Geosci. Model Dev., 8, 3321–3331, 2015
www.geosci-model-dev.net/8/3321/2015/
doi:10.5194/gmd-8-3321-2015

Geoscientific Model Development
Open Access

Par@Graph – a parallel toolbox for the construction and analysis of large complex climate networks

H. Ihshaish[1,3], **A. Tantet**[2], **J. C. M. Dijkzeul**[1], **and H. A. Dijkstra**[2]

[1]VORtech – Scientific Software Engineers, Delft, the Netherlands
[2]Institute for Marine and Atmospheric research Utrecht, Utrecht University, Utrecht, the Netherlands
[3]Department of Computer Science and Creative Technologies, UWE-Bristol, Bristol, UK

Correspondence to: H. Ihshaish (hisham.ihshaish@uwe.ac.uk)

Received: 23 November 2014 – Published in Geosci. Model Dev. Discuss.: 20 January 2015
Revised: 20 July 2015 – Accepted: 25 July 2015 – Published: 22 October 2015

Abstract. In this paper, we present Par@Graph, a software toolbox to reconstruct and analyze complex climate networks having a large number of nodes (up to at least 10^6) and edges (up to at least 10^{12}). The key innovation is an efficient set of parallel software tools designed to leverage the inherited hybrid parallelism in distributed-memory clusters of multi-core machines. The performance of the toolbox is illustrated through networks derived from sea surface height (SSH) data of a global high-resolution ocean model. Less than 8 min are needed on 90 Intel Xeon E5-4650 processors to reconstruct a climate network including the preprocessing and the correlation of 3×10^5 SSH time series, resulting in a weighted graph with the same number of vertices and about 3.2×10^8 edges. In less than 14 min on 30 processors, the resulted graph's degree centrality, strength, connected components, eigenvector centrality, entropy and clustering coefficient metrics were obtained. These results indicate that a complete cycle to construct and analyze a large-scale climate network is available under 22 min Par@Graph therefore facilitates the application of climate network analysis on high-resolution observations and model results, by enabling fast network construct from the calculation of statistical similarities between climate time series. It also enables network analysis at unprecedented scales on a variety of different sizes of input data sets.

1 Introduction

Over the last decade, the techniques of complex network analysis have found application in climate research. Many studies were focused on correlation patterns in the atmospheric surface temperature (Tsonis and Roebber, 2004; Tsonis et al., 2010; Donges et al., 2009b, a, 2011) and teleconnections (Tsonis et al., 2008). Up to now, the behavior of El Niño (Gozolchiani et al., 2008, 2011; Tsonis and Swanson, 2008; Yamasaki et al., 2008), the synchronization between different spatiotemporal climate variability patterns (Tsonis et al., 2007; Wyatt et al., 2011) and the connections between the sea surface temperature (SST) variability and the global mean temperature (Tantet and Dijkstra, 2014) have been investigated. In addition, network tools have also been used to detect the propagation of SST anomalies on multidecadal timescales (Feng and Dijkstra, 2014) and to develop early warning indicators of climate transitions (van der Mheen et al., 2013; Feng et al., 2014).

In most studies, the above so-called interaction networks were used. Here the observation locations serve as nodes and edges (links) are based on statistical measures of similarity, for example, a correlation coefficient, between pairwise time series of climate variables at these different locations. Given time series of climate data, represented by an $N \times M$ matrix, where N is the number of locations and M is the length of data attributes (daily or monthly values), one needs to calculate at least $N^2/2$ correlation values. Such computations become challenging for large N; for example, with a network of 10^6 nodes, this would result in 5×10^{11} calculations. A further challenge is the memory needed for such a computation. To only keep the calculated correlation matrix in memory for further processing, about 3.7×10^3 GB of memory is required (consider 8 bytes of memory for each of the 5×10^{11} matrix items), which is not available in the vast majority of current computing platforms.

On the other hand, analyzing the resulting network (graph) is non-trivial and also computationally challenging. Considering a graph G, with V vertices and E edges, a typical step in an algorithm to analyze G involves visiting each $v \in V$ and its neighbors $\bar{V} \subset V$ (the set of vertices connected to v by an edge $e \in E$), then their consecutive neighbors, and so on. Processing such steps is normally done within a computational complexity on the order of $\mid V \mid$ and/or $\mid E \mid$ squared or cubed. For example the computation of the clustering coefficient, which measures the degree to which its vertices tend to cluster together, has a time complexity of $\mathcal{O}(\mid V\mid^3)$. In practice, there are various available software tools for graph analysis, some providing implementations of single-machine algorithms such as BGL (Boost Graph Library) (Siek et al., 2002), LEDA (Mehlhorn and Näher, 1995), NetworkX (Hagberg et al., 2008), SNAP[1] and *igraph* (Csardi and Nepusz, 2006). However, the computation of a clustering coefficient for a network with $\mid V \mid = 10^6$ would be very challenging, if at all possible, with existing single-machine software.

The most popular approach to tackle such computational challenges is by exploiting parallelism for both the construction and the analysis of those massive graphs through the design of efficient algorithms for parallel computing platforms. In this regard, some contributions have been made to the development of algorithms that exploit parallel computing machines such as in The Parallel BGL (Gregor and Lumsdaine, 2005) and CGMgraph (Chan et al., 2005). However, due to structural irregularity and sparsity of real-world graphs, including those built from climate data, there are few parallel implementations that are efficient, scalable and can deliver high performance. Other factors which contribute to this inefficiency include a manifested irregularity of data dependencies in those graphs, as well as the poor locality of data, making graph exploration and analysis highly dominated by memory latency rather than processing speed (Lumsdaine et al., 2007). A recent intent with NetworKit[2] has shown a remarkable step forward towards providing parallel software tools capable of analyzing large-scale networks. Yet as in most of the existing libraries, the processing and memory challenges involved in the construction of graphs with large $|V|$ from statistical measures of time series, has not been addressed.

Indeed most researchers tend to develop their own tools to build correlation matrices beforehand, and thereafter they transform these matrices into appropriate graph data structures that can be handled by the existing libraries of graph analysis. An exception is the software package Pyunicorn[3] (Donges et al., 2013), developed at the Potsdam Institute for Climate Impact Research, that couples Python modules for numerical analysis with *igraph*. It can carry out both tasks; the construction of climate networks and the analysis of the resulted graphs. However, this software is bounded by the single-machine's memory and speed, making it impossible to reconstruct large-node climate networks and consequently, inappropriate to analyze them.

The networks which so far have been handled in climate research applications had only a limited (at most 10^4) number of nodes. As a consequence, coarse-resolution observational and model data have been used with a focus only on large-scale properties of the climate system. This system is, however, known for its multi-scale interactions and hence one would like to explore the interaction of processes over the different scales. Data are available through high-resolution ocean–atmosphere–climate model simulations but they lead to networks with at least 10^5 nodes and hence they can neither be reconstructed, nor efficiently analyzed using currently available software.

In this paper, we introduce a complete toolbox Par@Graph designed for parallel computing platforms, which is capable of the preprocessing of large number of climate time series and the calculation of pairwise statistical measures, leading to the reconstruction of large-node climate networks. In addition, Par@Graph is provided with a set of high-performance network analyzing algorithms for symmetric multiprocessing machines (SMPs). It is also coupled to a parallelized version of *igraph* (Csardi and Nepusz, 2006) – a widely used graph-analysis library. The presented toolbox is provided with an easy-to-use and flexible interface which enables it to be easily coupled to any existing graph-analysis software.

The rest of the paper is organized as follows. In Sect. 2, we give an overview of the computational challenges associated with the reconstruction of climate networks and their analysis. In Sect. 3, we provide a description of the design of Par@Graph and its parallel algorithms for the reconstruction and analysis of climate networks from climate time series. In Sect. 4, we describe the application of the toolbox to data from a high-resolution ocean model including a performance and scaling analysis. Section 5 provides a summary and discussion of the results.

2 Climate networks

A common data set of climate observations or model results consists of spatiotemporal grid points i, $i = 1, \ldots, N$ at a given latitude and longitude, each having a time series of a state variable, for example, temperature, $T_i(t_k)$ of length L, with $k = 1, \ldots, L$. In order to reconstruct a climate network, some preprocessing tasks are required beforehand, including the selection of grid locations and calculation of anomalies

[1]Stanford Network Analysis Platform see http://snap.stanford.edu.

[2]Networkit see http://networkit.iti.kit.edu.

[3]Pyunicorn see http://tocsy.pik-potsdam.de/pyunicorn.php.

(e.g., removal of a trend and/or a seasonal cycle that might produce strong autocorrelations between different locations). Having done this, each grid point is considered to be a node in the resulting network.

2.1 Network reconstruction

To define a link between two nodes, both linear and nonlinear dependencies can be considered. To measure linear correlations between the time series $T_i(t_k)$ and $T_j(t_k)$, the Pearson correlation coefficient R_{ij} given by

$$R_{ij} = \frac{\sum_{k=1}^{L} T_i(t_k)T_j(t_k)}{\sqrt{(\sum_{k=1}^{L} T_i^2(t_k))(\sum_{k=1}^{L} T_j^2(t_k))}} \tag{1}$$

is widely used (Tsonis and Roebber, 2004). Alternatively, measures of nonlinear correlation can be used, such as the mutual information M_{ij}, given by

$$M_{ij} = \sum_{T_i,T_j} P_{ij}(T_i,T_j) \log \frac{P_{ij}(T_i,T_j)}{P_i(T_i)P_j(T_j)}. \tag{2}$$

Here $P_i(T_i)$ is the probability density function (PDF) of time series T_i, and $P_{ij}(T_i,T_j)$ is the joint PDF for (T_i,T_j). The issue whether R_{ij} or M_{ij} is better to quantify the statistical similarity between nodes i and j is discussed in (Donges et al., 2009a). Whatever the choice, however, a correlation matrix ($\mathbf{C}$) of $N \times N$ elements is produced, where $\mathbf{C}_{ij} = R_{ij}$ or $\mathbf{C}_{ij} = M_{ij}$, and N is again the number of grid points.

In many climate applications, one is interested in propagating features, such as that of ocean Rossby waves. Time-delayed (time-lagged) relationships that exist between climate variables in different geographical locations have also been addressed by the climate networks approach (Gozolchiani et al., 2008; Berezin et al., 2012; Tirabassi and Masoller, 2013; Feng and Dijkstra, 2014; Tupikina et al., 2014). These are commonly measured by examining the correlation between the time series of two locations relatively shifted in time with respect to one another. Technically this can be done by defining a time-lag interval and computing the correlation measures between the shifted time series (Feng and Dijkstra, 2014). One can also define a time interval, say $[t_{\min}, t_{\max}]$, and then find the value of t in this interval where $\mathbf{C}_{ij}(t)$ is maximal (Gozolchiani et al., 2008).

Having derived the correlation matrix $\mathbf{C}$, a threshold τ is usually applied to define strong similarities between nodes as "links". The adjacency matrix $\mathbf{A}$ for the network is then found by

$$\mathbf{A}_{ij} = \mathbf{A}_{ji} = \Theta(\mathbf{C}_{ij} - \tau) - \delta_{ij}, \tag{3}$$

where Θ is the Heaviside function and δ is Kronecker delta. If correlation values are to be considered as weights for the resulted links, the elements of $\mathbf{A}$ after thresholding $\mathbf{C}$ with τ become

$$\mathbf{A}_{ij} = \begin{cases} 0 & \mathbf{C}_{ij} < \tau, \\ \mathbf{C}_{ij} & \mathbf{C}_{ij} \geq \tau. \end{cases} \tag{4}$$

Note that because $\mathbf{C}$ is symmetric, the resulting network is always undirected when $\mathbf{C}$ is calculated at zero time lag. However, when time-lagged correlations are studied, directions are added to the links between nodes, reflecting the direction of the shifting of their corresponding time series. If only thresholding is applied, but the values of the correlation matrix are kept, a weighted network will result.

2.2 Network analysis

Many properties in climate networks have interesting physical interpretations and it is important to compute them efficiently. For later reference in Sects. 3 and 4, we list here the most important properties.

– *Degree centrality.* The degree centrality k_i of a node i refers to the number of its incident vertices, that is, $k_i = |N(i)|$, where $N(i)$ is the set of vertices adjacent to i.

– *Strength centrality.* For a weighted network, the strength centrality is given by the sum of the weights of the edges between the node and its incident vertices.

– *Clustering coefficient.* The Watts–Strogatz clustering coefficient C_i measures the probability that two randomly chosen neighbors of a node are also neighbors (Watts and Strogatz, 1998). This metric is calculated for each $i \in V$ by

$$\mathbf{C}_i = \frac{2\varrho_i}{k_i(k_i-1)}, \tag{5}$$

where k_i is the number of neighbors of i and ϱ_i is the number of connected pairs between all its neighbors. When d indicates the average number of i's neighbors in a graph, this metric can be obtained in $\mathcal{O}(|V|d)$ time and in $\mathcal{O}(|V|)$ space.

– *Entropy.* The Shannon entropy (H_i) of the incident edges' weights (Anand and Bianconi, 2009) is given for node $i \in V$ by

$$H_i = -\sum_{j=1}^{k_i} p_{ij} \log\, p_{ij}\ ;\ p_{ij} = \frac{w_{ij}}{\sum_{l=1}^{k_i} w_{il}}, \tag{6}$$

where k_i is the (total) degree of node i and w_{ij} is the weight of edge(s) between nodes i and j. The computation of the entropy for all nodes in a graph is obtained in $\mathcal{O}(|V|+|E|)$ time and in $\mathcal{O}(|V|)$ space.

– *Eigenvector centrality.* This centrality metric is commonly used to evaluate the influence of a vertex in a network qualitatively. Unlike degree centrality, which weights every edge equally, the eigenvector centrality assigns relative scores to all vertices in the network based on the concept that edges with high-scoring vertices contribute more to the score of the vertex in question than equal edges with low-scoring nodes. As a result, one would find that a vertex having a high degree does not necessarily imply a high eigenvector centrality, since its connectivity might be with less important vertices. Equally, a vertex with a high eigenvector centrality is not necessarily highly linked (the vertex might have few but important links). As defined in Bonacich (1972), let $\mathbf{A} = (\mathbf{A}_{ij})$ be the adjacency matrix of the graph $G(V, E)$, the centrality score (x_i) of node $i \in V$ is then found by

$$x_i = \frac{1}{\lambda} \sum_{j \in V} \mathbf{A}_{ij} x_j, \tag{7}$$

where λ is a constant. Note that there could be many eigenvalues λ for which an eigenvector exists, however, the centrality score is determined by calculating the eigenvector corresponding to the largest positive eigenvalue of the adjacency matrix. This metric is obtained computationally in $\mathcal{O}(|V|+|E|)$ time and $\mathcal{O}(|V|)$ space.

– *Betweenness centrality.* This measure, indicated here by BC_i is based on the shortest-path enumeration. It is considered one of the more commonly used metrics to quantify the relative importance of nodes in a graph (Freeman, 1977). To obtain this metric given a graph $G(V, E)$, let σ_{st} denote the number of shortest paths between the vertices s and t. When the count of those which pass through the node i is $\sigma_{st}(i)$, then the BC_i is obtained by

$$\mathrm{BC}_i = \sum_{s \neq i \neq t \in V} \frac{\sigma_{st}(i)}{\sigma_{st}}. \tag{8}$$

With the sequential algorithm which has been proposed in Brandes (2001), it can be computed in $\mathcal{O}(|V|+|E|)$ space and $\mathcal{O}(|V||E|)$ time.

All these quantities can be obtained using the *igraph* library for relatively small-node networks.

3 Description of the toolbox

In practice, the reconstruction and analyses of climate networks are carried out through performing a set of separate tasks, progressively. First, the preprocessing of climate time series occurs, then the correlation matrix is calculated, followed by network construction from either the correlation matrix or another graph data structure like an adjacency matrix, and finally the network is analyzed using the selected graph algorithms library. Contrary to these sequence of computations, Par@Graph is designed to provide end-to-end support for the creation and analysis of climate networks by integrating parallel computing tools to perform all the involved processing efficiently, with attention at the same time to optimize required computing memory.

Par@Graph is composed of a set of coupled parallel tools designed to leverage the inherited hybrid parallelism in distributed-memory clusters of multi-core (SMPs) machines, using MPI/OpenMP standards. The provided tools are classified into two major software modules, which we refer to as the Network Constructor and the Analysis Engine, together with additional interfacing tools and wrappers.

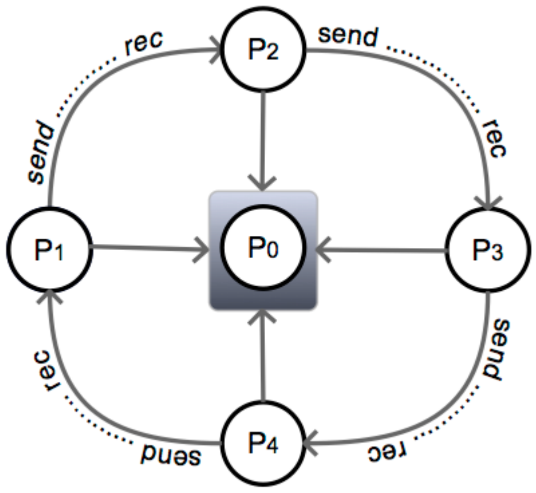


Figure 1. Provided a parallel machine of p processors, $p-1$ processes are initialized and assigned with equal blocks of time series, each block's set of time series are correlated, then these blocks are exchanged $(p-1)/2$ times (half round of the ring) between processes to complete the all-to-all correlations between the whole set of time series. Conversely, p_0 (the master computing element) is initialized as a master process to gather the resulting calculations and perform the analysis tasks on the resulted network.

3.1 Network Constructor

This module carries out the calculation of the correlation matrix $\mathbf{C}$ from the given time series. It also applies a user-defined threshold τ to generate the corresponding network adjacency matrix $\mathbf{A}$. Then it proceeds to the transformation of the resulted matrix into a network data structure which will later be analyzed by the analysis module.

The design of the constructer follows a master-worker parallel computing paradigm for distributed-memory parallel clusters of SMPs. The calculation of the correlations between time series is distributed over the computing elements (workers), forming a ring topology of processes (Fig. 1), which communicate between each other using MPI standards.

As soon as a process finds $\mathbf{C}_{ij} \geq \tau$, then the pair (i, j) is copied to a local process's buffer of a user-configurable size, and sends the iteratively filled buffer to the master p_0, where the network is to be analyzed. Note that if the network is weighted, the value of $\mathbf{C}_{ij}$ itself is also copied and sent to the master side by side with its pair of nodes i and j (and in like manner time-lag values).

A brief description of the processing associated with each ring process is described in Algorithm 1 below.

Algorithm 1 Network Constructor

```
 1: procedure RING PROCESS($p$)
 2:     $N_{local} \leftarrow p$'s block of time series
 3:     $N_{neighbor} \leftarrow$ neighbor's block of time series
 4:     neighbor($right$)$\leftarrow p+1$
 5:     neighbor($left$)$\leftarrow p-1$
 6:     preprocessing$\leftarrow$remove user specified time cycle
 7:     performance$\leftarrow$ reorder time series of $N_{local}$          ▷ for better memory-access
 8:     local block$\leftarrow$ cross correlate $N_{local}$                  ▷ performed once
 9:     for $i \leftarrow 0$ to $(p-1)/2$ do                                 ▷ iterate half ring
10:         function SEND($N_{local}$,$neighbor(right)$)                     ▷ send block to a neighbor
11:         function RECEIVE($N_{neighbor}$,$neighbor(left)$)                ▷ receive block from another
12:             function C($ij, \forall i,j \in N_{local} + N_{neighbor}$)
13:                 if $(C_{ij} >= \tau)$ then
14:                     if $weighted$ then
15:                         function SEND→MASTER(i, j, $C_{ij}$)   ▷ time-lag $t$ is also sent to master if needed
16:                     else
17:                         function SEND→MASTER(i, j)
18:             neighbor($right$)$\leftarrow$ neighbor($right$)+1
19:             neighbor($left$)$\leftarrow$ neighbor($left$)-1
20:         return $Done$
```

Note that only a subset $\bar{\mathbf{C}}$ of $\mathbf{C}$, such that $\forall \bar{\mathbf{C}}_{ij} \in \bar{\mathbf{C}}, \bar{\mathbf{C}}_{ij} \geq \tau$, is sent progressively to the master computing element. This indeed means that the under-threshold values of $\mathbf{C}$ are discarded directly at each ring process. This reduces both the amount of data sent to the master element and the memory required there for the construction of the network.

The process of constructing the network itself is performed progressively in the event that the master (p_0) receives edges' coordinates (and attributes, e.g., weights/lags) from any ring process. Initially p_0, having the number of data set grid points, constructs a completely unconnected network, that is, no edges between graph vertices. As soon as ring processes start sending edge coordinates to p_0, these edges are added to the network straightaway. In the long run, constructing the network following this approach results in saving time, since the master is idle (except when receiving data from workers) during the ring processing iterations. And more importantly, because the coming edges are added directly to the graph data structure, memory usage is optimized at the master as data redundancy are markedly minimized.

With attention to the overall performance, it is crucial not to overlook the I/O overhead, especially because the toolbox is intended to be processing large climate data sets. To that end, the Network Constructor is designed to perform multiple I/O collective operations at the same time (MPI-IO). In like manner, simultaneously, each ring process reads its chunk of time series from a parallel file system. Furthermore, owing to the fact that the elements of those time series are neither read nor stored contiguously, another key point in order to improve performance is to optimize memory access at each processor. This is provided at each process by performing preprocessing tasks that include the reordering of each process's chunk of time series, for the sake of reducing cache misses during calculation.

3.2 Analysis engine

Once correlations and their coordinates are available at the master machine, it consecutively runs graph algorithms to analyze the resulted network. The developed parallel algorithms for network analysis are based on those in *igraph*. The intent here is that this design (coupled with the Network Constructor) will achieve three primary goals: (1) to construct the network rapidly, (2) to enable efficient and safe multi-threading of the core library algorithms and (3) to reduce memory usage for network representation.

With respect to the analyzing algorithms, a set of 20 of the core algorithms of *igraph* have been parallelized using POSIX threads and OpenMP directives. Generally speaking, the embedded routines of those algorithms – a sample pseudocode is shown below in (a) – could naively be parallelized by transforming their iterative instructions into parallel loops; see (b) pseudocode:

```
while $i \leq vertices$
    do { $result(i) \leftarrow$ some processing
         $i \leftarrow i+1$
```

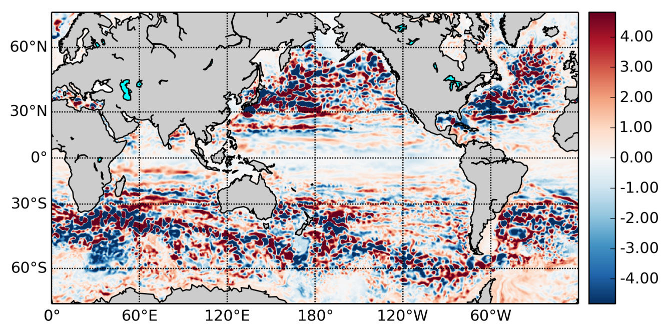
(a)

```
#pragma omp parallel for private($i$)
for $i \leftarrow 0$ to $vertices, i \leftarrow i++$
    do { $result(i) \leftarrow$ some processing
```

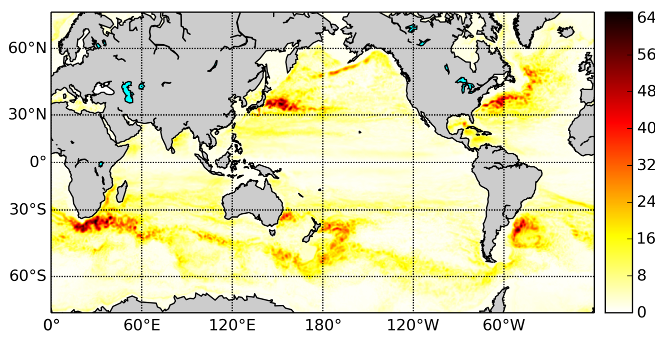
(b)

For instance, in a global transitivity routine, by which the network's average clustering coefficient is obtained, the value *result* is scalar (average value), so that parallelism appears straightforward and safe multi-threading could be achieved by applying *reduction* binary operators over its parallelizable loop. Although this may be approachable in similar cases, unfortunately in most routines *result*'s value does not depend linearly on the iteration variable i in some arbitrary way (depending on the algorithm). This is added to the synchronization overhead which could be imposed in algorithms where dependent iterative operations are found, which need careful consideration to prevent conflicts commonly caused by the concurrent access to shared memory spaces.

The parallelized algorithms of *igraph* are those mostly used to obtain important network metrics needed to evaluate structural (local and global) properties of graphs. Amongst them are the algorithms of the shortest paths, centrality measures (e.g., betweenness, closeness, eigenvector), transitivity and clustering coefficient, connected components, degree

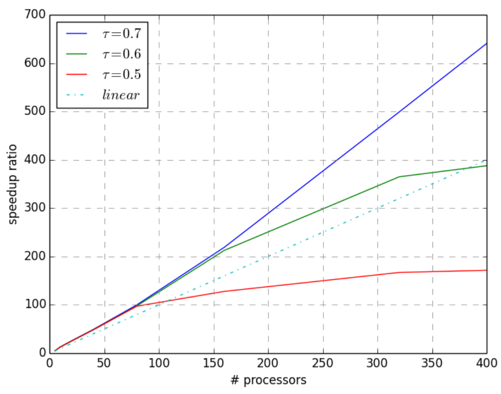


(a)

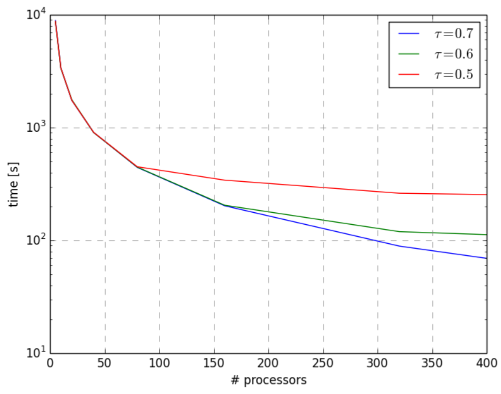


(b)

Figure 2. Mean **(a)** and standard deviation **(b)** of the daily SSH (units in cm) for year 136 of the POP control run as in (Weijer et al., 2012).



(a) Speedup



(b) Execution time corresponding to (a)

Figure 3. Speedup ratio **(a)** for the parallel construction of SSH climate networks from the POP model data having 0.4° spatial resolution. The shown speedup also includes the parallel reading and reordering of the input time series. The corresponding execution times (in seconds) over different sizes of computing processors starting from five processors upwards is given in **(b)**.

and strength centralities, entropy and diameter. A complete list of the parallelized routines and algorithms as well as the particular approach of parallelism for each would make this paper too technical and will be reported elsewhere. However, our approach to achieve efficient fine-grained parallelism for the targeted algorithms of *igraph* included major changes in their internal routines and the used data structures. For example, shared memory queues were added to achieve safe multi-threading, loops' iterations were optimized to minimize synchronization costs, and iterative workload was accordingly designed to be scheduled dynamically amongst threads in order to improve load imbalance caused by the poor locality of data. Furthermore, the internal data structure of the graph itself was modified from indexed edge lists (supported by iterators and internal stacks) to graph adjacency lists which resulted in achieving significant reduction of memory requirements, especially in the case of sparse networks.

Additionally, special attention was given to the calculation of both the degree and strength centralities. As such, both metrics' algorithms were redesigned to be computed progressively during the time the network is being constructed. In other words, each time the master receives edges from one of the ring processes, these are added to the accumulated count of the edges that corresponds to their relative vertices. As soon as the last packet of edges is received by the master, these metrics are instantly available. A notable benefit of this approach, of course apart from saving time, is the significant reduction of memory requirements, as each time the master receives a new set of edges, the previous ones are released. Such technique enables computing machines of rather few gigabytes of memory to process degree and strength centrality metrics for large-scale networks.

3.3 Interfaces and other features

In order to match a wider range of user requirements, Par@Graph is provided with all the necessary tools to do the job, including parallel collective tools to write the resulted correlation or mutual information matrices, where each ring

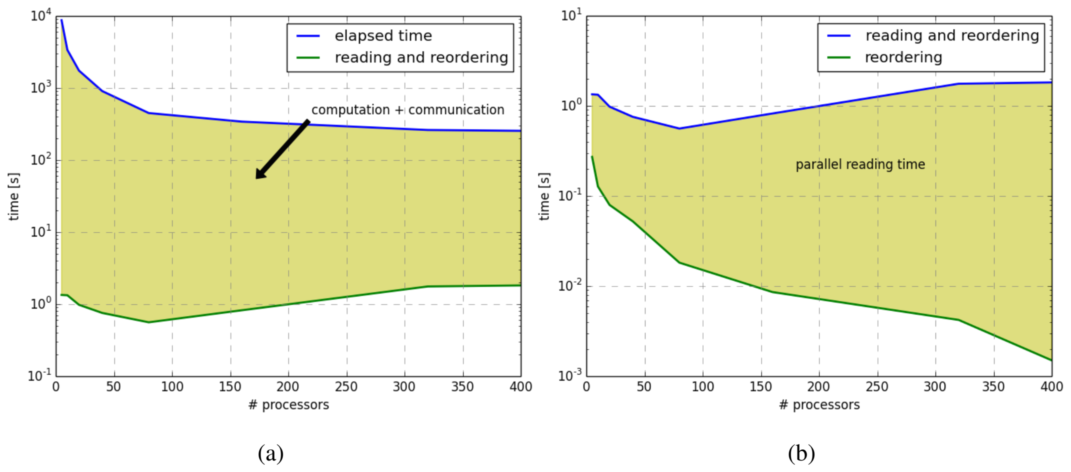


Figure 4. In **(a)** the overall runtime of the experiment in Fig. 3 (for $\tau = 0.5$) is shown in the upper curve and compared to the time for parallel reading and reordering of time series (the lower curve). The shaded area corresponds to real-cpu time for the calculation of the correlation matrix and the communication between processors. Both times for parallel reading and reordering preprocessing tasks are shown in **(b)**.

process writes its calculated portion to a common file in a parallel file system. This is added to other tools to read (in parallel) and also construct a graph directly from a matrix as well as tools to read and write standard graph formats, including edge lists, adjacency lists and the popular *Pajek* format which contains metadata added to an edge list.

Another key point is the flexible interface between the Network Constructor and the analysis engine. That is, although the toolbox provides wrappers to the parallelized *igraph*, those are quite flexible to be used with any other analysis library other than *igraph*, or any other user developed routines. Additionally, users are provided with a configuration input file where they can specify their experimental settings. These include the selection of the data grid (location coordinates), preprocessing parameters, the threshold (τ), the type of the network (weighted, unweighted, directed, etc.), time-lag intervals, whether to construct a network from time series, a matrix or another graph format.

4 Application and performance

In this section, we will apply Par@Graph to reconstruct and analyze networks obtained from high-resolution ocean model data. The motivation for performing these computations is to understand coherence of the ocean circulation at different scales (Froyland et al., 2014) .

4.1 The POP model data

The data used here are taken from simulations which were performed with the Parallel Ocean Program (POP; Dukowicz and Smith, 1994), developed at Los Alamos National Laboratory. This configuration has a nominal horizontal resolution of 0.1° and is the same as that used by Maltrud et al. (2010). We note that this configuration has a tripolar grid layout, with poles in Canada and Russia, and the model has 42 non-equidistant z-levels, increasing in thickness from 10 m just below the upper boundary to 250 m just above the lower boundary at 6000 m depth. We use data from the control simulation of this model as described in Weijer et al. (2012), where the POP is forced with a repeated annual cycle from the (normal-year) Coordinated Ocean Reference Experiment (CORE[4]) forcing data set (Large and Yeager, 2004), with the 6-hourly forcing averaged to monthly.

Correlation networks were built from 1 year (year 136 of the control run) of the simulated global daily sea surface height (SSH) data. The seasonal cycle was removed by subtracting for each day of the year its 5 days running mean averaged over years 131 to 141. The mean and standard deviation of the SSH for this year are plotted in Fig. 2a and b, respectively. Strong spatial and temporal variability can be observed in the region of the western boundary currents (e.g., the Gulf Stream in the Atlantic, the Kuroshio in the Pacific and the Agulhas Current in the Indian Ocean) and the Southern Ocean.

Two data sets have been used for network reconstruction, one with the actual 0.1° horizontal resolution of the model, resulting in 4.7×10^6 grid points, and an interpolated one with a lower 0.4° horizontal resolution resulting in 3×10^5 grid points. The latter data set has been used for the performance analysis in the next subsection.

4.2 Performance analysis

The results were computed on a bullx supercomputer [5] composed of multiple "fat" computing nodes of 4-socket bullx

[4]see http://www.clivar.org/clivar-panels/omdp/core-2.
[5]See https://surfsara.nl/systems/cartesius.

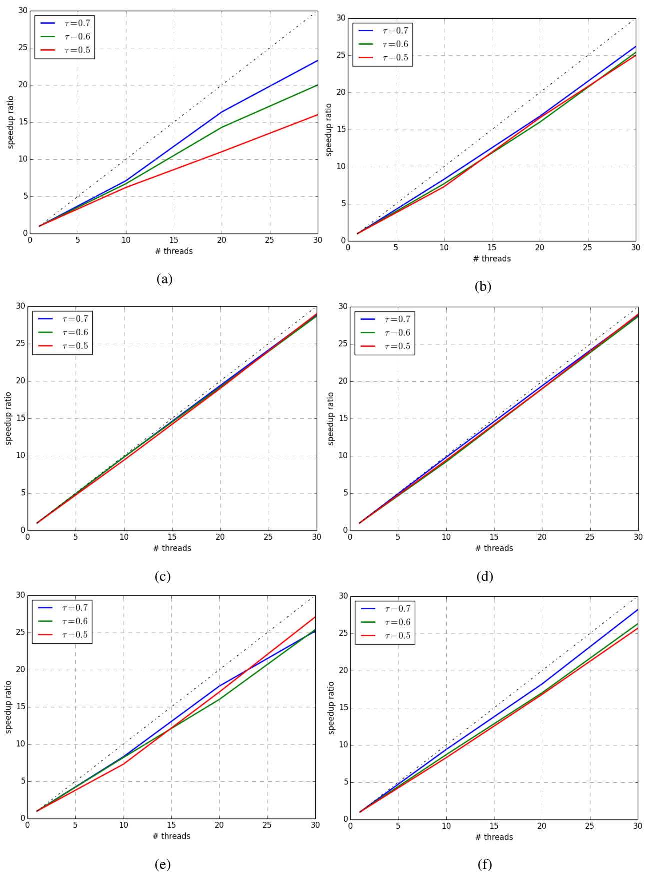


Figure 5. Performance of the parallel algorithms – **(a)** clustering coefficient, **(b)** entropy, **(c)** degree centrality, **(d)** strength centrality, **(e)** eigenvector centrality and **(f)** betweenness centrality. Algorithms are run on a single SMP bullx node of 30 compute cores. The speedup ratios correspond to the analysis of the networks 1–3 presented in Table 1.

R428 E3 each, having 8-core 2.7 GHz Intel Xeon E5-4650 (Sandy Bridge) CPUs, with a shared Intel smart cache of 20 MB at each socket, resulting in SMP nodes of 32 cores which share 256 GB of memory. The interconnection between those "fat" nodes is built on InfiniBand technology providing 56 Gbits s^{-1} of inter-node bandwidth. The same technology is used to connect the nodes to a Lustre parallel file system of 48 object storage targets (OSTs) each with multiple disks.

First experiments were performed to construct weighted Pearson correlation networks from the 0.4° POP grid, having 300842 grid points. Different edge densities (see Table 1) were obtained as a result of applying different threshold values τ for the link definition. The parallel speedup of the toolbox and the corresponding computational time are plotted in Fig. 3.

The execution time falls nearly super linearly with the number of processors up to 100. Moreover, the performance becomes strongly super linear for $\tau > 0.5$ as the number of

Table 1. Different threshold values τ used in the reconstruction of Pearson correlation networks from the 0.4 and 0.1° POP data sets and corresponding number of network vertices and edges.

Network	POP	τ	Vertices	Edges
1	0.4°	0.7	3.0×10^5	3.2×10^8
2	0.4°	0.6	3.0×10^5	1.5×10^9
3	0.4°	0.5	3.0×10^5	2.7×10^9
4	0.1°	0.4	4.7×10^6	1.4×10^{12}

processors increases. This super linearity is due to a reduction in cache misses at each processor's cache (note that 20 MB of cache are shared among each of the 8 cores) as less time series are needed to fit in those shared caches when more cores are implied. In a further analysis, we also observed that the reordering of input time series did improve the performance of the toolbox, mainly when the number of processors was less than 100. In the case of $\tau = 0.5$, performance drops with larger system sizes. First thing to remember here is that regardless of the value of the applied τ, the all-to-all correlations are calculated amongst the ring processes. However, the only difference when different values of τ are applied is the amount of data (edges, weights/attributes) that are sent to the master processor, as they will be more when lower values of τ are applied. That is to say that in such cases, communication overhead hinders the overall performance.

The timing for both the parallel reading and the reordering of time series is comparatively constant and pointless compared to the overall execution time, regardless of the number of processors, as shown in Fig. 4.

Results of the performance tests to determine the six network properties as discussed in Sect. 2.2 are shown in Fig. 5.

Although there are some differences in the performance gain in each of the algorithms, a general improvement is achieved by our fine-grained parallel implementation over the sequential *igraph* algorithms.

In some algorithms, like the clustering coefficient, parallel performance seems more sensitive to the density of the network, whereas in others, such as the degree centrality, performance remains intact. However, although an evident performance gain is observed here, one has to remember that the performance of the vast majority of network analyzing algorithms is highly dependent on the topology of the network itself, and thus a further study should be carried out to compare results for different types of networks.

In view of memory requirements, we show in Table 2 a comparison of the needed memory to represent an edge (for different types of networks) when using *igraph*'s data structures with Par@Graph's adjacency list. Indeed, the presented networks constructed by our toolbox, as a result of changing the internal data structure, are at least 60 % lighter in size

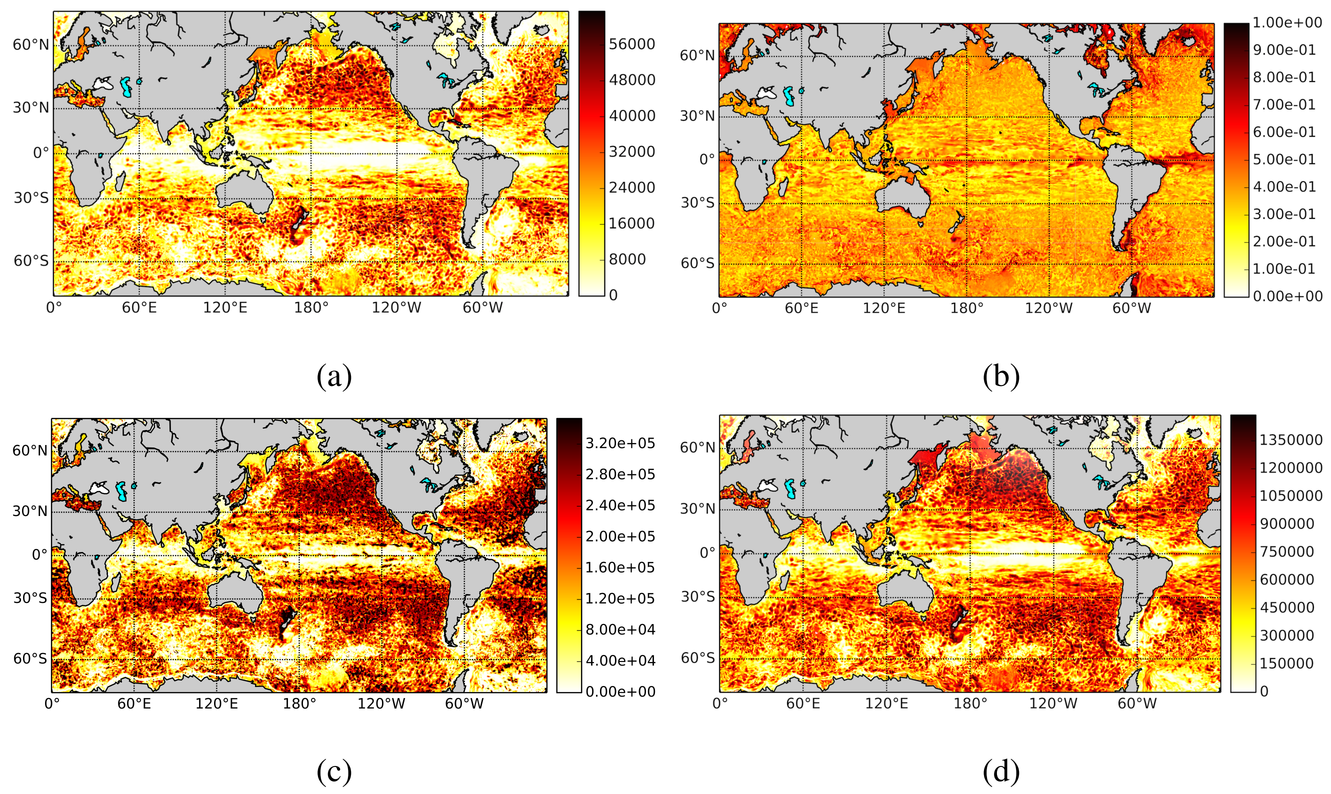


Figure 6. (a) Degree, **(b)** clustering and **(c)** betweenness for the SSH POP data interpolated on the 0.4° grid and a threshold of $\tau = 0.5$. **(d)** Degree field for the 0.1° grid and a threshold $\tau = 0.4$. Here the reconstructed network has 4.7×10^6 nodes and 1.4×10^{12} edges.

Table 2. A single edge's size in memory when using the indexed edge list used in *igraph* compared to its corresponding size when applying Par@Graph. Additionally, a vertex in *igraph* is represented by 16 bytes in memory, whereas it needs only 4 bytes in Par@Graph.

	Bytes/edge in *igraph*		Bytes/edge in *Par@Graph*	
	weighted	unweighted	weighted	unweighted
Directed	40	32	8	4
Undirected	40	32	16	8

compared to their size in memory when using the original data structures of *igraph*.

Similar performance results were obtained for tests using much larger correlation networks from the 0.1° POP grid, resulting in networks of 4.7×10^6 nodes and edges ranging from 1.5×10^{10} to 1.4×10^{12} for thresholds from 0.8 to 0.4, excluding, however, the performance for betweenness centrality and clustering coefficient algorithms for the network of 1.4×10^{12} that have not been performed. In summary, it is possible to construct large-scale climate networks in quite reasonable times on modest parallel computing platforms.

4.3 Coherence of global sea level

Being able to reconstruct and analyze the large complex networks arising from the POP ocean model, we now shortly demonstrate the novel results one can obtain. One of the important questions in physical oceanography deals with the coherence of the global ocean circulation. In low-resolution (non-eddying) ocean models, the flows appear quite coherent with near-steady currents filling the ocean basins. However, as soon as eddies are represented (when the spatial resolution is smaller than the internal Rossby radius of deformation) a fast decorrelation is seen in the flow field.

The issue of coherence has for example been tackled by looking at the eigenvalues of the transfer matrix (Dellnitz et al., 2009; Froyland et al., 2014) but also complex networks are very suited to address this question (Tantet and Dijkstra, 2014). Preliminary results on some of the important properties (degree, clustering and betweenness) of the complex network derived from the SSH data of the 0.4° POP simulation are shown in Fig. 6a–c. In all cases, a weighted, undirected network was constructed by using the Pearson correlation with zero lag and a threshold value $\tau = 0.5$.

In Fig. 6d, the degree field for the network constructed with $\tau = 0.4$ from the 0.1° POP SSH data is shown. The overall features of the degree field for the 0.1° POP data are already found in the degree field for the 0.4° POP data, but additional small-scale correlations can be distinguished.

The precise physical interpretation of these metrics is outside the scope of this paper as it requires a background in dynamical oceanography. However, one can observe that the subtropical gyres (Dellnitz et al., 2009; Froyland et al., 2014) tend to have a large degree while the regions of the western boundary currents, near the Equator and Southern Ocean tend to have smaller degree.

5 Summary and conclusions

Up to now, the data sets (both observational and model based) used to reconstruct and analyze climate networks have been relatively small due to computational limitations. In this paper we presented the new parallel software toolbox Par@Graph to construct and analyze large-scale complex networks. The software exposes parallelism on distributed-memory computing platforms to enable the construction of massive networks from a large number of time series based on the calculation of common statistical similarity measures between them. Additionally, Par@Graph is provided with a set of parallel graph algorithms to enable fast calculation of important properties of the generated networks on SMPs. These include those of the betweenness, closeness, eigenvector and degree centralities as well as the algorithms needed for the calculation of transitivity, connected components, entropy and diameter. Additionally, a parallel implementation of a community detection algorithm based on modularity optimization (Blondel et al., 2008) is provided.

The capabilities of Par@Graph were shown by using sea surface height data of a strongly eddying global ocean model (POP). The resulting networks had number of nodes ranging from 3.0×10^5 to 4.7×10^6, with the number of edges ranging from 3.2×10^8 to 1.4×10^{12}. The performance of Par@Graph showed excellent parallel speedup in the construction of massive networks, especially when higher thresholds were applied. When lower values of τ were used, communication overhead was seen to decrease the performance. On the other hand, we observed a significant speed gain in the calculation of the discussed network characteristics which was obtained by our parallel implementation of *igraph*.

With regards to the challenging issue of memory requirements in order to compute such big networks, we showed that the presented toolbox notably optimizes the usage of memory during the reconstruction of large-scale networks by minimizing the accompanying data redundancy. Additionally, the resulted networks themselves are markedly lighter in size compared to their equivalents in *igraph* as a result of changing the data structures from indexed edge lists to adjacency lists.

The availability of Par@Graph will allow one to solve a new set of questions in climate research, one of which, the coherence of the ocean circulation at different scales, was shortly discussed in this paper. Apart from higher resolution data sets of one observable, it will now also be possible to deal with data sets of several variables and to more efficiently reconstruct and analyze networks of networks (Berezin et al., 2012). However, apart from climate research, Par@Graph will also be very useful for all fields of science where very large-scale networks are applied, and it is hoped that the toolbox will find its way into the complexity science community.

Code availability

Par@Graph is not yet provided with a license. For the time being, source code will be available from authors upon request. Authors will also provide support in the initial software installation and setup.

Acknowledgements. The authors would like to acknowledge the support of the LINC project (no. 289447) funded by EC's Marie-Curie ITN program (FP7-PEOPLE-2011-ITN). The computations on the Cartesius machine were funded by the Exact Sciences division of the Netherlands Organization of Scientific Research under grant SH-284-14.

Edited by: D. Ham

References

Anand, K. and Bianconi, G.: Entropy measures for networks: Toward an information theory of complex topologies, Phys. Rev. E, 80, 045102, doi:10.1103/PhysRevE.80.045102, 2009.

Berezin, Y., Gozolchiani, A., Guez, O., and Havlin, S.: Stability of climate networks with time., Scientific Reports, 2, 666, doi:10.1038/srep00666, 2012.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E.: Fast unfolding of communities in large networks, J. Stat. Mech.-Theory E., 2008, P10008, doi:10.1088/1742-5468/2008/10/P10008, 2008.

Bonacich, P.: Factoring and weighting approaches to status scores and clique identification, J. Math. Sociol., 2, 113–120, 1972.

Brandes, U.: A Faster Algorithm for Betweenness Centrality, J. Math. Sociol., 25, 163–177, 2001.

Chan, A., Dehne, F., and Taylor, R.: CGMgraph/CGMlib: Implementing and Testing CGM Graph Algorithms on PC Clusters and Shared Memory Machines, in: International Journal of High Performance Computing Applications, vol. 19, Springer, 81–97, doi:10.1177/1094342005051196, 2005.

Csardi, G. and Nepusz, T.: The igraph software package for complex network research, InterJournal, Complex Systems, http://igraph.org (last access: 20 July 2015), p. 1695, 2006.

Dellnitz, M., Froyland, G., Horenkamp, C., Padberg-Gehle, K., and Sen Gupta, A.: Seasonal variability of the subpolar gyres in the Southern Ocean: a numerical investigation based on transfer operators, Nonlinear Proc. Geoph., 16, 655–663, 2009.

Donges, J. F., Zou, Y., Marwan, N., and Kurths, J.: Complex networks in climate dynamics, Eur. Phys. J.-Spec. Top., 174, 157–179, 2009a.

Donges, J. F., Zou, Y., Marwan, N., and Kurths, J.: The backbone of the climate network, Europhys. Lett., 87, 48007, doi:10.1209/0295-5075/87/48007, 2009b.

Donges, J. F., Schultz, H. C. H., Marwan, N., Zou, Y., and Kurths, J.: Investigating the topology of interacting networks, Eur. Phys. J.-B, 84, 635–651, 2011.

Donges, J. F., Heitzig, J., Runge, J., Schultz, H. C. H., Wiedermann, M., Zech, A., Feldhoff, J., Rheinwalt, A., Kutza, H., Radebach, A., Marwan, N., and Kurths, J.: Advanced functional network analysis in the geosciences: The pyunicorn package, in: EGU General Assembly Conference Abstracts, vol. 15, EGU General Assembly Conference Abstracts, http://adsabs.harvard.edu/abs/2013EGUGA..15.3558D (last access: 20 September 2015), 3558, 2013.

Dukowicz, J. K. and Smith, R. D.: Implicit free-surface method for the Bryan-Cox-Semtner ocean model, J. Geophys. Res., 99, 7991–8014, 1994.

Feng, Q. Y. and Dijkstra, H.: Are North Atlantic multidecadal SST anomalies westward propagating?, Geophys. Res. Lett., 41, 541–546, 2014.

Feng, Q. Y., Viebahn, J. P., and Dijkstra, H. A.: Deep ocean early warning signals of an Atlantic MOC collapse, Geophys. Res. Lett., 41, 6009–6015, 2014.

Freeman, L. C.: A Set of Measures of Centrality Based on Betweenness, Sociometry, 40, 35–41, 1977.

Froyland, G., Stuart, R. M., and van Sebille, E.: How well-connected is the surface of the global ocean?, Chaos: An Interdisciplinary Journal of Nonlinear Science, 24, 033126, doi:10.1063/1.4892530, 2014.

Gozolchiani, A., Yamasaki, K., Gazit, O., and Havlin, S.: Pattern of climate network blinking links follows El Niño events, Europhys. Lett., 83, 28005, doi:10.1209/0295-5075/83/28005, 2008.

Gozolchiani, A., Havlin, S., and Yamasaki, K.: Emergence of El Niño as an Autonomous Component in the Climate Network, Phys. Rev. Lett., 107, 1–5, 2011.

Gregor, D. and Lumsdaine, A.: The parallel bgl: A generic library for distributed graph computations, in: In Parallel Object-Oriented Scientific Computing (POOSC), Publications, O. I. U., OSL Indiana University Publications, Indiana, USA, http://www.osl.iu.edu/publications/prints/2005/Gregor:POOSC:2%005.pdf (last access: 20 September 2015), 2005.

Hagberg, A. A., Schult, D. A., and Swart, P. J.: Exploring network structure, dynamics, and function using NetworkX, in: SciPy 2008: Proceedings of the 7th Python in Science Conference, edited by: Varoquaux, G., Vaught, T., and Millman, J., Pasadena, CA USA, 11–15, 2008.

Large, W. G. and Yeager, S.: Diurnal to decadal global forcing for ocean and sea-ice models: the data sets and flux climatologies, Tech. rep., National Center for Atmospheric Research, Boulder, CO, USA, 2004.

Lumsdaine, A., Gregor, D., Hendrickson, B., and Berry, J. W.: Challenges in Parallel Graph Processing, Parallel Processing Letters, 17, 5–20, 2007.

Maltrud, M., Bryan, F., and Peacock, S.: Boundary impulse response functions in a century-long eddying global ocean simulation, Environ. Fluid Mech., 10, 275–295, 2010.

Mehlhorn, K. and Näher, S.: LEDA: A Platform for Combinatorial and Geometric Computing, Commun. ACM, 38, 96–102, 1995.

Siek, J., Lee, L.-Q., and Lumsdaine, A.: The Boost Graph Library: User Guide and Reference Manual, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

Tantet, A. and Dijkstra, H. A.: An interaction network perspective on the relation between patterns of sea surface temperature variability and global mean surface temperature, Earth Syst. Dynam., 5, 1–14, doi:10.5194/esd-5-1-2014, 2014.

Tirabassi, G. and Masoller, C.: On the effects of lag-times in networks constructed from similarities of monthly fluctuations of climate fields, Europhys. Lett., 102, 59003, doi:10.1209/0295-5075/102/59003, 2013.

Tsonis, A. A. and Swanson, K. L.: Topology and Predictability of El Niño and La Niña Networks, Phys. Rev. Lett., 100, 228502, doi:10.1103/PhysRevLett.100.228502, 2008.

Tsonis, A. A. and Roebber, P.: The architecture of the climate network, Physica A, 333, 497–504, 2004.

Tsonis, A. A., Swanson, K., and Kravtsov, S.: A new dynamical mechanism for major climate shifts, Geophys. Res. Lett., 34, L13705, doi:10.1029/2007GL030288, 2007.

Tsonis, A. A., Swanson, K. L., and Wang, G.: On the Role of Atmospheric Teleconnections in Climate, . Climate, 21, 2990–3001, 2008.

Tsonis, A. A., Wang, G., Swanson, K. L., Rodrigues, F. A., and Costa, L. D. F.: Community structure and dynamics in climate networks, Clim. Dynam., 37, 933–940, 2010.

Tupikina, L., Rehfeld, K., Molkenthin, N., Stolbova, V., Marwan, N., and Kurths, J.: Characterizing the evolution of climate networks, Nonlinear Proc. Geoph., 21, 705–711, 2014.

van der Mheen, M., Dijkstra, H. A., Gozolchiani, A., den Toom, M., Feng, Q., Kurths, J., and Hernandez Garcia, E.: Interaction network based early warning indicators for the Atlantic MOC collapse, Geophys. Res. Lett., 40, 2714–2719, 2013.

Watts, D. J. and Strogatz, S. H.: Collective dynamics of “small-world” networks, Nature, 393, 440–442, 1998.

Weijer, W., Maltrud, M. E., Hecht, M. W., Dijkstra, H. A., and Kliphuis, M. A.: Response of the Atlantic Ocean circulation to Greenland Ice Sheet melting in a strongly-eddying ocean model, Geophys. Res. Lett., 39, L09606, doi:10.1029/2012GL051611, 2012.

Wyatt, M. G., Kravtsov, S., and Tsonis, A. A.: Atlantic Multidecadal Oscillation and Northern Hemisphere’s climate variability, Clim. Dynam., 38, 929–949, 2011.

Yamasaki, K., Gozolchiani, A., and Havlin, S.: Climate Networks around the Globe are Significantly Affected by El Niño, Phys. Rev. Lett., 100, 1–4, 2008.