Короткий вступ до AMPL - сучасної алгебраїчної мови моделювання

(препринт)

О.О. Щербина

2012 р.

1. Загальні відомості

1.1. Автоматизація побудови математичних моделей оптимізації.

В теперішній час методологія математичного моделювання інтенсивно розвивається, охоплюючи все нові сфери - від розробки великих технічних систем та управління ними до аналізу складних економічних і соціальних процесів, піднімаючи загальний рівень теоретичних досліджень, даючи можливість проводити їх у більш тісному зв'язку з експериментальними дослідженнями.

Математичне моделювання може розглядатися як новий загальний спосіб дослідження об'єктів реального світу - метод пізнання, конструювання, проектування, який поєднує в собі достоїнства як теорії, так і експерименту [1]. Робота не з самим об'єктом (явищем, процесом), а з його моделлю дає можливість без шкоди для модельованого об'єкта, відносно швидко і без істотних витрат досліджувати його властивості і поведінку в цікавих для дослідника ситуаціях. У той же час обчислювальні (комп'ютерні, імітаційні) експерименти з моделями об'єктів дозволяють, спираючись на потужність сучасних обчислювальних методів і технічних інструментів інформатики, вивчати об'єкти в достатній повноті, недоступної чисто теоретичним підходам.

Математичні моделі застосовуються в системах підтримки прийняття рішень, що дозволяють непрофесійному користувачеві комп'ютера використовувати сучасні математичні моделі і методи при пошуку оптимального рішення[1].

Основу математичного моделювання становить тріада модель - алгоритм - програма [1]. Розглянемо детальніше вказані компоненти. На **першому етапі** обчислювального експерименту вибирається (або будується)

[1] Сараев А. Д., Щербина О. А. Системный анализ и современные информационные технологии // Труды Крымской академии наук. - Симферополь: СОНАТ, 2006. - С. 47-59.

модель досліджуваного об'єкта, що відображає в математичній формі найважливіші його властивості закони, яким він підпорядковується, зв'язки між його складовими елементами, і т. д. Під моделлю при цьому розуміється «"еквівалент" об'єкта, що відображає в математичній формі найважливіші його властивості - закони, яким він підпорядковується, зв'язки, властиві складовим його частинам, тощо» ([1], с.8). Математична модель (її основні фрагменти) досліджується традиційними аналітичними засобами обчислювальної математики для отримання попередніх (зазвичай якісних) знань про об'єкт.

Другий етап пов'язаний з вибором (або розробкою) обчислювального алгоритму для реалізації моделі на комп'ютері. Необхідно отримати шукані величини з заданою точністю на наявній обчислювальній техніці. Обчислювальні алгоритми повинні не викривляти основні властивості моделі і, отже, вихідного об'єкта, вони повинні бути адаптовані до особливостей розв'язуваних задач і використовуваних обчислювальних засобів.

На **третьому етапі** створюється програмне забезпечення (вирішувач) для реалізації моделі і алгоритму на комп'ютері. Програмний продукт повинен враховувати специфіку математичного моделювання, пов'язану з використанням системи математичних моделей, багатоваріантністю розрахунків.

Таким чином, розробка програм, що переводять модель і алгоритм на зрозумілу комп'ютеру мову, завершує створення робочого інструмента дослідника.

Готова тріада тестується в «пробних» експериментах. На цьому етапі за допомогою ланцюжка ускладнень (ієрархії все більш повних моделей) забезпечується її адекватність. Після цього можна переходити до «дослідів», що дають всі необхідні якісні і кількісні властивості і характеристики об'єкта.

Сама по собі побудова моделей є «застосування фундаментальних законів природи, варіаційних принципів, аналогій, ієрархічних ланцюжків» ([1], с.11), а процес побудови моделей включає в себе наступні етапи ([1], с.25-26):

1. «Словесно-смисловий опис об'єкта або явища» («формулювання передмоделі»);

2. «Завершення ідеалізації об'єкту» і спрощення опису;

3. Перехід «до вибору або формулювання закону (варіаційного принципу, аналогії і т.п.)» та його запису в математичній формі;

4. Завершує формулювання моделі її «оснащення» (завдання початкового стану і параметрів об'єкта). Цей етап особливо важливий, оскільки: «...нарешті, формулюється мета дослідження моделі ...»;

5. Модель вивчається всіма доступними методами (у тому числі із застосуванням різних підходів і обчислювальних методів);

6. В результаті дослідження математичної моделі досягається поставлена мета. При цьому «повинна бути встановлена всіма можливими способами (порівнянням з практикою, зіставленням з іншими підходами) її адекватність - відповідність об'єкту і сформульованим припущенням ...».

Зауважимо, що описані вище етапи 1-2 моделювання часто називаються в літературі побудовою концептуальної моделі[2]. Концептуальне моделювання визначає, аналізує і описує необхідні поняття та обмеження галузі застосування за допомогою деякої (зазвичай діаграмної) мови моделювання, заснованої на невеликому числі мета-понять (утворюючих мета-модель). Мета концептуального моделювання - зробити об'єкти реального світу доступними на абстрактному рівні, а також допомогти досліднику у виключенні структур, що не відносяться до справи, шляхом побудови відносин між ідеалізованими елементами, зосереджуючись лише на суттєвих

[2] Рыжаков А.Н., Щербина О.А. Современные проблемы математического моделирования в исследовании операций // Динамические системы. - 2006. - Вып. 21. - С. 115-129.

зв'язках. Концептуальна модель визначає структуру модельованої системи, властивості її елементів, зв'язки між елементами, істотні для модельованої системи (з точки зору мети системи або мети моделювання). Як правило, розробка концептуальної моделі проводиться перед розробкою більш деталізованої математичної моделі.

Спираючись на тріаду модель - алгоритм - програма, дослідник отримує в руки універсальний, гнучкий і відносно недорогий інструмент, який спочатку відлагоджується, тестується і калібрується на вирішенні змістовного набору пробних задач. Після цього проводиться широкомасштабне дослідження математичної моделі для отримання необхідних якісних і кількісних властивостей і характеристик досліджуваного об'єкта за допомогою обчислювального експерименту, який за своєю природою має міждисциплінарний характер. У спільних дослідженнях беруть участь фахівці у прикладній області, з прикладної та обчислювальної математики, з прикладного і системного програмного забезпечення. Обчислювальний експеримент проводиться з використанням найрізноманітніших методів і підходів.

Слід зазначити еволюційний процес «зміни» парадигм моделювання. Перший етап моделювання характеризувався математичним записом окремих моделей реальних об'єктів. Для них характерна простота описів, типова лінійність рівнянь і мала розмірність (часто відтворюється лише одна або дві змінних). Методи аналізу пов'язані в основному з отриманням аналітичних рішень.

Другий етап характеризується спробами можливо більш повного опису об'єктів реального світу, причому об'єкт представлений в моделі у вигляді великої або складної системи, причому враховується нелінійність залежностей, зростає розмірність моделей, досягаючи кількох десятків. При

цьому для запису моделей в комп'ютері та рішення відповідних задач обчислювальної математики використовуються мови програмування. Надзвичайно важливу роль на цьому етапі набуває обчислювальний експеримент. В даний час починається перехід до третього покоління математичних моделей, в яких використовуються методи автоматизації формування моделей і даних, виникають мови моделювання, що суттєво відрізняються своєю декларативною природою від звичайних мов програмування.

У зв'язку з вищевикладеним надзвичайно актуальні питання автоматизації різних аспектів математичного моделювання: вибору і побудови моделей, записи моделей у формі, близькій до звичайного математичного запису, створення бібліотек вирішувачів[3].

[3] Вирішувач - програмне забезпечення, призначене для рішення даної математичної задачі.

1.2. Про методологічні питання моделювання

Протягом останніх десятиліть інтенсивно розвивалися різні парадигми[4] моделювання. При цьому кожна парадигма моделювання включає систему накопичених знань, досвіду, методології і засобів моделювання, призначених для вирішення відповідних цій парадигмі завдань. Парадигми моделювання розрізняються за типом моделей (наприклад, статичні, динамічні, безперервні, дискретні, детерміновані, стохастичні, нечіткі, моделі з «м'якими обмеженнями»), за різними методами аналізу моделей (наприклад, імітація, оптимізація, багатокритеріальна оптимізація, нечітка оптимізація). Крім цього, в рамках тих чи інших парадигм моделювання для вирішення відповідних завдань розробляються спеціалізовані програмні продукти - вирішувачі.

Зупинимося на методологічних питаннях моделювання. У ряді випадків не зовсім зрозуміло, про які саме моделі та задачі йдеться. Так, в російськомовній літературі часто пишуть «В результаті формалізації даних поставленої задачі отримаємо модель, яка являє собою задачу лінійного програмування. Вирішимо отриману задачу за допомогою симплекс-методу». У цьому реченні двічі зустрічалося слово «задача», яке має зовсім різний зміст. Спробуємо описати можливі розходження в ступені абстракції моделей.

Джеффріон (Geoffrion)[5] виділяє 4 рівні абстракції для моделей. На самому нижньому рівні знаходиться «конкретна модель» (model instance), яка є конкретизацією моделі, що виходить шляхом присвоєння параметрам моделі їх числових значень (або, згідно [1], шляхом «оснащення» моделі

---

[4] **Парадигма** (від грец. paradeigma приклад, зразок) - загальні принципи, концептуальна схема постановки проблем і їх вирішення, методів дослідження, загальноприйнятих в науковому товаристві. При цьому наукова парадигма складається з теорій, законів, правил, моделей, концепцій і визначень, загальноприйнятих в науковому товаристві.

[5] Geoffrion A.M. Integrated modeling system // Comp.Sci.Econ.Manag. – 1989. – V. 2. - P. 3-15.

даними). Наступний, більш високий рівень абстракції клас моделей, який є множиною подібних моделей з відомим математичним формулюванням без конкретних числових даних. Третім рівнем абстракції є парадигма моделювання, яка є множиною класів подібних моделей. І, нарешті, на четвертому рівні абстракції перебуває напрям моделювання (або підхід моделювання), який є групою схожих парадигм моделювання і може бути розділом прикладної математики. Наприклад, дослідження операцій напрямок моделювання, в який входять такі парадигми моделювання, як математичне програмування, імітаційне моделювання, теорія масового обслуговування, теорія марковських процесів.

Графовий підхід до управління моделями, наприклад, використовує методи теорії графів для представлення математичної моделі. Далі використовується *концептуальна основа (схема) моделювання* - спеціалізація в підході моделювання. Чисто *концептуальне представлення* проблеми будується всередині *концептуальної основи*.

Мова моделювання - формальна комп'ютерна здійснима нотація (система позначень), яка може бути використана для вираження абстрактних понять концептуальної основи.

Система моделювання - це комп'ютерна система, яка приймає модель користувача, транслює її в формат, що сприймається відповідним вирішувачем, викликає, переводить результат роботи вирішувача в форму, зрозумілу користувачеві. Вирішувачі звичайно не є частиною системи моделювання, система з ними лише взаємодіє, причому вирішувачі зазвичай розробляються незалежно.

Типовий процес моделювання включає побудову моделі, вибір методу вирішення, переклад прикладної моделі в модель пошуку рішення, вибір

вирішувача, переклад моделі пошуку рішення в формат вирішувача, виклик вирішувача, інтерпретація рішення.

Система моделювання взаємодіє з користувачем і вирішувачем. Хороша система моделювання повинна здійснювати цю взаємодію досить легко, тобто інтерфейс користувача повинен бути орієнтований на потреби користувача, а інтерфейс вирішувача повинен бути орієнтований на потреби вирішувачів.

1.3. Підходи до представлення і створення моделей

Підхід баз даних. У цьому підході моделі представляються за допомогою деякої моделі даних, у тому числі з використанням мережевої моделі даних CODASYL DBTG, моделі сутність-об'єкт, реляційної моделі.

Графовий підхід. Згідно з цим підходом, математична модель представляється за допомогою одного або більше графів або орграфів (див. главу 4). Використання графів при поданні знань має багато переваг, таких як концептуальна ясність, легкість програмування і простота маніпулювання. Граф складається з вершин і дуг, що відображають семантику моделі, причому вершини інтерпретуються як об'єкти. Подання складних математичних відносин за допомогою графів приводить до більш ефективній взаємодії між фахівцями з моделювання та замовниками.

Сучасною концептуальною основою моделювання, що використовує графи як частину представлення моделей, є *структурне моделювання*[6], запропоноване Джеффріоном. Представлення проблеми у вигляді структурної моделі є ациклічним графом, що описує всі компоненти проблеми і співвідношення між ними.

Підхід, заснований на знаннях. Підхід використовує засоби теорії штучного інтелекту для управління моделями. За допомогою різних схем

[6] Рыжаков А.Н., Щербина О.А. Современные проблемы математического моделирования в исследовании операций // Динамические системы. - 2006. - Вып. 21. - С. 115-129.

знань, можуть бути представлені синтаксичне знання структури проблеми, семантичне знання про різні компоненти системи, а також (процедурне) знання про те, як маніпулювати моделями. Причому у ролі схем подання знань використовуються такі концептуальні основи, як

• семантичні мережі,

• обчислення предикатів першого порядку,

• продукційні правила.

Маніпулювання моделями

Маніпулювання моделями включає можливості вибору моделі, встановлення послідовності (графіка) виклику моделей, інтеграції моделей, виклику моделей, генерації компонент нової моделі.

• Вибір моделі. Ця функція намагається знайти ті моделі, які можуть використані при формуванні адекватної моделі і потім автоматично вибирає або дозволяє користувачеві вибрати відповідну модель. Для користувача важливо те, який тип моделі є відповідним, але не те, як працює процедура пошуку рішення. Зазвичай для вибору типу моделі необхідні різні види інформації, при цьому неможливо використовувати всю цю інформацію одночасно. Тому доцільно використовувати покрокове покращення процедури вибору моделі, здійснюючи систематичне застосування цієї інформації.

- **Встановлення послідовності (графіка) виклику моделей**. Графік моделей є послідовністю моделей, заданої в порядку їх виконання, заснованої в основному на зв'язках вхід-вихід. Таким чином, якщо виходи моделі А є входами моделі В, то модель А виконується до моделі В і результат моделі А передається моделі В.

- **Інтеграція моделей.** Ця функція полягає в ідентифікації і правильному поєднанні моделей та інших необхідних для моделювання

компонент (таких як файли даних, процедури аналізу даних), необхідних для підготовки відповіді за конкретним запитом. Інтелектуальні системи, розроблені з цією метою, засновані на логічному програмуванні (див. главу 10), графах зв'язків, евристичному пошуку, правилах, семантичних мережах, фреймах.

- **Виклик моделі**. Ця функція включає активацію вибраних або інтегрованих моделей для реалізації конкретного процесу моделювання проблеми користувача.
- **Генерація нових компонент моделі.** Можлива ситуація, що жодна з існуючих моделей не здатна вирішити проблему. Це може виникнути в двох випадках: 1) Основні компоненти моделі вже є в базі моделей, але необхідні функції або тести не побудовані, 2) немає ні основних компонент моделі, ні функцій і тестів.

Інтерпретація результатів. Ця фаза полягає в перевірці припущень моделі, проведенні аналізу на чутливість і, якщо потрібно, коригуванні моделі,. Існуючі в даний час вирішувачі мають можливості проведення аналізу на чутливість у вигляді, який важко зрозуміти неспеціалістам. Дружній інтерфейс повинен показувати результати аналізу на чутливість у вигляді графів або інших простих уявлень. Прикладом реалізації системи за допомогою користувачам при роботі з моделями ЛП є **ANALYZE**[7]. Ця система не тільки допомагає допомогою користувачам проводити аналіз на чутливість, але також і в документуванні моделі, верифікації, відладці та інтерпретації результатів.

1.4. Основні риси мов моделювання

Розробка мов моделювання почалася в кінці 70-х років, коли в Світовому банку був створений GAMS (General Algebraic Modeling System).

[7] Greenberg H. Intelligent analysis support for linear programs // Computers and Chemical Engineering. – 1992. - 16(7). - P.659-674.

У мові моделювання модель записується у формі, близькій до математичного запису, що є важливою рисою алгебраїчних мов моделювання.

Запис моделі став незалежним від формату даних вирішувача. До мови моделювання можуть бути підключені різні вирішувачі, при цьому процес перекладу моделі і даних до відповідного вхідного формату вирішувача виконується комп'ютером автоматично, що дозволяє зробити помилки перекладу неможливими (зрозуміло, після ретельного налагодження). У мові моделювання моделі і дані зберігаються роздільно, тобто є чіткий поділ моделі і даних для неї. Це дозволяє багаторазово використовувати одну й ту ж модель для вирішення різних завдань, змінюючи тільки дані.

Багато систем використовують інтерфейс ODBC (open database connectivity) для автоматичного доступу до даних, а також інтерфейс до більшості табличних процесорів (таких, як Excel). Це звільнює користувача від трудомісткого обов'язку пошуку відповідних даних при кожному використанні моделі. Другою перевагою концепції поділу моделі і даних є можливість використання в період розробки моделі «іграшкових» моделей з невеликими штучними масивами даних, а в подальшому налагоджена на цих простих прикладах модель без будь-яких змін може бути використана для вирішення великих виробничих завдань з реальними обсягами інформації . Обчислення похідних даних може бути автоматизовано за допомогою використання автоматичного диференціювання.

Після того, як модель запрограмована, далі використовується вирішувач, здатний вирішувати описані вище завдання оптимізації. Алгебраїчні мови моделювання (АММ) беруть на себе завдання перекладу математичної постановки завдання в форму, зрозумілу для відповідного вирішувача (тобто програми, що вирішує оптимізаційну задачу). Після виконання завдання вирішувачем АММ видають звіт у зручній для користувача формі.

АММ дозволяють працювати з моделями, заданими у вхідному файлі у вигляді алгебраїчних рівнянь і нерівностей. АММ транслює вхідний файл в зрозумілому для вирішувача вигляді і здійснює виклик цього вирішувача. В даному випадку вирішувач розглядається як «чорний ящик» (відомо, які вихідні дані потрібні для нього і що буде отримано в результаті його роботи). У ряді випадків вирішувач може запросити додаткову інформацію, необхідну для його роботи. Як тільки вирішувач вирішить задачу, рішення буде повернуто АММ і результати будуть повідомлені користувачеві.

АММ дозволяють фахівцеві в області математичного моделювання записати модель в математичній формі, що використовує множини індексів, параметри, змінні і константи. Дуже важлива можливість АММ записувати подібні обмеження у вигляді множин, до яких потім можна звертатися, використовуючи відповідні індекси. Це дозволяє компактно записувати моделі у вигляді, близькому до природного математичного запису.

2. Алгебраїчна мова моделювання AMPL.

2.1. Основні особливості програмування на AMPL

AMPL (**A** **M**athematical **P**rogramming **L**anguage) [2] - це мова високого рівня для опису задач математичного програмування, яка використовує декларативно-алгебраїчний стиль представлення моделей математичного програмування, близький до традиційної математичного нотації. Разом з тим AMPL дає можливість описати і складні моделі оптимізації з різними логічними умовами, з використанням складних систем індексації змінних і обмежень. AMPL дозволяє задати модель математичного програмування незалежно від даних, що використовуються для конкретного прикладу моделі. В даному розділі наведені лише базові відомості про AMPL, які можуть знадобитися для опису моделей і виконання лабораторних робіт цього курсу.

AMPL вимагає також завдання вихідних даних (оснащення моделі). Модель і один (або більше) файлів даних направляються в систему AMPL. AMPL працює подібно компілятору: модель і дані з'єднуються в проміжний файл, який передається вирішувачу. Вирішувач фактично знаходить оптимальне рішення задачі, використовуючи проміжний файл, побудований AMPL, і застосовуючи відповідний алгоритм. Вирішувач видає рішення у вигляді текстового файлу.

Робота з AMPL з командного рядка

Моделі записуються у вигляді текстового файлу <назва файлу>.mod. При написанні моделі на мові AMPL можна використовувати будь-який текстовий редактор.

При написанні моделей використовуються основні команди AMPL. AMPL-модель містить опис об'єктів моделі, тобто множин, змінних, параметрів, цільової функції і обмежень. Для опису об'єктів використовуються службові слова **set, var, param, minimize / maximize, subject to** (або коротко **s.t.**). При цьому AMPL-модель містить кілька типів елементів, докладніше описуваних нижче: декларації з ключовими словами: **set** (множина індексів), **param** (параметр), **var** (змінна), **arc** (дуга - для опису мережевих моделей); з цільових функцій виду **maximize/minimize**, обмежень, **subject to** (при обмеженнях), **node** (вершина - для опису мережевих моделей).

Багато в чому синтаксис команд AMPL дуже подібний до C. AMPL підтримує такі функції, як abs (), cos (), sin (), log (), sqrt (), exp () з використанням основних операцій +, -, *, /, ^ або **. Усі команди закінчуються крапкою з комою «;». До командам виведення відносяться **display**, а також команди **write** і **print**. До інших корисним елементів відносяться **option** - для зміни опцій AMPL або вирішувача, **include** - читання з окремого файлу, а також **quit** - для виходу з AMPL.

Імена (ідентифікатори) складаються з латинських букв (великих і малих), цифр і знаків підкреслення. Символ # означає початок коментаря. Все, що знаходиться за цим символом, ігнорується AMPL. Коментарі можуть бути також обмежені символами /* і */, причому вони можуть відокремлені один від одного кількома рядками.

Команди AMPL використовують простий синтаксис:

Змінні описуються з використанням службового слова **var**.

Параметри описуються з використанням службового слова **param**.

Підсумовування $\sum_{i=1}^{n}$ записується так: sum{i in 1..n}.

Службові слова AMPL (такі як **var**, **param**, **solve**, **maximize** тощо), а також імена функцій (наприклад, **sum**, **log**, **sin**) зарезервовані і не можуть використовуватися для імен об'єктів. До службових зарезервованих слів відносяться також **for**, **if**, **elseif**, **else**, **while**, **file**, **system**.

Індекси змінних і обмежень укладаються в квадратні дужки (наприклад, a[i]).

Числа можуть записуватися у різних форматах. Так, 0.0123, 1.23D-2, 1.23e-2, 1.23E-2 - це еквівалентні записи одного і того ж числа 0,0123.

Літерали - це рядки у лапках (одинарних або подвійних), наприклад, ‘abc’, ‘x’, ‘y’, “ABC”.

Команда виду

include <ім'я файлу>

вставляє вказаний файл.

Команди

model; include < ім'я файлу >

data; include < ім'я файлу >

можуть бути скорочені до наступних:

model < ім'я файлу >;

data < ім'я файлу >;

Команда **commands** аналогічна команді **include,** але це оператор і повинен закінчуватися крапкою з комою.

В AMPL модель і дані розділені. Множини, описані в моделі, не мають розмірів або заданих елементів. Використання в моделі множини індексів означає лише те, що використовуються елементи цієї множини, причому неважливо, скільки їх є. Завдання величин робиться тільки в файлі даних, який має вигляд <ім'я файлу>.dat. Тут елементи множин і параметри визначаються явно, повторно записуючи службове слово, ім'я об'єкта і перераховуючи значення після ":=".

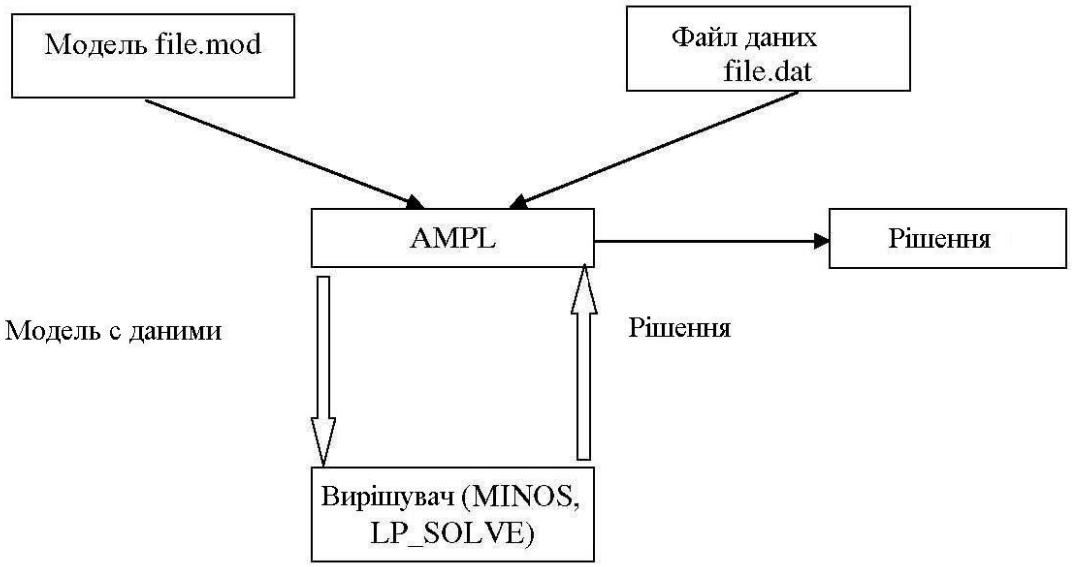

Рис.1. Схема роботи системи моделювання AMPL.

Приклад створення моделі AMPL

Вирішити задачу, використовуючи вирішувач MINOS[8]

$z = x_1 + x_2 \to \min$

при обмеженнях

$x_1 \cdot x_2 \geq 4$,

$0 \leq x_1, x_2 \leq 3$.

Створимо наступний текстовий файл з ім'ям ex1.mod

```
# ex1.mod - Model

       var x1 >= 0, <= 3;   # Завдання нижньої і верхньої границь
змінної x1
       var x2 >= 0, <= 3;   # Завдання нижньої і верхньої границь
змінної x2
      minimize z:  x1 + x2;           # Цільова функція z
      s.t. con1:  x1 * x2 >= 4; # Обмеження 1 – йому присвоєно
ім'я con1
      data;          # Завдання початкової точки для вирішувача
      var x1    := 1;
      var x2    := 1;
```

Далі наберіть `> ampl` (перебуваючи в папці, де є здійснимих файл ampl). З'явиться командне вікно AMPL з підказкою **ampl:**

У вікні AMPL наберіть наступні команди для вирішення цього завдання (не забудьте про точку з комою ";" в кінці кожної команди) для вирішення цієї задачі за допомогою вирішувача MINOS (за замовчуванням).

```
ampl: include ex1.mod;
ampl: solve;
ampl: display x1, x2, z;
```

AMPL видає наступну інформацію про рішення x1 = 2, x2 = 2 з оптимальним значенням цільової функції `z` = 4.

```
MINOS 5.5: optimal solution found.
15 iterations, objective 4
Nonlin evals: constrs = 38, Jac = 37.
x1 = 2
x2 = 2
z = 4
```

[8] MINOS – вирішувач для задач нелінійного програмування.

Для редагування текстового файлу (у зв'язку з виявленими помилками або необхідністю зміни моделі), внесіть необхідні зміни в текстовому файлі моделі і збережіть його. Потім у вікні AMPL наберіть наступні команди для введення зміненої моделі:

```
ampl:  reset;
ampl:  model ex1.mod;
```

Запишемо на AMPL модель цілочисельного лінійного програмування (ЦЛП) виду:

$$\sum_{j=1}^{n} c_j x_j \to \max$$

при обмеженнях

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = 1,\ldots, m,$$

$$x_j = 0, 1, \quad j = 1,\ldots, n.$$

Опис параметра m в AMPL виглядає так: `param m;`

Аналогічно опишемо параметр n:`param n;`

Опишемо множини індексів $i = 1,\ldots, m$ и $j = 1,\ldots, n$:

```
set I=1..m;
set J=1..n;
```

Для опису параметрів c_j, b_i, a_{ij} використовуємо запис:

```
param c{J};
param b{I};
param a{I, J};
```

Для опису змінних використовуємо запис:

```
var x {J} binary;
```

який означає, що x_j - бінарна змінна, приймаюча лише значення 0 або 1.

Опис цільової функції $\sum_{j=1}^{n} c_j x_j \to \max$ виглядає наступним чином:

```
maximize z: sum {j in 1..n} c[j] * x[j];
```

Обмеження $\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = 1,\ldots, m$ запишуться так:

```
con{i in 1..m}: sum {j in 1..n} a[i,j] * x[j] <= b[i];
```

Таким чином, модель ЦЛП записується на AMPL у вигляді ip.mod:

```
param m;
param n;
set I:=1..m;
set J:=1..n;
param c {J};
param b {I};
param a {I, J};
var x {J} binary;
maximize z:  sum{j in J}c[j]*x[j];
subject to res {i in I}:
sum{j in J}a[i,j]*x[j]<= b[i];
```

Для рішення конкретної задачі ЦЛП

$z = 2x_1 + 3x_2 + x_3 \to \max$

при обмеженнях

$x_1 + 2x_2 + x_3 \le 2,$

$2x_1 + 3x_2 + 2x_3 \le 4,$

$x_1, x_2, x_3 = 0, 1$

можна використовувати описаний вище файл моделі ip.mod разом з файлом даних ip.dat наступного вигляду

```
 param m:=2;
 param n:=3;
 param c [1] 2 [2] 3 [3] 1;
 param b [1] 2 [2] 4;
 param a: 1   2   3:=
  1       1   2   1
  2       2   3   2;
```

У вікні AMPL набираємо оператори:

```
model ip.mod;

data ip.mod;

option solver lpsolve;

solve;

display x, z;
```

В результаті отримаємо:

```
LP_SOLVE 4.0.1.0: optimal, objective 3

5 simplex iterations

3 branch & bound nodes: depth 2

x [*] :=

1  1

2  0

3  1

;



obj = 3
```

Зауваження. Запис `option solver lpsolve;` означає явне завдання вирішувача задач змішаного цілочисельного лінійного програмування ***lp_solve (****http:// lpsolve.sourceforge.net/).*

Загальні синоніми для змінних, обмежень і цільових функцій

У ряді випадків буває потрібно роздрукувати або перевірити всі наявні в моделі змінні, обмеження, цільові функції. З цією метою AMPL використовує загальні синоніми всіх компонентів:

_nvars - число змінних в даній моделі;

_ncons - число обмежень у розглянутій моделі;

_nobjs - число цільових функцій в даній моделі;

Відповідно параметри з індексами містять імена всіх компонент моделі AMPL:

_varname{1.._nvars} – імена змінних в розглянутій моделі;

_conname{1.._ncons} – імена обмежень в розглянутій моделі;

_objname {1 .. _nobjs} - імена цільових функцій в даній моделі;

І, нарешті, є такі синоніми для компонент:

_var {1 .. _nvars} - синоніми змінних в даній моделі;

_con {1 .. _ncons} - синоніми обмежень у розглянутій моделі;

_obj {1 .. _nobjs} - синоніми цільових функцій в даній моделі.

Графічний інтерфейс AMPL

Є графічний інтерфейс AMPL, званий AMPL Studio, що доступний на Web-сторінці ***http://www.optirisk-systems.com***.

2.2.2. Декларації компонентів моделі

Надалі для формального опису операторів використовуємо форму Бекуса-Наура (скор. БНФ, Бекуса-Наура форма) - формальну систему опису синтаксису, в якій одні синтаксичні категорії послідовно визначаються через інші категорії. БНФ-конструкція складається з пропозицій виду

<визначуваний символ> ::= <посл.1> | <посл.2> | . . . | <посл.n>,

що описують правила. Таке правило означає, що символ < визначуваний символ > може замінюватися на одну з послідовностей посл.1, посл.2, ..., посл.n. Знак визначення зазвичай виглядає як ::=. Крім того, використовуються квадратні дужки: [A] - елемент A входить або не входить.

Декларації об'єктів моделі AMPL мають такий загальний вигляд

<об'єкт> <ім'я об'єкта> [<аліас>] [<індексний вираз>] [<тіло>]

< об'єкт >::= set | param | var | arc | minimize | maximize | subject to | node

Декларации параметров

Параметрами можуть бути скалярні величини, вектори, матриці та масиви відомих даних. Декларація параметра має вигляд:

param <ім'я_змінної> <аліас> <індексний_вираз> <атрибути>;

<аліас> :: = <літерал>

<індексний вираз> :: = <set_expression_list

<аріфметичний_оператор> :: = + | - | less | * | / | mod | div | ^ | ** |

<унарний оператор> :: = + | -

<оператор редукції>*::= sum* | *prod* | *max* | *min*

Необов'язкові атрибути декларацій параметрів можуть розділятися комами, нижче наведено зазначені атрибути:

Атрибут:	Примітка
binary	Обмежує можливі значення змінної 0 або 1.

integer	Обмежує можливі значення змінної цілочисельними значеннями
symbolic	Якщо задано **symbolic**, то повинні також бути задані **in sexpr**, а також виключаються атрибути, що вимагають завдання числових значень, такі як > = <вираз>
>= < вираз >	>= задає нижню границю
<= < вираз >	<= задає верхню границю
:= < вираз >	:= задає початкове значення
default < вираз >	Задає значення за замовчуванням для початкових значень, які можуть бути задані в розділі даних **data**
= < вираз >	Визначає певну змінну

AMPL дозволяє визначати параметри через інші, раніше визначені, параметри. Параметри можуть також обчислюватися і визначатися рекурсивно. Наприклад, для числа сполучень C_n^m по m з n предметів може бути визначено, використовуючи наступну декларацію

```
param comb {i in 0..n, k in 0..m} :=

if k = 0 or k = i then 1 else comb[i-1,k-1] + comb[i-1,k];
```

Декларації змінних

Декларації змінних починаються зі службового слова ***var:***

var < ім'я _ змінної

> <*аліас*> <*індексний_вираз*> <*атрібути*>

де всі елементи декларації такі ж, як в декларації параметра (див. вище).

Декларації обмежень

Обмеження записуються в такій загальній формі[9]:

<декларация обмеження> :: =
[**subject to**] < ім'я обмеження > [<*аліас*>] [<*індексний*

[9] Наводиться лише найпростіша форма обмеження.

вираз>] <*атрібути*>
[: < *вираз* обмеження >];

< *вираз* обмеження >*::=* < *вираз* > <= < *вираз* > |

< *вираз* > = < *вираз* > | < *вираз* > >= < *вираз* > |

Декларація цільової функції

Декларація (опис) цільової функції має вигляд:

<декларація цільової функції> :: = **maximize** < ім'я > [<аліас>] [<*індексний вираз* >] [: *вираз*]; |
minimize < ім'я > [<аліас>] [<*індексний вираз* >] [: *вираз*];;

Огляд команд AMPL

Команда	Коментарь
call	Виклик імпортованої функції
cd	Перехід в інший каталог
check	Виконує всі команди check
close	Закриває файл
commands	Читання та інтерпретація команд з файлу
data	Перехід до даних
delete	Видалення компонентів моделі
display	Друк компонентів моделі і виразів
drop	Виключення обмеження або цільової функції
end	Закінчення введення з поточного файлу введення
environ	Встановити середовище для моделі
exit	Вийти з AMPL зі значенням статусу
expand	Показати детально компоненти моделі
fix	Зафіксувати змінну на її поточному значенні
include	Включити вміст файлу
let	Змінити значення даних
load	Завантажити динамічну бібліотеку
model	Завантажує модель
objective	Вибрати цільову функцію для оптимізації
option	Встановити або видати значення опцій
print	Неформатований друк компонентів моделі і виразів
printf	Форматований друк компонентів моделі і виразів
problem	Визначити завдання або перейти до задачі

purge	Видалення компонентів моделі
quit	Закінчити роботу AMPL
read	Читання з файлу
read table	Читання з таблиці даних
redeclare	Зміна декларації об'єкта
reload	Перезавантаження динамічної бібліотеки функцій
remove	Видалення файлу
reset	Скидання об'єктів, відновлення їх початкового стану
restore	Скасувати дію команди drop
shell	Тимчасовий вихід в операційну систему
show	Показати імена компонентів моделі
solexpand	Показати детальне розширення, видиме вирішувачеві
solution	Імпортує значення змінної з вирішувача
solve	Конкретний приклад моделі надсилається вирішувачу і повертається знайдене рішення
update	Дозволити зміну даних
unfix	Скасувати дію команди **fix**
unload	Вивантажити динамічну бібліотеку
write	Видача конкретного прикладу завдання
write table	Записати таблицю в таблицю даних
xref	Показати залежності між компонентами моделі

Файли, що згадуються в командах **include, model, data**, **commands**, які мають прості імена (наприклад, не містять слеш /), шукаються в папках, що задаються опцією **ampl_include**: кожен непорожній рядок $**ampl_include** задає таку папку, якщо ж $**ampl_include** пусто або є чистим, файли шукаються в поточній папці.

Перенаправлення виводу

Для перенаправлення виводу у файл, досить додати > і ім'я файлу. Для відкриття вже існуючого файлу і додавання до нього виводу потрібно використовувати >> замість>.

Наприклад,

ampl: model ip.mod;

ampl: data ip.dat;

ampl: solve;

```
LP_SOLVE 4.0.1.0: optimal, objective 3

5 simplex iterations

3 branch & bound nodes: depth 2
```

ampl: display x, obj > ip.out;

Скрипти та оператори потоку управління

В AMPL передбачені оператори, подібні операторам потоку керування у звичайних мовах програмування, які дають можливість писати програму за допомогою операторів, яка буде виконуватися автоматично.

При роботі з моделлю користувач часто застосовує послідовності одних і тих же команд, які потрібно набирати на екрані, Для прискорення процесу, можливо записати ці команди в спеціальний скриптова файл <file>. run, викликаючи його потім за допомогою

include <file>.run

Так, для виклику файлів з прикладу, можна записати у файлі ip.run

```
model ip.mod;

data ip.dat;

option solver lpsolve;

solve;

display x, z;
```

Для запуску скрипта на екрані AMPL набирається include ip.run;

Стандартний формат даних

AMPL підтримує стандартний формат, щоб описати значення множин і параметрів, які об'єднуються з моделлю, щоб отримати конкретну задачу оптимізації.

Розділ даних складається з послідовності токенів (лексем) (літералів і рядків друкованих символів), відокремлений пробілами, символами табуляції і переведення рядка. Лексеми включають ключові слова, літерали, числа, і роздільники () []:;: = *. Затвердження (оператор) - послідовність лексем, що закінчується крапкою з комою. Коментарі можуть використовуватися так само, як в деклараціях. У всіх випадках, розташування даних в чітких рядках і стовпцях для зручності читання людиною; AMPL ж ігнорує форматування.

Розділ даних починається з команди data і закінчується кінцем-вводу або командою, яка повертає в режим моделювання.

У розділі даних, об'єктам моделі можуть бути призначені значення в будь-якому зручному порядку, незалежному від порядку їх декларування.

Дані для множин

Оператори, що визначають множину, складаються з ключового слова set (множина), імені множини, опціональних: =, і членів. Одномірну множину найбільш просто визначити, задаючи список його членів, опціонально

відокремлених комами. Одиничні або подвійні лапки літерального рядка можуть бути опущені, якщо рядок алфавітно-цифровий, але не визначає число.

Приклади завдання даних для параметрів наведені вище, в прикладі для моделі ЦЛП.

Більш докладні відомості про мову алгебраїчного моделювання можна знайти у книзі [2] і на сайті http://www.ampl.com.

Література

1. Самарский А.А. Математическое моделирование. Идеи. Методы. Примеры / А.А. Самарский, А.П. Михайлов. - М.:ФИЗМАТЛИТ, 2005. – 320 с.

2. Fourer R. AMPL, A Modeling Language for Mathematical Programming, Second Edition / R. Fourer, D. Gay, B. Kernighan. - Belmont: Duxburry Press, 2003. – 517 p.