Physica A 392 (2013) 2226–2234

Contents lists available at SciVerse ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

Minimal spanning tree problem in stock networks analysis: An efficient algorithm

Maman Abdurachman Djauhari *, Siew Lee Gan

Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor Bahru, Johor Darul Takzim, Malaysia

ARTICLE INFO

Article history:
Received 21 July 2012
Received in revised form 26 November 2012
Available online 3 January 2013

Keywords:
Adjacency matrix
Euclidean distance
Membership function
Sub-dominant ultrametric
Ultrametric distance

ABSTRACT

Since the last decade, minimal spanning trees (MSTs) have become one of the main streams in econophysics to filter the important information contained, for example, in stock networks. The standard practice to find an MST is by using Kruskal's algorithm. However, it becomes slower and slower when the number of stocks gets larger and larger. In this paper we propose an algorithm to find an MST which has considerably promising performance. It is significantly faster than Kruskal's algorithm and far faster if there is only one unique MST in the network. Our approach is based on the combination of fuzzy relation theory and graph theoretical properties of the forest of all MSTs. A comparison study based on real data from four stock markets and four types of simulated data will be presented to illustrate the significant advantages of the proposed algorithm.

1. Introduction

There are two main streams in stock networks analysis. The first is about the mathematical law that governs the time series data of stock prices. That law has a long history before it becomes a main stream in the study of stock prices. The empirical study is dated back to 1863 when Jules Regnault formulated 'the law on the square root of time' [1]. It says that if $p_i(t)$ is the price of stock i at time t and $p_i(t)$ is a Brownian motion process, then the deviation $|p_i(t + \Delta t) - p_i(t)|$ is about $\sqrt{\Delta t}$. But, it was Bachelier in 1900 who first derived a mathematical model of the dynamic of stock prices. He assumed that the changes in stock prices are subject to fixed mathematical laws. With this assumption he then used the increments of Brownian motion to model absolute price changes. Since this model could make the prices of stocks become negative [2], instead of modeling the absolute price changes, now it is common to model the relative price changes by using geometric Brownian motion (GBM) process. This model was popularized by Paul Samuelson who received the Nobel Prize in Economics in 1970 [1]. The popularity is due to the fact that this model facilitates a convenient work environment. More specifically, let $p_i(t)$ be a GBM process and $r_i(t)$ be the logarithm of i-th stock price return at time t. Then $r_i(t)$ follows a Gaussian distribution. Thus, under this model, similarity among stocks viewed as time series could be measured by the Pearson correlation coefficient.

The second is the search for the tool to filter the important information in stock networks. Only recently has the graph theoretical approach been used to filter the information contained in stock networks governed by GBM process. It was pioneered by Mantegna in 1999 [3]. He used the minimal spanning tree (MST) to study the topological properties of stocks and the corresponding sub-dominant ultrametric (SDU) to understand the hierarchical tree of stocks. After the work of Borůvka in 1926, people had to wait for 73 years before MST became another mainstream in stock networks analysis. Nowadays, the role of MST together with SDU as information filter can be found in many areas of financial industries and also

* Corresponding author. Tel.: +60 17 5200795; fax: +60 7 55 66162.
E-mail addresses: maman@utm.my (M.A. Djauhari), slgan3@live.utm.my (S.L. Gan).

0378-4371/$ – see front matter
doi:10.1016/j.physa.2012.12.032

in many areas of scientific investigations. See, for example, in general complex system [4], foreign exchange [5], economy analysis [6], portfolio analysis [7], risk assessment in term of single linkage [8], trading [9], and volatility [10].

Unlike the use of GBM process which has been accepted as the appropriate model to describe stock price, the construction of MST and SDU to filter the important information in stock networks is still in development. A significantly faster algorithm than the known ones is always sought after. It is in this spirit that this paper is presented; the spirit to contribute in searching for a better solution to the MST problem.

1.1. The role of MST in stock networks analysis

Network among stocks is a complex system [4]. It is usually represented as correlation networks where the interrelationship between two different stocks is quantified by using Pearson coefficient of correlation (PCC) between their respective logarithm of price return. This linear relationship is justified by the use of GBM process to model the mathematical law of $p_i(t)$. Under that model, the random variable $r_i(t)$ follows a Gaussian distribution and thus the similarity between two different stocks can be represented in terms of PCC. The network is then summarized in the form of a correlation matrix [3,7,11] and analyzed through the corresponding distance (dissimilarity, in general) matrix. This is to understand the topological structure and economic classification of the stocks [12]. If the topological structure is helpful in identifying the topological properties of the stocks such as stocks' centrality, which are useful in the study of the relative position of a particular stock with respect to the others, economic classification is to identify the stocks that have similar behavior [3,11].

The topological properties are usually identified by using the network topology representing an MST in the distance networks [13] while the economic classification in the form of an indexed hierarchical tree is constructed based on the SDU of the distance networks. Due to that strategic role of MST and SDU, it is important to note here three properties of MST in conjunction with the SDU. First, once an MST has been obtained, then it can be used to find the SDU. This is the current practice in stock networks analysis. Second, although the MST in a given distance networks might not be unique, the SDU always is. Third, the SDU can actually be obtained without passing through MST. A recent result is presented by Djauhari in Ref. [14]. He shows that if one has obtained the SDU, then only the forest of all MSTs can directly be identified and not an MST except of course when there is only one single MST in the network. We will discuss further the last two properties in Section 3.

1.2. The MST problem

Recent work on the history of the MST problem shows that there were a variety of independent discoveries of the algorithms and ideas to solve that problem [15]. See also Refs. [16,17] for other important historical background on that problem. The first efficient solution, which has become popular since the last decade, is credited to the work of Borůvka in 1926. Since Borůvka's algorithm requires that all edges have distinct weights, which is very rare in a real situation, in the current practice among stock players and researchers the MST in stock networks analysis is usually obtained from the two most suggested algorithms in the literature, i.e., Kruskal's algorithm and Prim's algorithm [3,11,16]. See also Zhang et al. Ref. [6] who use these algorithms in economy analysis and also Ulusoy et al. [9] in trading.

The popularity of those two algorithms motivates Huang et al. [18] to conduct a study to compare their computational complexity. They have reported the conditions where one algorithm is superior to the other. However, many researchers agree that the prominent role is played by Kruskal's algorithm. It is mathematically very appealing [15] and easy to formulate although it is not the fastest algorithm [17]. This is perhaps the reason why many researchers prefer to use Kruskal's algorithm instead of Prim's algorithm. See, for example, Refs. [5,19,20].

In this paper an efficient algorithm which considerably improves the running time of Kruskal's algorithm will be proposed. Since Kruskal's algorithm determines an MST directly from the distance networks, its performance becomes slower and slower when the number of stocks gets larger and larger, say of the order of hundreds. The proposed algorithm does not search MST directly from the distance networks. It searches MST in two steps. The first step is to find the forest of all possible MSTs (or the 'forest' in brief) in the distance networks. The second is the search for an MST from the forest. The details are in Section 3. This method will reduce considerably the running time needed to obtain an MST and the SDU simultaneously.

In the next section, we begin our discussion by recalling the construction process of the SDU from the fuzzy relation viewpoint discussed in Ref. [14]. This will allow us to see in greater depth the properties of distance networks which will be useful to find the SDU efficiently. Based on the SDU, in Section 3 we construct the forest, then find an MST in the sub-graph of the forest obtained after a repeated process of leaves (nodes of degree one) removal by using Kruskal's algorithm, and finally construct an MST in the network. A comparison study with Kruskal's algorithm and also Prim's algorithm in terms of the running time, discussed in Section 4, will illustrate the significant advantages of the proposed algorithm and concluding remarks in the last section will close this presentation.

2. Construction of SDU from fuzzy relation viewpoint

In this section we recall and discuss briefly the standard procedure to filter the important information contained in stock networks. Let E be a set of n stocks in a portfolio under study, and again $p_i(t)$ and $r_i(t)$ be the price of stock i and the logarithm

of its price return at time t, respectively. Thus, if $\Delta p_i(t) = p_i(t) - p_i(t-1)$,

$$r_i(t) = \ln\left(1 + \frac{\Delta p_i(t)}{p_i(t-1)}\right)$$

for all $i = 1, 2, \ldots, n$. In practice, the right hand side can be approximated by $\frac{\Delta p_i(t)}{p_i(t-1)}$ if this quantity is small. Under assumption that $p_i(t)$ is a GBM process for all $i = 1, 2, \ldots, n$, $\{r_1(t), r_2(t), \ldots, r_n(t)\}$ follows an n-dimensional Gaussian distribution. Therefore, the interrelationships among stocks can be summarized in the form of a Pearson correlation matrix C of size $(n \times n)$ where its element of the i-th row and j-th column is,

$$c(i,j) = \frac{\langle r_i r_j \rangle - \langle r_i \rangle \langle r_j \rangle}{\sqrt{\langle r_i^2 \rangle - \langle r_i \rangle^2}\sqrt{\langle r_j^2 \rangle - \langle r_j \rangle^2}}$$

with $\langle r_i \rangle$ is the average of $r_i(t)$ for all t [7,11]. Therefore, C is a numerical representation of the complex system of stocks' interrelationships. To analyze those interrelationships, it is customary to transform C into a distance matrix D where its i-th row and j-th column is

$$\mathrm{d}(i,j) = \sqrt{2\{1 - c(i,j)\}}$$

for all $i, j = 1, 2, \ldots, n$. Then, the network among stocks is analyzed through (i) the network topology representing an MST in D to explore the topological properties of stocks, and (ii) the SDU of D to understand the classification of stocks in the form of a hierarchical tree [11].

The current practice to obtain an MST is by using Kruskal's algorithm on D. Once it has been obtained, it is then used to find the SDU. In what follows, we propose a method using the reverse direction; first we find the SDU and then use it to determine an MST. This will give us a faster algorithm; even faster if there is only one single MST in D. To begin the discussion, let us consider the distance d not only as a real non-negative function defined on $\mathrm{E} \times \mathrm{E}$ but also as a membership function of a fuzzy relation. As a real function on $\mathrm{E} \times \mathrm{E}$, d satisfies

(i) $0 \leq \mathrm{d}(i,j) < \infty$ and $\mathrm{d}(i,j) = 0$ if and only if $i = j$,
(ii) $\mathrm{d}(i,j) = \mathrm{d}(j,i)$, and
(iii) $\mathrm{d}(i,j) \leq \mathrm{d}(i,k) + \mathrm{d}(k,j)$.

for all i, j and k in E. On the other hand, from the fuzzy relation viewpoint, those properties show that d is the membership function in D viewed as a symmetric and anti-reflexive fuzzy relation [21]. From this point of view, Djauhari [14] shows that the SDU of D is the min–max transitive closure of D. To be more specific, let us consider k times min–max transitive operation '$*$' on D and itself defined, see Ref. [21], as

$$\mathrm{D}^{*k} = \mathrm{D} * \mathrm{D}^{*(k-1)} \quad \text{for all } k = 2, 3, \ldots$$

where

(i) $\mathrm{D}^{*1} = \mathrm{D}$,
(ii) the membership function d^{*k} in D^{*k} is

$$\mathrm{d}^{*k}(i,j) = \bigwedge_{m=1}^{n} \left\{\mathrm{d}(i,m) \vee \mathrm{d}^{*(k-1)}(m,j)\right\}, \tag{1}$$

(iii) $\mathrm{a} \wedge \mathrm{b} = \min\{\mathrm{a}, \mathrm{b}\}$ and $\mathrm{a} \vee \mathrm{b} = \max\{\mathrm{a}, \mathrm{b}\}$ for all real numbers a and b.

The fuzzy relation D^{+} defined by

$$\mathrm{D}^{+} = \mathrm{D} \cap \mathrm{D}^{*2} \cap \mathrm{D}^{*3} \cap \cdots,$$

is called min–max transitive closure of D. Since E is finite, then there exists a positive integer K; $0 < K < n$, such that Ref. [21]

$$\mathrm{D}^{+} = \mathrm{D} \cap \mathrm{D}^{*2} \cap \mathrm{D}^{*3} \cap \cdots \cap \mathrm{D}^{*K}.$$

This closure is exploited in Ref. [14] to construct the forest in D and then use it as a robust filter. Here, it will be used to construct an efficient algorithm to solve the MST problem. For this purpose, we recall the following two results given in Ref. [14].

(i) $\mathrm{D}^{+} = \mathrm{D}^{*K}$. This is the consequence of Theorem 1 in Ref. [14] which states that the sequence D, D^{*2}, D^{*3}, …, D^{*K} is monotone decreasing, i.e., $\mathrm{D}^{*K} \subseteq \cdots \subseteq \mathrm{D}^{*3} \subseteq \mathrm{D}^{*2} \subseteq \mathrm{D}$.
(ii) D^{+} is the SDU of D. See Theorems 3 and 4 in Ref. [14] for the proof.

These properties allow us to find the SDU in a very efficient way. It can be found by using iterative process where each iteration consists only of a matrix multiplication in the usual sense but multiplication and summation of two real numbers a and b are defined as $\max\{\mathrm{a}, \mathrm{b}\}$ and $\min\{\mathrm{a}, \mathrm{b}\}$ as described in Eq. (1). In the next section, these properties will be exploited to solve the MST problem by using Kruskal's algorithm applied to the forest and not to D. Of course, one can also use Prim's algorithm instead.

3. Proposed solution to MST problem

3.1. Construction of the forest

Once D^+ has been obtained, it can be used to construct the forest and not an MST. Let M be an MST in D and $(i = i_1, i_2, \ldots, i_p = j)$ be the chain from i to j in M. Then, according to Eq. (3) in Ref. [14], the distance $\widehat{\mathrm{d}}$ between i and j in M defined by

$$\widehat{\mathrm{d}}(i,j) = \bigwedge_{\gamma \text{ in } \Gamma} \bigvee_{k=1}^{p-1} \mathrm{d}(i_k, i_{k+1})$$

where Γ is given in Theorem 2 in Ref. [14], is the SDU of d. Therefore, there exists an integer m; $1 \leq m < p$, and an element γ in Γ such that

$$d^+(i,j) = \widehat{\mathrm{d}}(i,j) = \mathrm{d}(i_m, i_{m+1}).$$

Consequently, if Δ is a fuzzy relation where its membership function is given by

$$\delta(i,j) = \begin{cases} 1; & \mathrm{d}(i,j) - \mathrm{d}^+(i,j) = 0 \text{ and } i \neq j \\ 0; & \mathrm{d}(i,j) - \mathrm{d}^+(i,j) \neq 0 \text{ or } i = j, \end{cases} \tag{2}$$

then, Δ is the adjacency matrix that represents the forest.

Based on Eq. (2) and the properties of D^+ discussed in the previous section, the following algorithm can be used to find the forest [14]. In this algorithm, D and Δ are considered as matrices of size $(n \times n)$.

Step 1. Let $k = 2$,
Step 2. Compute D^{*k} where $\mathrm{D} * \mathrm{D}^{*(k-1)}$ is a matrix multiplication in the usual sense but multiplication and summation of two real numbers a and b are defined as max{a, b} and min{a, b}, respectively,
Step 3. If $\mathrm{D}^{*k} = \mathrm{D}^{*(k-1)}$, then the SDU of D is D^{*k} and go to Step 4. Otherwise, let $k := k + 1$ and then go back to Step 2,
Step 4. Compute Δ as defined in (2). Then Δ is the adjacency matrix representing the forest in D.

To increase the performance of this algorithm, as suggested in Ref. [14], we compute directly D^{*2}, D^{*4}, D^{*8}, ..., instead of D^{*2}, D^{*3}, D^{*4}, ... Therefore, to obtain D^{*2^K} as the SDU of D we need K iterations where $2^K \leq n$. As an illustration of that performance, the number of iterations to obtain the forest of the network among 100 stocks traded at NYSE is $K = 4$ and the running time is only 0.49 s by using Matlab version 7.8.0.347 (2009a).

3.2. Finding an MST in D

According to Eq. (2), the number of edges in the forest is $N/2$ where $N = \sum_{i=1}^{n} \sum_{j=1}^{n} \delta(i,j)$. By using this property we have a necessary and sufficient condition for the uniqueness of MST in D, formulated in Theorem 1, which will play an important role in the development of the proposed algorithm. See also Ref. [14].

Theorem 1. *The MST in D is unique if and only if* $N = 2n - 2$.

The algorithm to solve the MST problem that we are proposing here is constructed based on the properties of D^+, Theorem 1, and the following four properties of the forest.

Property 1. *If* D *contains more than one MST, all MSTs define the same SDU. It is also the SDU related to the forest. Let* M_1 *and* M_2 *be two MSTs in* D, *and* U_1 *and* U_2 *be their respective SDU. Then* $U_1 = U_2$. *Furthermore, if* U_3 *is the SDU that corresponds to the forest, then* $U_1 = U_2 = U_3$. *This is a direct consequence of the definition of* $L(\gamma)$ *and* Γ *in Theorem 2 in Ref.* [14]. *According to that definition, it might happen that;*

(i) *For a given* γ *in* Γ *there are two positive integers k and m such that* $\mathrm{d}(i_k, i_{k+1}) = \mathrm{d}(i_m, i_{m+1})$; $k, m = 1, 2, \ldots, p$.
(ii) *For two* γ_1 *in* γ_2 *in* Γ, *we have* $L(\gamma_1) = L(\gamma_2)$.

Property 2. *If M is an MST in the forest, then M is an MST in* D.

Property 3. *All leaves in the forest are also the leaves in any MST in* D.

Property 4. *Suppose we remove all leaves from* F *and denote* H *the remaining sub-graph of* F. *Then, we have two possibilities;*

(i) H *contains at least one leaf. In this case, leaves removal process is repeated until we get the sub-graph without any leaf, and*
(ii) H *contains no leaf.*

Let H^* *be the sub-graph of* F, *without any leaf, obtained after m repetitions of leaves removal process. If M is an MST in* H^*, *then the union of M and all removed leaves in the m-th repetition and all removed leaves in the* $(m-1)$*-th repetition and ..... all removed leaves in the first removal, is an MST of* D.

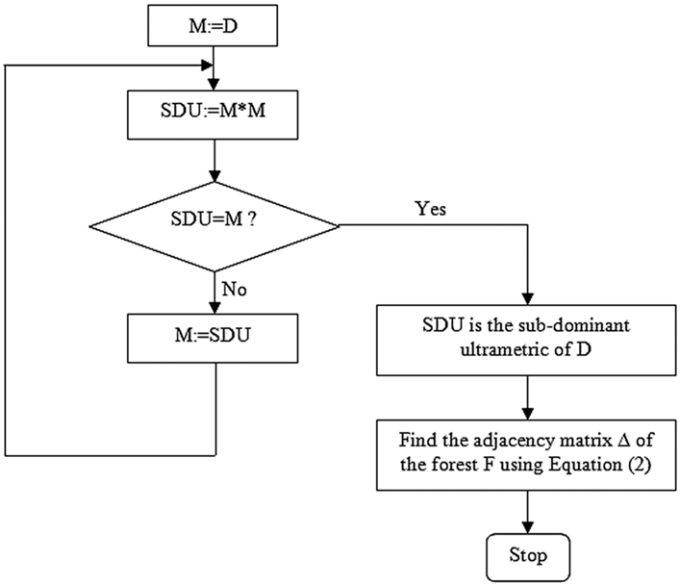

Fig. 1. Flow chart for constructing SDU and the forest.

Table 1
Hypothetical dissimilarity matrix D.

Node	*p*	*q*	*r*	*s*	*t*	*u*
p	0	0.1	0.3	0.2	0.1	2
q	0.1	0	0.3	0.4	1.7	2
r	0.3	0.3	0	0.6	0.5	2
s	0.2	0.4	0.6	0	1.7	2
t	0.1	1.7	0.5	1.7	0	0.7
u	2	2	2	2	0.7	0

Based on these properties and the properties of D^+ and the uniqueness theorem of MST, we propose to solve the MST problem by using the following four steps:

Step 1. Determine the SDU of the network D and then construct the forest F by using the algorithm described in Section 3.1. For clarity, in Fig. 1 we present the flowchart for constructing SDU and the forest.

Step 2. Test the uniqueness of MST in D by using Theorem 1. If the MST is unique, then it is the only element of the forest. Hence, we get the SDU and an MST and the process is stopped.

Step 3. If the MST is not unique, then find the sub-graph H^* described in Property 4.

Step 4. Find an MST in H^*, say M, by using Kruskal's algorithm or Prim's algorithm. Then, the union of M and all removed leaves described in Property 4 is an MST in D.

Diagrammatically, this method is presented in detail in Fig. 2 in the form of a flowchart. In that flowchart, to test the uniqueness of MST in the network, Theorem 1 can be used.

As an illustration on how the above algorithm works, consider a hypothetical dissimilarity network D of six nodes p, q, r, s, t, and u represented in Table 1. The forest F in D is given in Fig. 3(a). In that forest there is two leaves s and u. The sub-graph H of F after removing all leaves is in Fig. 3(b). The sub-graph H^* obtained after the second leaves removal is in Fig. 4(a). If an MST in H^*, say M, is such as given in Fig. 4(b), then the union of M and all removed leaves in the second removal and all leaves removed in the first, is an MST in D and represented in Fig. 5.

In the next section some experiences will be reported to show that the running time of this method is very promising.

4. How fast is the proposed method?

To illustrate the advantages of the proposed algorithm compared to Kruskal's algorithm and also Prim's algorithm, daily data of stock prices traded at four stock markets as well as data from simulation will be used. For each data set, we report the running time of each of those two algorithms.

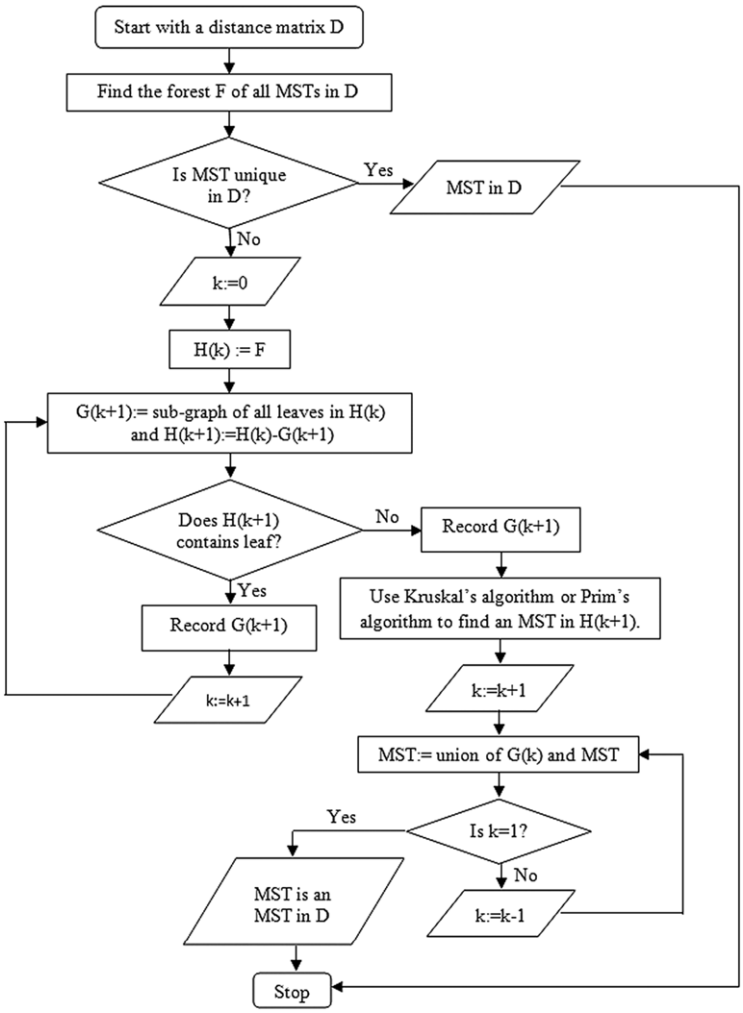

Fig. 2. Flowchart of the proposed algorithm.

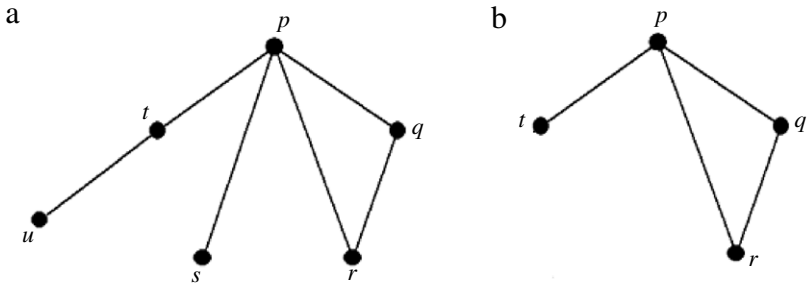

Fig. 3. (a) The forest F in D and (b) the sub-graph of F after the first leaves removal.

4.1. Running time comparison using real data

Daily stock prices data from Singapore Stock Exchange (SGX), Kuala Lumpur Stock Exchange (KLSE), New York Stock Exchange (NYSE), and Shanghai and Shenzhen Stock Exchange (SHSZ) are used in this study to illustrate the advantage of the proposed method in terms of the running time. There are 28 of the most capitalized stocks at SGX that we study in the period of January 2, 2008 until December 31, 2010. The daily price data were obtained from http://sg.finance.yahoo.com. At KLSE, NYSE, and SHSZ there are 90, 100, and 300 stocks that have been studied, respectively. The periods of study are,

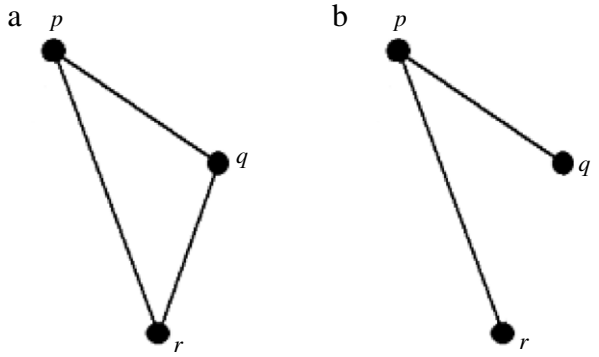

Fig. 4. (a) The sub-graph H* obtained after the second leaves removal and (b) an MST M in H*.

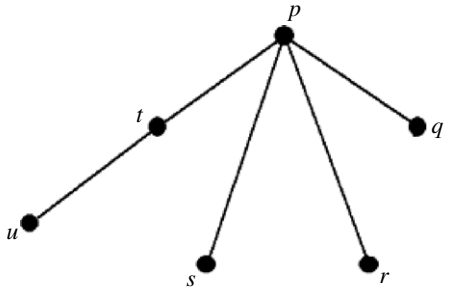

Fig. 5. An MST in D (the union of M and all removed leaves).

Table 2
Running time comparison for four stock markets.

Stock market	SGX	KLSE	NYSE	SHSZ
n	28	90	100	300
KA	0.087326	4.526329	7.785613	714.214164
PA	0.052719	0.413261	0.523906	7.892574
PAK	0.136161	0.422025	0.516481	7.267885
PAP	0.114484	0.396230	0.481573	6.669153

This table shows that, if n is small such as $n = 28$, PA dominates the others. If n is large, the proposed algorithm is faster than PA and even faster than KA.

respectively, from January 2, 2007 until December 31, 2009, from November 18, 2010 until March 28, 2012, and from July 28, 2011 until June 1, 2012. The data were downloaded from http://finance.yahoo.com.

In the last two rows of Table 2 we present the running time (in the second) of the proposed algorithms to find SDU, forest, and MST while in the third and fourth rows the running time to obtain MST using Kruskal's algorithm and Prim's algorithm, respectively, followed by SDU construction. In that table, KA, PA, PAK, and PAP stand, respectively, for

1. The algorithm to find an MST, using Kruskal's algorithm, and the SDU,
2. The algorithm to find an MST, using Prim's algorithm, and the SDU,
3. The proposed algorithm where Kruskal's algorithm is used to find an MST in H*,
4. The proposed algorithm where Prim's algorithm is used to find an MST in H*.

4.2. Running time comparison using data from simulation

Random nodes were simulated to construct a random distance matrix D of size $(n \times n)$ and then find an MST and SDU in D by using KA and PA, and find SDU, forest, and MST by using the proposed algorithm. The running time (in second) of each of these algorithms to get an MST, for $n = 75$, 150, 500, and 1000 is given in Table 3.

According to this table, for $n = 75$, 150 and 500, the computational efficiency of the proposed algorithm is similar to that of PA. It is higher than that of PA for $n = 1000$ and even higher than that of KA for all those values of n. It is also higher than in the case of real data as shown in Table 2. For example, the running time for $n = 75$ in a simulated network is higher than for $n = 90$ in a KLSE network. This phenomenon is quite reasonable because, for each value of n, the simulated network contains only one single MST.

Table 3
Running time comparison for four simulated networks.

n	75	150	500	1000
KA	0.793579	16.783875	825.200616	13654.491728
PA	0.283900	1.491310	30.369413	239.615803
PAK	0.275373	1.462183	30.531042	203.254527
PAP	0.275715	1.464457	30.551389	204.003943

4.3. Discussion

The current practice to filter important information in a network D among stocks is to find first an MST by using Kruskal's algorithm or Prim's algorithm. It is helpful to identify the topological properties such as the stock's centrality. Once an MST is obtained, it is also used to construct the SDU which is useful in economic classification of stocks in the form of an indexed hierarchical tree.

Since Kruskal's algorithm applies directly to D, its performance becomes slower and slower if the number of stocks gets larger and larger. This is the main concern of the present paper. We speed up its running time by changing the direction of computational process. If in the current practice the SDU is determined from an MST, in the proposed method an MST is determined from the SDU passing through the construction of the forest. By using the properties of the forest in a network, the search for an MST in D is reduced to the search of an MST in the forest. If the MST is unique, then it is the only element of the forest and hence we get the SDU and the MST. Otherwise, we construct the sub-graph H^* of F according to Property 4 in the previous section and then, by using Kruskal's algorithm, we find an MST in H^*, say M. The union of M and all removed leaves described in that property is an MST in D. This algorithm is considerably faster than KA and it is far faster if there is only one single MST in D.

5. Concluding remarks

There are a variety of algorithms to solve the MST problem efficiently. Among them, the prominent role is played by Kruskal's algorithm. But it is not the fastest algorithm. Once an MST has been obtained by using that algorithm, it is then used to find the SDU. This paper deals with an algorithm that can speed up Kruskal's algorithm. In the proposed algorithm first we find the SDU and the forest, and then an MST in the sub-graph H^* of the forest described in Property 4 by using Kruskal's algorithm, and finally we construct an MST in D. This algorithm is considerably faster than KA to obtain an MST and the SDU. This is very reasonable because, as degree distribution follows a power law [22], the number of nodes of low degree is large. In particular the number of leaves is the largest compared to the other nodes of degree more than one. Therefore, the number of nodes in H^* will be relatively small compared to n. Consequently, if D contains only one single MST, meaning that H^* is empty, the proposed algorithm is far faster than KA. To test the uniqueness of MST, Theorem 1 can be used; MST is unique if and only if the sum of all entries of the adjacency matrix Δ is $N = 2(n-1)$.

The algorithm proposed in this paper needs to find the SDU first and then the forest. Fortunately, based on the fuzzy relation approach, they can be found in a simple manner and operate quickly. We only need,

(i) matrix multiplication in the usual sense but multiplication and summation of two real numbers a and b are defined as max{a, b} and min{a, b}, respectively.
(ii) matrix subtraction in the usual sense.

Furthermore, according to Steps 1–4 in Section 3.1, the SDU can be found in only K iterations where $2^K \leq n$. Consequently, since Kruskal's algorithm is only used to find an MST in H^* and not in D, the proposed algorithm to find an MST in D is considerably faster than the current practice of Kruskal's algorithm applied on D.

It is important to note that the proposed algorithm provides us not only with MST and SDU but also the forest. This is another advantage of that algorithm compared to KA and PA which only give MST and SDU.

Acknowledgments

This research is sponsored by the Ministry of Higher Education, Government of Malaysia, through Fundamental Research Grant Schemes vote number 4F014 and Research University Grant vote number Q.J1300.7126.02H18. The author gratefully acknowledges the Ministry of Higher Education for the sponsorships and Universiti Teknologi Malaysia for the opportunity to conduct this research. Special thanks go to the Editor and anonymous referees for their constructive comments and suggestions that led to the final version of the presentation.

References

[1] M.S. Taqqu, in: H. German, et al. (Eds.), Bachelier and His Times: A Conversation with Bernard Bru, in: Mathematical Finance–Bachelier Congress 2000, Springer-Verlag, 2001.
[2] A.C. Silva, Applications of Physics to Finance and Economics: Returns, Trading Activity and Income, 2005, arXiv:physics/0507022v1, accessed on 17 June 2012.

[3] R.N. Mantegna, Hierarchical structure in financial markets, Euro. Phys. J. B 11 (1999) 193–197.
[4] M. Tumminello, T. Aste, T.D. Matteo, R.N. Mantegna, A Tool for Filtering Information in Complex Systems, in: Proceedings of the National Academy of Science, USA, 2005, pp. 10421–10426.
[5] G-J. Wang, C. Xie, F. Han, B. Sun, Similarity measure and topology evolution of foreign exchange markets using dynamic time warping method: evidence from minimal spanning tree, Physica A 391 (2012) 4136–4146.
[6] Y. Zhang, G.H.T. Lee, J.C. Wong, J.L. Kok, M. Prusty, S.A. Cheong, Will the US economy recover in 2010? A minimal spanning tree study, Physica A 390 (2011) 2020–2050.
[7] J.P. Onnela, A. Chakraborti, K. Kaski, J. Kert'esz, A. Kanto, Dynamic of market correlations: taxonomy and portfolio analysis, Phys. Rev. E. 68 (2003) 056110(1)–056110(12).
[8] V. Tola, F. Lillo, M. Gallegati, R.N. Mantegna, Cluster analysis for portfolio optimization, Econom. Dynam. Control 32 (2008) 235–258.
[9] T. Ulusoy, M. Keskin, A. Shirvani, B. Deviren, E. Kantar, C.C. Donmez, Complexity of major UK companies between 2006 and 2012: Hierarchical structure method approach, Physica A (2012). http://dx.doi.org/10.1016/j.physa.2012.01.026.
[10] S. Micciche, G. Bonanno, F. Lillo, R.N. Mantegna, Degree stability of a minimum spanning tree of price return and volatility, Physica A 342 (2003) 66–73.
[11] R.N. Mantegna, H.E. Stanley, An Introduction to Econophysics: Correlation and Complexity in Finance, Cambridge University Press, 2000.
[12] R. Zhuang, B. Hu, Z. Ye, Minimal Spanning Tree for Shanghai–Shenzhen 300 Stock Index, in: IEEE World Congress on Computational Intelligence, Hong Kong, 2008, pp. 1417–1424.
[13] B.M. Tabak, T.R. Serra, D.O. Cajueiro, Topological properties of stock market networks: the case of Brazil, Physica A 389 (2010) 3240–3249.
[14] M.A. Djauhari, A robust filter in stock networks analysis, Physica A 391 (2012) 5049–5057.
[15] J. Malkevitch, Trees: A Mathematical Tool for All Seasons, in: American Mathematical Society, Feature Column, May 2012.
[16] R.L. Graham, P. Hell, On the history of the minimum spanning tree problem, Ann. Hist. Comput. 7 (1985) 43–57.
[17] J. Nesetril, A few remarks on the history of MST-problem, Arch. Math. (Brno) 33 (1997) 15–22.
[18] F. Huang, P. Gao, Y. Wang, Comparison of Prim and Kruskal on Shanghai and Shenzhen 300 Index Hierarchical Structure Tree, in: International Conference on Web Information Systems and Mining, Shanghai, 2008, pp. 237–241.
[19] D. Cieslik, The vertex degree of minimum spanning trees, European J. Oper. Res. 125 (2000) 278–282.
[20] J.G. Brida, W.A. Risso, Hierarchical structure of the German stock market, Expert Syst. Appl. 37 (2010) 3846–3852.
[21] A. Kaufmann, Introduction a la théorie des sous ensemble flous: éléments théoriques de base, Vol 1, second ed., Masson, Paris, 1977.
[22] C. Eom, G. Oh, S. Kim, Topologies properties of a minimal spanning tree in the Korean and the American stock markets, J. Korean Phys. Soc. 51 (2007) 1432–1436.