

BPinSN

Developer guide

May 15, 2010

1 Introduction

The software package BPinSN, which stands for *Basis Pursuit in Sensor Networks*, is a tool to solve the basis pursuit (BP) problem

$$\begin{aligned} & \text{minimize} && \|x\|_1 \\ & \text{subject to} && Ax = b \end{aligned} \tag{BP}$$

in a computer cluster or in a sensor network. The mathematical details of the BP and the algorithm BPinSN applications can be found <http://users.isr.ist.utl.pt/~jmota/Software/DoubleNest.pdf>. We assume the reader is familiar with the terms defined there.

1.1 A brief overview over the BP

The BP arises when we want to find the solution of a linear system of equations of the form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{1}$$

where we have more unknowns (x_1, x_2, \dots, x_n) than equations. We can represent this linear system in a more compact form by $Ax = b$, where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \tag{2}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{3}$$

and

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} . \tag{4}$$

What the BP does is to find the solution of (1) with the least ℓ_1 -norm. The ℓ_1 -norm of the vector $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, which we denote by $\|x\|_1$, is defined as the sum of the absolute values of the entries of the vector, i.e., $\|x\|_1 = |x_1| + |x_2| + \dots + |x_n|$. This ℓ_1 -norm is usually used when we seek sparse solutions of the linear system (1), i.e., vectors with many components equal to zero or near zero. There are many applications which require sparse solutions of linear systems (see examples in <http://users.isr.ist.utl.pt/~jmota/Software/DoubleNest.pdf>).

2 Organization

The software package BPInSn consists of three main files and their respective header files:

- *Initialize_MPI.c* and *Initialize_MPI.h*
- *main_node.c* and *main_node.h*
- *fastprojector.c* and *fastprojector.h*

There are other two important files: *call_MPI.c* and *headers_constants.h*. The file *call_MPI.c* exists just to make the call to the main program in an automatic way. The reason for that is because the standard calls to an MPI program assume that the number of nodes is known beforehand, and that information is used in the command line to call the main function of the program. In our case, we decided to create a “small” program called *call_MPI* that just reads the number of nodes from the network file (*Network.txt*) and then calls *BPInSN* with the correct options in the command line.

The file *headers_constants.h* contains all the standard libraries used in the program and filenames, as well as the parameters of the algorithm.

Initialize_MPI.c This file contains the main function of the program. It starts by extracting all the information contained in the network file *Network.txt*. With this information, it runs a standard MPI code to generate a virtual topology in the computer cluster, with the same structure of the input network. After creating the virtual topology, and identifying the rank of each node, it calls the function *MN_main_node* which is implemented in the file *main_node.c*.

main_node.c The main function of this file is *MN_main_node*. The name is due to its similarity to a main function, but for each node. It starts by reading the data from the file *input.txt*. Then, it recasts the data so that BP can be solved as a linear program. After that, it calls the function *algorithm*, which implements the outer loop of Nesterov’s algorithm, as described in <http://users.isr.ist.utl.pt/~jmota/Software/DoubleNest.pdf>. To implement the algorithm described in the previous link, one needs to compute a projection of a point onto a cone of the form $\{z : Mz = y, z \geq 0\}$. That is where the function *fastprojector* is called.

fastprojector.c A detailed description of the algorithm implemented in this file will be provided soon.

3 Immediate improvements that can be done

- The installation can be simplified by making use of the environment variables MPI-COMP and BPINSN in the code. This information can be integrated in the file *headers_constants.h* and would simplify the installation process in that the user would not need to modify this file during installation. Although this has been attempted, there was always an error in the call of the function *MPI_Init*.
- The algorithm implemented in *fastprojector.c* once in a while gets stuck. It requires a debug to check if the problem is in the algorithm or in the implementation.

- The software can be easily generalized to solve any linear program in a standard form, but where the matrix of the constraints is distributed as in BPinSN. This only requires minor changes in the interface of the program.
- The program needs to output more information, such as the time it takes to solve the algorithm, the communication time, and other kind of information, such as the number of iterations in each loop.

4 Credits

The algorithm for solving the BP was developed by João F. C. Mota, João M. F. Xavier and Pedro M. Q. Aguiar from Institute of Systems and Robotics, IST, Lisboa, Portugal and João F. C. Mota and Markus Püschel from the department of Electrical and Computer Engineering at CMU, Pittsburgh, USA. The implementation of BPinSN was done by João F. C. Mota.

This work was supported by the grants SIPM PTDC/EEA-ACR/73749/2006 and SFRH/BD/33520/2008 (through the Carnegie Mellon/Portugal Program managed by ICTI) from Fundação para a Ciência e Tecnologia and also by ISR/IST plurianual funding (POSC program, FEDER). This work was also supported by NSF through award 0634967.

We would like to thank Florin Manolache, the Director of Scientific Computing for the Mellon College of Science, for providing the platforms needed for the development of this project, and also for his technical support.

5 Copyright

BPinSN - Solving the Basis Pursuit in Sensor Networks.
An implementation in C using MPI and the MKL library
Copyright © 2010 João Mota

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

6 Contact

For any questions, comments, bug reports, contact João F. C. Mota through email joaomota@cmu.edu.