最適化・シミュレーション演習

第1回 AMPLの基礎

- 授業サポートページ
  http://www.shiina.mgmt.waseda.ac.jp/optsim/
- 数理計画による最適化と(離散事象型)シミュレーションに関する演習を行う. 使用するソフトウェアは, AMPL(+GurobiまたはCPLEX), および, Simul8を想定している.
- 演習では, 数理計画による最適化やシミュレーションの実践的能力を身につけることを目指す。履修者は, Cを使用できる環境を有するPCを持参すること。受講者は, 実験室にて, 演習で使用する C, AMPL, Simul8をダウンロードできる。

主な内容(最適化)

- 最適化の演習では, AMPLを用いて線形計画問題や各種の組み合わせ最適化問題を解けるようにするとともに, 列生成法, ラグランジュ緩和法, ベンダースの分解原理などのより高度な求解テクニックを学ぶ.
- **第1回 本演習の目的と概要, AMPLの基礎**
  本演習の目的, 概要, 集合, 配列, ソーティング, 乱数, その他
- **第2回 AMPLによる基本的な数理計画モデルの作成と求解**
  線形計画問題, ナップザック問題, 施設配置問題など
- **第3回 動的計画法**
  ナップザック問題, 動的ロットサイズ決定モデル(Wagner-Whitinモデル)
- **第4回 分枝限定法**
- **第5回 列生成法** カッティングストック問題
- **第6回 最短路問題** 時間制約付き最短路問題, 施設配置問題
- **第7回 ベンダース分解原理** 施設配置問題
- 第3回以降は独立したテーマで、取り扱う内容は前後する可能性があります。

主な内容(シミュレーション)

- 離散型シミュレーションの演習では, C, および, Simul8を用いて様々な待ち行列系システム, 離散事象システムのモデル化, 実験の進め方, 簡単な分線減少法と結果の分析について学ぶ. さらに, シミュレーションを用いた最適化の演習を行う.
- **第8回 シミュレーションの種類とモンテカルロシミュレーション**
  シミュレーションの種類, モンテカルロシミュレーション, 一様乱数, C による円周率πの推定
- **第9回 離散事象型シミュレーションの考え方** 事象, 事象カレンダ, 事象ルーチン, 事象制御ロジック, 単一サーバ待ち行列モデル実装の考え方
- **第10回 CによるM/M/1モデルの実装**
  指数乱数など, 実装, 結果の分析, 解析結果との比較
- **第11回 Simul8によるシミュレーション**
  M/M/1, 並列キャッシュマシンの並び方, レンタルビデオの本数
- **第12回 シミュレーション結果の分析と分散減少法**
  定常状態シミュレーションの留意点, 実験の進め方, 分散減少法(円周率πの推定, M/M/1)
- **第13回 やや複雑なSimul8のモデル**
- **第14回 シミュレーションを用いた最適化** サーバー数の最適化
- **第15回 未定 自作の最適化モデルとシミュレーションモデル**, および, それらを用いた分析

AMPL (A Modeling Language for Mathematical Programming)

- 数理計画問題用モデリング言語(Bell Laboratory)
- 各種数理計画問題をモデル化し、記述
- AMPLモデルは、数理計画ソルバー(CPLEX, lp_solve, MINOS, Xpress-MP, Gurobi など)を用いて求解
- 同時に、ソルバーの反復使用機能により、様々の解法アルゴリズムを実装できることが大きな特徴
- AMPLのホームページ http://www.ampl.com/
- 解説書「AMPL: A Modeling Language for Mathematical Programming」(by Robert Fourer (数理計画法), David M. Gay, and Brian W. Kernighan『プログラミング言語C』, Duxbury Press / Brooks/Cole Publishing Company, 2002. ISBN 0-534-38809-4)上記HPから読める
- C言語と関連

AMPL コマンドスクリプト

- AMPLのコマンドの列は *.runファイル（バッチファイル）に記述（複数のコマンドを逐次実行）
- 実行はコマンドプロンプトから「ampl *.run」と入力（*はファイル名）
- コマンドスクリプトの終わりには「exit;」を記述し、AMPLを終了
- コメントを記入したい場合は、行頭に「#」をk記入#以降が無視される
- コマンドを記述したら、行の終わりに「;」（セミコロン）を記入

表示について

- display ＊; で＊の内容が画面に表示
- display ＊ > ファイル名; とすると画面ではなくファイルに書き込まれる
- display文は改行が複数行われるため、表示を制御したい場合はprintf文を使用
- 例：「printf : "i= %d ¥n", j; 」の意味は、「j」の内容を「i= %d ¥n」という形式で表示させる。
- 「%d」は整数型、「%f」は実数型
- 記述形式はC言語に準拠
- 「¥n」は改行コード
- まとめると、「printf : "i= %d ¥n", j; 」は「i=」をテキストとして表示した後に、「j」を形式「%d」で表示して改行するというコマンド

パラメタ、変数の定義、代入文

- 一般的にはコマンドスクリプトの中で使われるパラメタ（定数、あるいは式の値など）は最初に定義しなければならない。
- パラメタの定義は次のように行う。
- param a;「aというパラメタを定義」
- var x;「xという変数を定義」
- AMPLでは変数とは、数理計画問題の定式化で用いられる変数（ソルバで最適化されるもの）を表す
- 変数や反復回数などに応じて変わる値はすべてパラメタ
- パラメタや変数に値を代入する場合、単純に「a=a+1;」と記述してはならない。
- 正しくは、let文を「let a:=a+1; 」（コロン＋等号）と記述

反復と条件

- 反復 for 文
- 「a..b」（ただし、a<b は整数値）はaからbまでの整数値の集合
- #カウンタiの値が1から10まで文1を実行
- for {i in1..10}
  {
  文1;
  }
- 注意事項：for文で使われたfor {i in 1..5}などに含まれる反復回数を示すカウンタ「i」は、for文の中でのみ使われていれば、パラメータとして定義する必要はない。

- repeat while 文
- repeat while{条件}
  {
  文1：
  }
- #条件が成立する限り文1を実行
- If-then-else 文
  if(条件 (and/or 条件)) then
  {
  文1;
  }
  else #else以下は省略可能
  {
  文2;
  }
- break;で反復より脱出

乱数と関連する関数

- 区間(m,n)における一様分布に従う乱数はUniform(m,n)
- 正規分布$N(\mu,\sigma^2)$に基づく乱数はNormal(μ,σ^2)
- 区間$[0,2^{24})$における整数値をとる一様乱数はIrand224( )
- 例：(0,1)における一様乱数を10個生成

```
option randseed 0; #乱数の初期値を更新
#0以外の正整数の場合は同じ系列の乱数
for{i in 1..10}
{
display Uniform(0,1); #Uniform(0,1)は0以上1以下の乱数
}
```

- ガウス記号[x]（x以下の最大整数） floor(x)
- x以上の最小整数を表す関数 ceil(x)

受講に際して注意する点

- 手法の理解⇒講義時間は限られている
- 演習時間が足りない⇒演習時間も限られている
- 大学設置基準⇒1コマの授業につき，1.5コマ分の予習，1.5コマ分の復習を「必要とする」
- オペレーションズ・リサーチの研究室⇒OR-Aなどの講義資料を見直すこと（他にも、所属研究室の研究に関連する講義を受講すること）