=con_rule)
```

Этот пример включает компоненты модели, которые предоставляют абстрактные, или символические, определения множества и значений параметров. Объект `AbstractModel` откладывает инициализацию компонентов модели до момента создания *экземпляра модели*, когда передаются заданные пользователем данные множества и параметры. И конкретные, и абстрактные модели можно инициализировать данными из различных источников, включая файлы команд данных, например:

```
param : N : c :=
1 1
2 2 ;
param : M : b :=
1 1
2 2 ;

param a :=
1 1 3
1 2 4
2 1 2
2 2 5 ;
```

1.2.2. Пример раскраски графа

Мы продолжим иллюстрировать возможности моделирования Pyomo на примере простой хорошо известной задачи оптимизации: минимальной раскраски графа (известной также как раскраска вершин). В задаче о раскраске графа требуется назначить цвета вершинам графа, так чтобы никакие две смежные вершины не были окрашены в один цвет. У раскраски графов есть много практических применений, включая распределение регистров в компиляторах, планирование ресурсов и сопоставление с образцами, а кроме того, она лежит в основе развлекательных головоломок типа судоку.

Обозначим $G = (V, E)$ граф с множеством вершин V и множеством ребер $E \subseteq V \times V$. Для данного G целью в задаче минимальной раскраски графа является нахождение допустимой раскраски минимальным числом цветов. Для

простоты предположим, что ребра из множества E упорядочены таким образом, что если $(v_1, v_2) \in E$, то $v_1 < v_2$. Обозначим k максимальное число цветов и определим множество возможных цветов $C = \{1, \ldots, k\}$.

Задачу минимальной раскраски графа можно представить в виде следующей целочисленной программы (ЦП):

$$\begin{aligned} & \min\ y \\ \text{при условиях } & \textstyle\sum_{c \in C} x_{v,c} = 1 && \forall v \in V \\ & x_{v_1,c} + x_{v_1,c} \leqslant 1 && \forall (v_1, v_2) \in E \\ & y \geqslant c \cdot x_{v,c} && \forall v \in V, c \in C \\ & x_{v,c} \in \{0, 1\} && \forall v \in V, c \in C \end{aligned} \tag{1.1}$$

В этой формулировке переменная $x_{v,c}$ равна 1, если вершина v окрашена цветом c, и 0 в противном случае, а y – число использованных цветов. Первое ограничение требует, чтобы каждая вершина была окрашена ровно одним цветом. Второе ограничение означает, что вершины, соединенные ребром, должны быть окрашены разными цветами. Третье ограничение определяет нижнюю границу y и гарантирует, что y не меньше числа цветов, использованных для раскраски. Четвертое, и последнее, ограничение означает, что $x_{v,c}$ могут принимать только два значения.

На рис. 1.1 приведена формулировка описанной задачи раскраски графа в Pyomo с использованием конкретной модели; пример взят из работы Gross and Yellen [27]. Эта спецификация состоит из предложений Python, определяющих объект `ConcreteModel`, и последующего определения различных атрибутов этого объекта, включая переменные, ограничения и целевую функцию оптимизации. В строках 10–24 определены данные модели. Строка 28 – стандартное предложение импорта Python, которое вносит все символы (например, классы и функции), определенные в `pyomo.environ`, в текущее пространство имен. В строке 31 создается объект модели – экземпляр класса `ConcreteModel`. В строках 34 и 35 определены переменные модели. Отметим, что `y` – скалярная переменная, а `x` – двумерный массив переменных. В остальных строках определяются ограничения и целевая функция модели. Класс `Objective` определяет единственную целевую функцию с помощью именованного параметра `expr`. Класс `ConstraintList` представляет список ограничений, которые добавляются по одному.

По сравнению со специализированными AML, модели Pyomo, очевидно, более многословны (см., например, Hart et al. [30]). Однако этот пример иллюстрирует, что синтаксис Python все же позволяет выразить математические идеи лаконично и интуитивно понятно. Оставляя в стороне классы Pyomo, в этом примере нет ничего, кроме стандартного синтаксиса и методов Python. Например, в строке 4 используются генераторы Python для обхода всех элементов множества `colors` и применения к ним функции `sum`. В Pyomo имеются некоторые служебные функции, позволяющие упростить конструирование выражений, но никаких хитроумных расширений базовой функциональности Python не требуется.

```
#
# Пример раскраски графа, написанный на основе программы из книги
#
#  Jonathan L. Gross and Jay Yellen,
#  "Graph Theory and Its Applications, 2nd Edition",
#  Chapman & Hall/CRC, Boca Raon, FL, 2006.
#

# Определить данные графа
vertices = set (['Ar', 'Bo', 'Br', 'Ch', 'Co', 'Ec',
                 'FG', 'Gu', 'Pa', 'Pe', 'Su', 'Ur', 'Ve'])

edges = set ([('FG', 'Su'), ('FG', 'Br'), ('Su', 'Gu'),
              ('Su', 'Br'), ('Gu', 'Ve'), ('Gu', 'Br'),
              ('Ve', 'Co'), ('Ve', 'Br'), ('Co', 'Ec'),
              ('Co', 'Pe'), ('Co', 'Br'), ('Ec', 'Pe'),
              ('Pe', 'Ch'), ('Pe', 'Bo'), ('Pe', 'Br'),
              ('Ch', 'Ar'), ('Ch', 'Bo'), ('Ar', 'Ur'),
              ('Ar', 'Br'), ('Ar', 'Pa'), ('Ar', 'Bo'),
              ('Ur', 'Br'), ('Bo', 'Pa'), ('Bo', 'Br'),
              ('Pa', 'Br')])

ncolors = 4
colors = range(1, ncolors+1)


# Предложение импорта Python
import pyomo.environ as pyo

# Создать объект модели Pyomo
model = pyo.ConcreteModel()

# Определить переменные модели
model.x = pyo.Var(vertices, colors, within=pyo.Binary)
model.y=pyo.Var()

# Каждая вершина окрашивается одним цветом
model.nodecoloring = pyo.ConstraintList()
for v in vertices:
    model.node_coloring.add(
                sum(model.x[v,c] for c in colors) == 1)

# Вершины, соединенные ребром, нельзя окрашивать одним цветом
model.edge_coloring = pyo.ConstraintList()
for v,w in edges:
for c in colors:
    model.edge_coloring.add(
                model.x[v,c] + model.x[w,c] <= 1)

# Задать нижнюю границу минимального числа потребных
# цветов
model.min_coloring = pyo.ConstraintList()
for v in vertices:
    for c in colors:
        model.mincoloring.add(
                model.y>=c_model.x[v,c])

# Минимизировать число потребных цветов
model.obj = pyo.Objective(expr=model.y)
```

Рис. 1.1 ❖ Конкретная модель Pyomo для задачи минимальной раскраски графа

1.2.3. Ключевые особенности Pyomo

Python

Понятный синтаксис Python позволяет Pyomo выражать математические идеи интуитивно понятно и лаконично. Кроме того, выразительные средства программирования Python можно использовать для формулирования сложных моделей и определения высокоуровневых решателей, обращающихся к высокопроизводительным библиотекам оптимизации. Python предоставляет развитые возможности написания скриптов и позволяет пользователям анализировать модели и решения Pyomo, привлекая весь потенциал сторонних библиотек (например, numpy, scipy и matplotlib). Наконец, поскольку Pyomo погружена в Python, пользователи могут изучать ее базовый синтаксис, пользуясь обширной документацией по Python.

Настраиваемые возможности

Pyomo спроектирована с учетом модели разработки по принципу «каши из топора», когда каждый разработчик делает то, что ему хочется. Ключевым элементом такого дизайна является каркас на основе плагинов, который Pyomo использует для интегрирования компонентов модели, преобразований модели, решателей и диспетчеров решателей. Каркас управляет регистрацией этих возможностей. Таким образом, пользователи могут настраивать Pyomo модульным образом, не опасаясь дестабилизировать базовую функциональность.

Командные инструменты и скрипты

Модели Pyomo можно анализировать с помощью командных утилит или скриптов на Python. Командная утилита `pyomo` предоставляет обобщенный интерфейс к большинству возможностей моделирования Pyomo и реализует обобщенный процесс оптимизации. Этот процесс можно легко реализовать Python-скриптом и настроить под конкретные потребности пользователя.

Определение конкретных и абстрактных моделей

Примеры в разделе 1.2.1 иллюстрируют поддержку конкретных и абстрактных моделей Pyomo. Разница между этими подходами к моделированию заключается в том, когда именно инициализируются компоненты модели: в конкретных моделях компоненты инициализируются сразу, а в абстрактных инициализация откладывается до момента инициализации модели. Следовательно, оба подхода эквивалентны, а выбор зависит от контекста, в котором используется Pyomo, и от вкусов пользователя. Модели обоих типов легко инициализировать данными из самых разных источников (например, файлов в форматах csv, json, yaml, excel и баз данных).