Introduction

In this project, you will use the knowledge acquired in the course to develop specialised versions of the alternating direction method of multipliers (ADMM) that can solve different types of problems.

This project consists of two problem. In the first problem, you are asked to devise an ADMM-based method to solve linear programming problems. The second problem requires you to develop a method that uses ADMM to solve a large-scale optimisation problem with a decomposable structure. You are expected to implement both methods using Julia. In addition, you must write a report where you (i) discuss how ADMM can be adapted to both of these cases and (ii) describe implementation details and performances of your ADMM implementations.

Learning objectives

1. Learn the ADMM and its variants in greater depth and understand its potential to solve optimization problems with decomposable structures.
2. Learn how to implement and apply ADMM to solve mathematical programming problems.
3. Learn how to employ the knowledge acquired in the course to understand technical literature.

General requirements

1. Students can work individually or in pairs to complete this project. Groups of three or more individuals will not be accepted.
2. Students are expected to submit a report (see details in the "Report format" section) and **all Julia code files** developed for the project. The codes will be tested and, therefore, should be commented accordingly.
3. **Deadline** for submitting the project report and the code files is **16/12/2018**. Late submissions will not be accepted.

Specific project requirements

1. You must implement the following algorithms:
   (a) An ADMM method to solve linear optimization problems.
   (b) An ADMM method to solve a large-scale linear optimisation problem with a decomposable structure. The method must be able to solve decomposed problems in parallel.

   The methods must be implemented using skeletons codes provided here for part (a) and here for part (b). The values you select for the penalty parameter ρ must be discussed in the report. All other algorithmic parameters, such as initial values for primal and dual variables, will be provided in the skeleton code.
2. The following two problems must be optimised:
   (a) A linear programming problem (P) : minimize $\left\{c^\top x : Ax = b,\ x \geq 0\right\}$.
   (b) An instance of a two-stage stochastic programming problem.

   For problem (b), we will consider a randomly generated instance of the *stochastic capacity expansion problem* which is described next.

   Stochastic capacity expansion problem

   Let $i \in I$ be a set of suppliers and $j \in J$ a set of clients with unknown demands that are represented by a finite set of scenarios $s \in S$ with corresponding demand realisations d_{js} and scenario probabilities p_s. The objective is to reserve minimum cost capacity amounts x_i from each supplier $i \in I$ in advance so that realised demands d_{js} of all customers $j \in J$ in each scenario $s \in S$ can be optimally fulfilled (on average) when they are observed.

   Let y_{ijs} be the amount of capacity reserved from supplier $i \in I$ that is used to fulfil the demand of client $j \in J$ in scenario $s \in S$. Let u_{js} be the amount of demand of client $j \in J$ in scenario $s \in S$ that is not fulfilled in case not enough capacity was reserved. We assume that unfulfilled amounts u_{js} are penalised at a unit cost q_j for each client $j \in J$ in all scenarios $s \in S$.

Each unit of capacity x_i reserved from supplier $i \in I$ costs c_i, and f_{ij} is the unit cost to fulfil the demand of client $j \in J$ using supplier $i \in I$. A maximum capacity b_i can be reserved from each supplier $i \in I$, and a total budget of B is available for capacity acquisition. This problem can be formulated as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in I} c_i x_i + \sum_{s \in S} p_s \sum_{i \in I} \sum_{j \in J} (f_{ij} y_{ijs} + q_j u_{js}) & \\
\text{subject to} \quad & \sum_{i \in I} c_i x_i \leq B & \\
& \sum_{j \in J} y_{ijs} \leq x_i, & \forall i \in I, \ \forall s \in S \\
& \sum_{i \in I} y_{ijs} = d_{js} - u_{js}, & \forall j \in J, \ \forall s \in S \\
& x_i \leq b_i, & \forall i \in I \\
& x_i \geq 0, & \forall i \in I \\
& y_{ijs} \geq 0, & \forall i \in I, \ \forall j \in J, \ \forall s \in S \\
& u_{js} \geq 0, & \forall j \in J, \ \forall s \in S
\end{aligned}$$

To accurately model the random demands d_{js} of each customer $j \in J$ in all scenarios $s \in S$, a large number of scenarios $|S|$ might be needed, which makes this class of problems challenging from a computational perspective. In this assignment, you are required to implement an ADMM-based algorithm that decomposes the problem into several subproblems, one for each scenario $s \in S$, that could be solved in parallel if needed.

Information concerning ADMM and how it can be applied to these problems can be found in this technical paper which is strongly recommended as a main reference.

Report format

A Latex template for the report is provided and can be downloaded here. Font sizes, margins, and other layout settings will be set for the template and are not to be tampered with.

The report has a strict page limit of five pages, which includes figures. References are not necessary, but if used, they are not included in the page limit.

The report will consist of the following sections:

1. **Backgroung**

   Describe the ADMM and its relevant technical details for the two applications. Provide a general pseudocode of ADMM and describe the steps based on the knowledge acquired in the course. A discussion on stopping conditions and convergence would also be of value in this section.

2. **Applications**

   Describe how ADMM can be specialised for the two applications. It is recommended that an individual subsection is used for each. Provide a detailed description of the ADMM steps in both applications, relating to the pseudocode presented in the previous section. Provide all details necessary for implementing the specialised variants of ADMM for both problems.

3. **Discussion and conclusions**

   Describe the results obtained in terms of number of steps. Assess the ADMM implementations in terms of parameter values, in particular, in terms of values used for the penalty term ρ. Present a structured performance comparison in terms of number of iterations before converging to a solution for different values of the penalty term ρ.

Assessment

The assessment is based on the report and the correctness of the ADMM implementations. The report and the Julia code will be assessed by the lecturer and the TA according to the following criteria:

(1) Correctness of the algorithms
(2) Technical quality of analysis
(3) Format, clarity, and presentation

Each criterion (1), (2), and (3) will be graded by both the lecturer and the TA, and the final grade will be the average of 5 best criterion grades.