

BPinSN

User 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>.

In this guide, the focus is on the interface of BPinSN, and mainly, how to use the program in the correct way. Therefore, we will try to keep the mathematical details out of our discussion as much as possible.

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

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \quad (4)$$

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 <http://users.isr.ist.utl.pt/~jmota/Software/DoubleNest.pdf>).

1.2 When to use BPinSN?

The BPinSN was designed to solve the BP in a distributed scenario, when both the matrix A and the vector b (see (2) and (3)) are distributed over a network of nodes. This can happen in real scenarios when the matrix A is too large to fit in the memory of a single computer or in an inherently distributed scenario such as a sensor network, where each node has stored a part of the matrix A and the corresponding part of the vector b . In BPinSN we assume that the matrix A is partitioned vertically

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_P \end{bmatrix}$$

and the vector b has a similar partition.

We emphasize that there are more efficient solvers¹ for the centralized case, i.e., when both the matrix A and the vector b are known in one node.

2 Installation

The steps required for the installation are described in the file `INSTALL.TXT`. Note that it is required to have installed the Intel Math Kernel Library, at least the version 11.0², and the MPICH³.

3 How to communicate with the program

The communication with the BPinSN is established through files. There are two files: one for defining the network of nodes, and the other for providing the data to each node. The former must be called *Network.txt*. The latter is a single file named *input.txt*.

¹For example the spg11: <http://www.cs.ubc.ca/labs/scl/spg11/>

² <http://software.intel.com/en-us/intel-mkl/>

³ <http://www.mcs.anl.gov/research/projects/mpich2/>

3.1 Defining the network

The BPinSN reads the specifications of the network through the file *Network.txt*. This file must be organized as follows:

```
P = 7
NE = 7
E =
1 2 2 4 4 5 5
2 3 4 5 6 6 7
degrees = 1 3 1 3 3 2 1
corresp_node2edge =
1
1 2 3
2
3 4 5
4 6 7
5 6
7
edges = 2 1 3 4 2 2 5 6 4 6 7 4 5 5
```

This description corresponds to the network of figure 1.

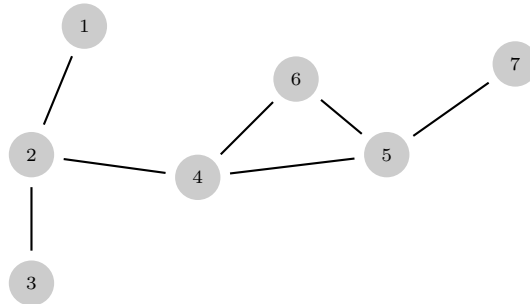


Figure 1: Network of the example.

We now describe each one of the lines of the file.

P is the number of nodes of the network

NE is the number of edges of the network

E is the representation of the network that BPinSN uses. It is a $2 \times NE$ matrix where each column contains a pair of nodes that are directly connected. **IMPORTANT NOTE:** the numeration of the nodes goes from 1 to P

degrees is a vector of length P , where each entry contains the degree of the corresponding node. For example, node 4 in figure 1 is connected to three other nodes, 2, 5 and 6, so its degree is 3.

corresp_node2edge is a list with P entries. In the file, we must have one entry per line. Each entry of the list contains the indexes of the columns of the matrix E in which node p appears. For example, node 4 appears in the 3rd, 4th and 5th columns of E . So, the 4th entry of the list should be 3 4 5.

edges is a vector constructed in the following way

$$[\text{neighbors of node 1} \quad \text{neighbors of node 2} \quad \dots \quad \text{neighbors of node P}]$$

In our example, node 1 has only one neighbor: node 2. So, the first entry of edges must be 2. Node 2, in turn, has three neighbors: nodes 1, 3 and 4. So, the next entries should be any of $[1 \ 3 \ 4]$, $[1 \ 4 \ 3]$, $[3 \ 1 \ 4]$, $[3 \ 4 \ 1]$, $[4 \ 1 \ 3]$, or $[4 \ 3 \ 1]$ (the order does not matter).

The file containing this example is provided in this package, and is named *Network_example.txt*.

3.2 Specifying the matrix A and the vector b

The matrix A , the vector b and their dimensions must be specified in a file named *input.txt*. By convention, we say that the dimensions of the matrix A are $m \times n$ and the vector b has length m . In the file *input.txt* it must be also specified the number of rows of A that go to each node of the network (every node must store the same number of rows). We designate that number by m_p . Note that $m_p = m/P$.

We will now see an example of such file (this package provides the file *input_example.txt*, which also contains an example).

```

m = 14
m_p = 2
n = 10
A =
-3 7 -1 -9 1 -12 7 3 -11 -11 -6 -21 12 7 1 -5 3 3 -2 -3
-8 -9 -4 0 16 3 21 16 -2 1 -23 -6 -7 -1 -11 1 12 -5 4 -9
5 -12 -1 -6 10 -4 -14 7 12 -20 -12 -7 0 9 5 -8 -3 -13 -3 -12
15 -1 15 5 -16 1 -10 20 -11 -5 11 -10 -8 23 0 7 -1 11 -17 -10
-5 -1 -6 6 -1 -4 10 5 6 5 -1 -2 6 5 -3 -24 -13 -7 -10 -4
-8 5 -13 -2 -7 -5 -4 19 -6 -3 4 15 -3 0 13 10 -17 0 2 2
-2 -6 5 -21 -10 4 -14 -3 6 12 9 0 5 9 19 2 -7 0 -13 -1
b =
1.064 1.278
-0.245 -0.547
-1.517 0.260
0.009 -0.013
0.071 -0.580
0.316 2.136
0.499 -0.257
c = 1 1 1 1 1 1 1 1 1 1

```

The first line contains the number of rows of A , $m = 14$. The second line specifies the number of rows that each node stores, in this case 2. Next line must contain the number of columns of A : 10 in our case. After the number of columns, it must be specified the matrix A . The rule is that there are always P lines in the file, and each line contains m_p rows. In other words, each line contains the information for a node. The vector b is specified analogously: each line of the file contains m_p (consecutive) entries of b , representing the information that each node will store. Finally, the last line of the file contains a vector c of length n full of ones⁴.

3.3 Where is the output?

If the network has P nodes, the output will consist of P files. Their names will range from *output1.txt* to *outputP.txt*, corresponding each file to the (hopefully) optimal vector x^* solving the BP. If the number of iterations of the algorithm is too small (this can be set in the file *headers_constants.h*), the output of the algorithm at each node will differ from the optimal

⁴This vector has no influence on the behavior of the program. It was put there with the hope that a generalization of this software to solve any linear program will be made.

solution and the solutions at each node will differ between themselves. An example of an output file at node 3 (the file would be *output3.txt*) is

```
x3 =  
0.000000  
0.000000  
-0.336844  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000  
0.000000
```

4 Executing the program

The program is executed with the command line

```
$. /call_MPI
```

at the directory where the program was installed.

5 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.

6 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/>.

7 Contact

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