Stock Prediction Based on HyperGraph Clustering

Department of Electrical and Computer Engineering

Panagiotis Tzimotoudis

Supervised by:

Katsaros Dimitrios

Antonopoulos Christos

October 2018

Περίληψη

Σε αυτό το έγγραφο, το οποίο είναι γραμμένο στην Αγγλική γλώσσα, παρουσιάζονται τρεις διαφορετικοί αλγόριθμοι ομαδοποίησης σε υπεργραφήματα, χρησιμοποιώντας δεδομένα από το χρηματιστήριο της Αμερικής. Παρουσιάζεται ο τρόπος που λειτουργούν, ο ορισμός τους και γίνονται για τον καθένα πειράματα, με την ίδια είσοδο. Επιπλέον, παρουσιάζεται ένας αλγόριθμος εύρεσης κοινών χαρακτηριστικών διάφορων στοιχείων που θα χρησιμοποιηθεί για την δημιουργία του υπεργραφήματος των μετοχών. Η υλοποίηση των αλγορίθμων που χρησιμοποιήθηκαν έγινε σε Python, εκτός από τα συστήματα ομαδοποίησης υπεργραφημάτων που χρησιμοποιήθηκαν τα πακέτα που προσφέρονται δωρεάν από τους δημιουργούς τους.

Έπειτα, τα αποτελέσματα συγκρίνονται μεταξύ τους με βάση τα πραγματικά μελλοντικά δεδομένα του χρηματιστηρίου, προσπαθώντας να προβλέψουμε τις κινήσεις των μετοχών μία συγκεκριμένη μελλοντική ημέρα. Χρήσιμα συμπεράσματα αποκομούνται την αποδοτικότητα και την ορθότητά τους στον συγκεκριμένο τομέα.

Abstract

In this thesis, the use of various hypergraph clustering algorithms is examined, some of which are commonly used in the scientific community, in hypergraph networks and their application in stock prediction. Stock prediction is the study of stock market trends to predict future stock prices. After creating the hypergraph using one of asssociation rules algorithms eg. Apriori [6], three algorithms are used in this project to find stock clusters and these are by order: hmetis [13, 14], KaHyPar [21] and PaToH [7]. Each of these has its own advantages, however, the best method of these three is defined based on which one produces more accurate results.

Acknowledgments

This thesis is dedicated to my family, for always supporting me and be there for me.

I would like to thank my supervisor Professor Dimitrios Katsaros, for the opportunity he gave me and his constant help on the production of this project.

Special thanks to Giorgios Tziokas, student of University of Thessaly, for providing feedback on this project and the current document.

Panagiotis Tzimotoudis, Volos, September 2018

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The extension of conventional clustering to hypergraph clustering, which involves higher order similarities instead of pairwise similarities, is increasingly gaining attention in many scientific areas. This is due to the fact that many clustering problems require an affinity measure that must involve a subset of data of size more than two. Hypergraph clustering can be very usefull to solve some problems in variety of social, biological, and technological fields, including data mining, wireless communications, computer vision, very large scale integration circuits (VLSI), and stock prediction.

In this chapter, hypergraph theory will be introduced. In the following, hypergraph clustering problem and stock prediction problem will be introduced.

1.1 Hypergraph Thoery

Hypergraphs are generalization of graphs having edges, called hyperedges that connect more than two vertices. As shown in Fig. 1-1 a hyperedge is an edge that can be any subset of a given set of vertices rather than two-element subsets. In Fig. 1-1, we can see that e_5 is a two-element set of vertices (4,7), so edge e_5 is both edge and hyperedge, but e_1 is a three-element subset of vertices (4,5,6) called hyperedge. Following the nomenclature of [23], we have:

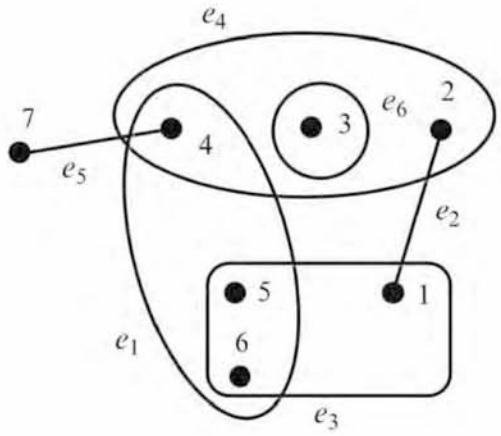


Figure 1-1: An example of hypergraph

Definition 1.1. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a finite set, and let $E = \{e_1, e_2, \ldots, e_n\}$ be a familt ob subsets of X such that:

$$\begin{aligned} e_i \neq (i = 1, 2, \ldots m), \\ \bigcup_{i=1}^{m} e_i = X. \end{aligned} \tag{1.1}$$

The pair $H = (X, E)$ is called a hypergraph with vertex set X and hyperedge set E. The elements $x_1, x_2, \ldots, x_n$ of X are vertices of Hypergraph H, and the sets $e_1, e_2, \ldots, e_n$ are hyperedges of hypergraph H.

In addition, $E(x), x \in X$ denotes the set of all the hyperedges which contains vertex x. The cardinality of $E(x), i.e., |E(x)|$, is called the degree of vertex x. The maximum degree of the hypergraph H is denoted by:

$$\Delta(H) = \max_{x \in X} |E(x)|. \tag{1.2}$$

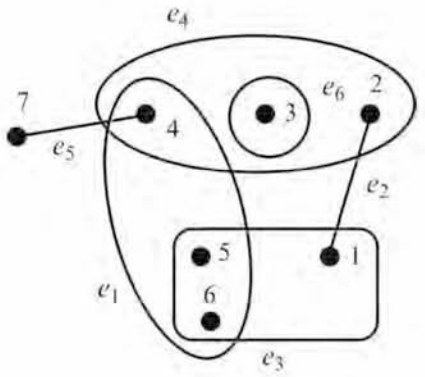
A hypergraph in which each vertex in the vertex set is with the same degree $k > 0$ is called k-regular.Also a hypergraph in which each hyperedge in the hyperedge set have the same degree of cardinality $r > 0$ is called r-uniform. It is clear, that if all hyperedges have same degree $r = 2$ the hypergraph will be two-uniform graph and it will be a standard graph where a hyperedge connects two verticies like the standard edge.

Definition 1.2. The incidence matrix of a hypergraph $H(X, E)$ is a matrix $I(H)$ with rows representing the vertices and columns representing the hyperedges of H such that:

$$I(i,j) = \begin{cases} 1, & \text{when} \quad x_i \in e_j \\ 0, & \text{when} \quad x_i \notin e_j \end{cases} \tag{1.3}$$

An example of the incidence matrix is shown in Fig. 1-2. In contrast with regular graph, which can be specified with both its incidence matrix or its adjacency matrix, in hypergraph there is no one-to-one correspondence with its adjacency matrix, so a hypergrapg can only be determined by its incidence matrix. An entry of 1 in location (i, j) where i corresponds to i-th vertex and j corresponds to j-th hyperedge, means that this vertex belongs to this hyperedge. On the contrary an entry of 0 in location (i, j) means that the i-th vertex is not part of the j-th hyperedge. Similar to the

incidence matrix of graph, it also follows that empty hyperedges mean zero columns and isolated vertices mean zero rows in the incidence matrix. If the vertex set of a hypergraph is empty, then the incidence matrix consists only of the row containing the names of hyperedges. Similarly, if the hyperedge set is empty, then the incidence matrix consists only of the column containing the names of vertices.



	e_1	e_2	e_3	e_4	e_5	e_6
1	0	1	1	0	0	0
2	0	1	0	1	0	0
3	0	0	0	1	0	1
4	1	0	0	1	1	0
5	1	0	1	0	0	0
6	1	0	1	0	0	0
7	0	0	0	0	1	0

H = (X,E) I(H)

Figure 1-2: Example of a hypergraph and its incidence matrix I(H)

1.2 Hypergraph Clustering

CLustering is a grouping task that intends to group a set of objects, in our case vertices, in such way that objects who belong in the same group, called a cluster, are more similar to each other, in some way, than to those who belongs in other groups. The theory behind the hypergraph clustering generalises the the traditional idea of clustering, whereby the affinity measure is now defined over more than a pair of objects. The process itself become more complex than the corresponding proccess of graph clustering, beacause, the graph itself become more complex and the similarity of objects begin to be more uncertain.

In the graph partitioning view, a partitioning of H split the set of objects (vertices) into K discrepant clusters. Especially, a two-way partitioning results in (V_1, V_2), where $V_1 \bigcup V_2 = V$ and $V_1 \bigcap V_2 = 0$. The "goodness" of the partitioning is inversely proportional to the cost of the cut that separates the vertices. Many methods have been proposed to find the best K-way partitioning, given an arbitrary hypergraph H.

1.3 Stock prediction

Stock prediction is the act of trying to determine the future value of a company stock, or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. But many studies show that stocks belonging to the same group tend to rise or fall together in a specific time-space.

So instead of trying to predict the actual prices of stocks, clustering technics will be used to group the stocks. After that, an algorithm will be used to predict an increase or decrease of the stocks prices in specific future date.

Chapter 2

Input Data

The main input data for the tests applied on algorithms was a dataset extracted from S&P 500 index[1], which is an American stock market index based on the capitalizations of 500 large companies.

In this chapter, data extraction will be representing. In the following, the data transformation will be explained and the hypergraph creation approach will be introduced.

2.1 S&P 500 extraction

The data was extracted from the pandas-datareader[2] which is a Python [3] library primarly has the form as shown in Table 2.1. The table consists of the date that we are referring, the opening price in US dollars, the higher and lower price of the stock in this particular date, the closing price, the volume which is the number of shares or contracts traded in a security or an entire market during a given period of time and the reference name of the stock (eg. AAL is the reference name of American Airlines Group Inc.).

date	open	high	low	close	volume	Name
2016-08-11	33.64	34.4552	33.6302	34.3374	7666723	AAL
2016-08-12	34.1213	34.5436	34.0427	34.2981	6064005	AAL
2016-08-15	34.3374	35.408	34.2784	34.2312	7137204	AAL
2016-08-16	35.0347	36.1348	34.9954	34.0955	9384959	AAL
2016-08-11	105.0504	105.4473	104.4019	104.4793	27484506	AAPL
2016-08-12	104.3341	104.973	104.3341	104.7213	18660434	AAPL
2016-08-15	104.6826	106.0378	104.6245	105.9798	25868209	AAPL
2016-08-16	106.125	106.7058	105.7184	105.883	33794448	AAPL
2016-08-11	167.1584	169.7684	166.0228	166.0726	829368	AAP
2016-08-12	165.4549	165.8634	162.8649	165.3653	1165191	AAP
2016-08-15	166.1124	169.2504	164.917	167.1185	1854564	AAP
2016-08-16	161.3806	164.7676	159.7269	159.7867	3030792	AAP
2016-08-11	55.1848	55.9403	54.4637	55.1942	9353242	ABBV
2016-08-12	60.159	60.7174	56.6958	58.7887	18693998	ABBV
2016-08-15	59.1829	61.0506	58.9295	60.0651	15948718	ABBV
2016-08-16	59.6334	59.9243	58.9295	59.2204	10201578	ABBV

Table 2.1: Example of S&P 500 extracted data

2.2 S&P 500 transformation

Because the information shown in Table 2.1 is not very handy additional algorithm running over this data to create a binary table as shown in Table 2.2. Two datasets will be used in this project. The first data set consists of a binary table 1010 x 525. Each row of this table corresponds to up or down movement indicator for one of the 505 stocks in S&P 500, and each collumn corresponds to one of 525 trading days from 1st Mar. 2016 to 31st Mar. 2018. The second data set consists of a binary table 1006 x 525. Similarly as the first data set each row of this table corresponds to up or down movement indicator for one of the 503 stocks in S&P 500, and each collumn corresponds to one of 525 trading days from 1st Sept. 2015 to 30st Sept. 2017. An entry of 1 in each data set in location (i,j) where i corresponds to up indicator of a stock means that the closing price of this stock on j-th day is significantly higher (2% or 1/2 point or more) than the day before. Similarly, an entry of 1 in location (i,j) where i corresponds to down indicator of a stock means that the closing price of this stock on j-th day is significantly lower (2 % or 1/2 point or more) than the day before.

		2016-08-11	2016-08-12	2016-08-15	2016-08-16
Up Indicator	AAL	0	0	1	1
	AAPL	0	0	0	0
	AAP	0	0	0	0
	ABBV	0	1	1	0
Down Indicator	AAL	0	0	0	0
	AAPL	0	0	0	0
	AAP	0	0	0	1
	ABBV	0	0	0	0

Table 2.2: Example of binary table based on the extracted data used in Table 2.1

2.3 HyperGraph Creation

In order to create the hypergraph from the binary table, we can assume that the hyperedges will be the set of vertices that have the same up or down indicator for each day. This process will lead to a very small amount of hyperedges and in some cases, this relation between two or more vertices may be useless. So to overcome these problems in the determination of related items that can be grouped as hyperedges and to include some useful information, we will use frequent item sets computed by an association rule algorithm. In this project, the association rule algorithm that will be used is the well known Apriori [6] algorithm to find related item sets.

2.3.1 Apriori

One of the first algorithms for mining all frequent itemsets and strong association rules trying to solve the problem of mining association rules over basket data was introduced by in [5]. Shortly after that, the algorithm was improved and renamed to Apriori [6]. Apriori algorithm is, the most classical and important algorithm for mining frequent itemsets. Apriori is used to extract all frequent itemsets from input data that consists of a set of transactions, where each transaction is a set of items.

An example of an association rule migth be that 98% of customers that purchase tomatos and bread also get and cheese.

The following is a formal statement of the problem as described in [5]. Let $I = \{i_1, i_2, \cdots, i_n\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Associated with each transaction is a unique identifier, called its TID. We say that transaction T contains X, a set of some items in I, if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = 0$. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence c if c% of transactions in D that contains X also contains Y. The rule $X \Rightarrow Y$ has a support s in the transaction set D if s% of the transactions in D contains $X \cap Y$.

Given a set of transactions D, the problem of mining association rules is to generate all association rules that have support and confidence greater than the user-specified minimum support (called minsup) and minimum confidence (called minconf) respectively.

Apriori Algorithm Description

The key idea behind the Apriori is to make multiple passes over the dataset. It make use of breadth-first search to iterate through the search space, where k-itemsets are used to find (k+1)-itemsets. The basic steps to mine the frequent elements are as follows:

- Generate and Test: In this step, find the 1-itemset frequent elements L_1 by scanning the database and removing all those elements from C_1 which cannot satisfy the minimum support criteria.
- Join step: To proceed in the next step all elements of C_k join the previous frequent elements by self join i.e. $L_{k-1} * L_{k-1}$, which is also known as Cartesian product of L_{k-1}. This step generates new candidate k-itemsets based on joining

L_{k-1} with itself which is found in the previous iteration. Let C_k denote candidate k-itemset and L_k be the frequent k-itemset.

- Prune step: In this step let C_k be the superset of L_k so members of C_k may be frequent or not, but all $K-1$ frequent itemsets will be included in C_k thus prunes the C_k to find K frequent itemsets with the help of Apriori property. In this step, some of the candidate k-itemsets will be removed using the Apriori preperty. A scan of the database to determine the count of each candidate in C_k would result in the determination of L_k, all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k. C_k, however, can be very huge, so in order to shrink its size, the Apriori property is used as follows:

  - Any (k-1)-itemsets that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k-1)-subset of candidate k-itemset is not in L_{k-1} then the candidate cannot be frequent either and so can be removed from C_k.

The last two steps (Join step and Prune step) is repeated until no new candidate set is generated.

Apriori Algorithm

Summarizing all the above and as described in [6] we have the following:

k-itemset	An itemset having k items
L_k	Set of large k-itemsets (those with minimum support). Each member of this set has two fields: i) itemset and ii) support count.
C_k	Set of candidate k-itemsets (potentially large itemsets). Each member of this set has two fields: i) itemset and ii) support count.

Table 2.3: Notation used in Apriori algorithm

Algorithm 1 Apriori Algorithm

```
1: procedure APRIORI
   Input: Transactions list, minsup (minimum support)
   Output: Large itemsets
2:     L_1 = {large 1-itemsets};
3:     for (k = 2; L_{k-1} ≠ 0; k++) do
4:         C_k = apriori_gen(L_{k-1}); // New candidates
5:         for all transactions t ∈ D do
6:             C_t = subset(C_k, t); // Candidates contained in t
7:             for all candidates c ∈ C_t do
8:                 c.count++;
9:         L_k = {c ∈ C_k | c.count ≥ minsup};
10:    Answer = ∪_k L_k
```

The apriori_gen function takes as argument L_{k-1}, the set of all large (k-1) -itemsets. The function works as follows:

First, in the join step L_{k-1} and L_{k-1} will be joined.

```
1: insert into C_k
2: select p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
3: from L_{k-1} p, L_{k-1} q
4: where p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-2} < q.item_{k-2};
```

Next, in the prune step, all itemsets c $\in C_k$ such that some (k-1)-subset of c is not in L_{k-1} will be deleted:

```
1: for all itemsets c ∈ C_k do
2:     for all (k-1)-subsets s of c do
3:         if (s ∉ L_{k-}) then
4:             delete c from C_k
```

Apriori Example

As an example Table 2.4, 2.5, 2.6 of running the Apriori algorithm we get some hypothetical data from a super market [4]. Assuming that the "Tid" be the hypothetical transaction identifier and "items bought" the hypothetical items bought in each transaction. Also, assume that the min support will be 2, then we wil have:

Tid	Items Bought
1	Milk,Tea,Cake
2	Eggs,Tea,Cold Drink
3	Milk,Eggs,Tea,Cold Drink
4	Eggs,Cold Drink
5	Juice

Items Bought	Support
Milk	2
Eggs	3
Tea	3
Cold Drink	3
Juice	1
Cake	1

Items Bought	Support
Milk	2
Eggs	3
Tea	3
Cold Drink	3

Items Bought
Milk, Eggs
Milk, Tea
Milk, Cold Drink
Eggs, Tea
Eggs, Cold Drink
Tea, Cold Drink

Table 2.4: First Iteration of Apriori algorithm

Items Bought	Support
Milk, Eggs	1
Milk, Tea	2
Milk, Cold Drink	1
Eggs, Tea	2
Eggs, Cold Drink	3
Tea, Cold Drink	2

Items Bought	Support
Milk, Tea	2
Eggs, Tea	2
Eggs, Cold Drink	3
Tea, Cold Drink	2

Items Bought
Milk, Tea, Eggs
Milk, Tea, Cold Drinks
Eggs, Tea, Cold Drink

Table 2.5: Second Iteration of Apriori algorithm

Items Bought	Support
Milk, Tea, Eggs	1
Milk, Tea, Cold Drinks	1
Eggs, Tea, Cold Drink	2

Items Bought	Support
Eggs, Tea, Cold Drink	2

Table 2.6: Third Iteration of Apriori algorithm

In the first table in Table 2.4 the data in the database will present. Each transaction has its own unique identifier Tid and the items bought in this transaction. In the second table the support (frequency) of each item will be computed. After this step, in the third table all items with support less than the minimum support threshold, which defined in the beginning of the Apriori algorithm, will be removed from the item list. Finally, in the fourth table, the remaining items will be combined to generate the two-items item set.

In the second Table 2.5 the second iteration of Apriori will be present. Firstly, in the first table the support of all items from two-items item set will be computed. After that, all items with support less than than the minimum support threshold, will be removed from the item list as shown in the second table. At the end of the second iteration, as we can see in the third table all the remaining items will be combined to generate the three-items item set.

Finally, in the third and last iteration, for this example, of Apriori (Table 2.5) the support of all items from three-items item set will be computed as shown in the first table. After removing all the items that have support less than the minimum support we get that only one itemset is frequent (Eggs, Tea, Cold Drink). So at the end of Apriori running we will get nine frequent itemesets as shown in Table 2.7.

Itemsets	Itemsets	Itemsets
{Milk}	{Cold Drink}	{Eggs, Cold Drink}
{Eggs}	{Milk, Tea}	{Tea, Cold Drink}
{Tea}	{Eggs, Tea}	{Eggs, Tea, Cold Drink}

Table 2.7: Apriori result itemsets

2.3.2 HyperGraph Modeling

In order to create the hypergraph from the previously detailed binary table using the Apriori algorithm, two things must be computed. The first is to find the hyperedges, which is the easy part of the hypergraph creation. All the frequent itemsets produced from Apriori, with minimum support being equal to 0.04 which means that all stocks in a frequent itemset must have moved together for at least 21 days, will be the hyperedges for our hypergraph.

The second thing that must be computed, and the more tricky, is the assignment of weights in each resulting hyperedge. As proposed in [11, 10], weights in hyperedges will be a function of the confidence of the underlying association rules. For example, if {A, B, C} is a frequent itemset, then the hypergraph will contain a hyperedge that will connect A, B, and C. Consider the following rules for this itemset {A, B, C}: $\{A, B\} \stackrel{0.4}{\Longrightarrow} \{C\}$, $\{A, C\} \stackrel{0.4}{\Longrightarrow} \{B\}$, $\{B, C\} \stackrel{0.4}{\Longrightarrow} \{A\}$. Then a weight of 0.6 ($\frac{0.4+0.6+0.8}{3} = 0.6$) will be assigned to the hyperedge that connects A, B, and C.

Chapter 3

Clustering Algorithms

After creating the hypergraph, the next step in this project is to partition the stocks in proper clusters by partitioning the vertices (stocks) into k roughly equal parts, such that the number of hyperedges connecting vertices in different parts is minimized. To achieve that, one of the following partitioning tools (hMetis, KaHyPar, and PaToH) will be used.

In this chapter, three partitioning algorithms will be representing and by order these will be hMetis, KaHyPar, and PaToH. In the following, two algorithms for making the clusters more connected will be introduced.

3.1 hMetis

hMetis[16, 12, 15] is a well known and state of the art software package for partitioning large hypergraphs. It was original developed for partitioning large hypergraphs that arising in circuit design, and more specific in VLSI, but it has very good capabilities for partitioning any hypergraph. The algorithms used by hMetis are based on multilevel hypergraph partition. The multilevel algorithm consists of 4 phases: the coarsening phase, the initial partition phase, the uncoarsening and refinement phase, and the v-Cycle refinement phase. Unlike, the traditional graph partitioning

algorithms that compute a partition of a graph or a hypergraph directly on the original graph, multilevel partitioning algorithms take a completely different approach as shown in Fig. 3-1. The traditional algorithms are often too slow and very often produce poor quality of partitions especially in large hypergraphs. The multilevel algorithms, on the other hand, can produce very quickly high-quality partitions for a large variety of hypergraphs.

As detailed in [16] and shown in 3-1, the multilevel algorithm consists of the following phases:

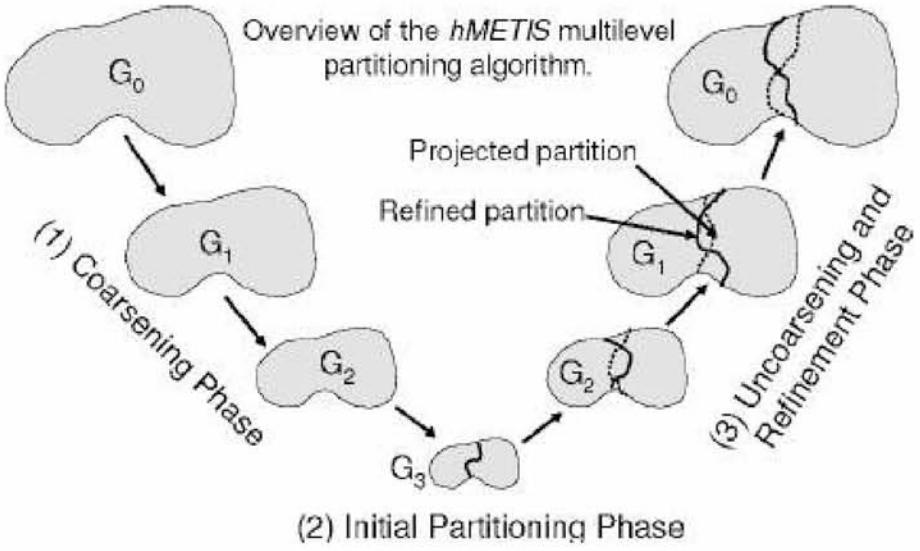


Figure 3-1: Multilevel partitioning algorithms

- **Coarsening Phase**: During the hypergraph coarsening phase, a sequence of successively smaller hypergraphs is constructed. The purpose of coarsening is to create a small hypergraph, such that a good bisection of the small hypergraph is not significantly worse than the bisection directly obtained for the original hypergraph. In addition to that, hypergraph coarsening also helos in successively reducing the size of the hyperedges. That is particularly hepful, since refinement heuristics based on Kernighan-Lin [9, 17], Appendix A.2, algorithm are very effective in refining small hyperedges but are quite ineffective in

refining hyperedges with a large number of vertices belonging to different partitions. The group of vertices that are contracted together to form single vertices in the next level coarse hypergraph can be selected in different ways. hMetis implements various such grouping schemes,which is also known as matching schemes. More specific in shMetis, which is the algorithm used in this project, the hybrid first-choice scheme (HFC) is implemented. This scheme is a combination of the first-choice (FC) and greedy first-choice (GFC) schemes. In the first-choice scheme vertices are grouped together if they are present in multiple hyperedges. Groups of vertices of arbitary size are allowed to be collapsed together. In greedy first-choice scheme vertices are grouped based on the first-choice scheme, but the grouping is biased in favor of faster reduction in the number of the hyperedges that remain in the coarse hypergraphs.

- **Initial Partitioning Phase**: During the initial partitioning phase, a bisection of the coarsened hypergraph is computed. In this many different algorithms can be used without significantly affecting the overall runtime and quality of the algorithm, because in most cases the hypergraph will be having a small number of vertices. More specific in shMetis, a multiple random bisection followed by the Fiduccia-Mattheyses (FM) [9] ,Appendix A.1, refinement algorithm will be used.

- **Uncoarsening and Refinement Phase**: During the uncoarsening phase, the partitioning of the coarsest hypergraph is used to obtain a partitioning for the finer hypergraph. This is done by successively projecting the partitioning to the next level finer hypergraph and using a partitioning refinement algorithm to reduce the cut and thus improve the quality of the partitioning. Since the next level finer hypergraph has more degrees of freedom, such refinement algorithms tend to improve the quality. As described in [16] hMetis and especially in shMetis, that is used here, implements an FM based algorithm (Appendix A.1).

- **v-Cycle Phase**: The idea behind this refinement algorithm is to use the power of the multilevel paradigm to further improve the quality of a bisection. The

V-cycle refinement algorithm consists of two phases, namely a coarsening and an uncoarsening phase. The coarsening phase preserves the initial partitioning that is input to the algorithm. This will be refered as restricted coarsening scheme. In this restricted coarsening scheme, the groups of vertices that are combined to form the vertices of the coarse graphs correspond to vertices that belong only to one of the two partitions. As a result, the original bisection is preserved through out the coarsening process, and becomes the initial partition from which the refinement performing during the uncoarsening phase is started. The uncoarsening phase of the V -cycle refinement algorithm is identical to the uncoarsening phase of the multilevel hypergraph partitioning algorithm described earlier. It moves vertices between partitions as long as such moves improve the quality of the bisection. Note that the various coarse representations of the original hypergraph, allow refinement to further improve the quality as it helps it climb out of local minima. In shmetis the v-Cycle refinement performs on the final solution of each bisection step.

3.2 PaToH

PaToH [7] is developed by Catalyurek and Aykanat and it is another state of the art hypergraph partitioning tool. Patoh is a fast multilevel recursive bipartitioning based tool that supports partitioning with fixed vertices and multi-constrained objectives. It is optimized for partitioning sparse matrix instances and uses multilevel paradigm as shown in Fig. 3-1.

As mentioned and before and detailed in [7], the multilevel algorithm, implemented in PaToH, consists of the following phases:

- **Coarsening Phase**: During the hypergraph coarsening phase, as detailed and in hMetis algorithm 3.1, a sequence of successively smaller hypergraphs is constructed. For this purpose PaToH uses two algorithms: the heavy connectivity matching algorithm (HCM) and the heavy connectivity clustering algorithm

(HCC).

- **Heavy Connectivity Matching (HCM)** is a matching based coarsening scheme. In this scheme, if a hypernode v should be matched with another hypernode, then all unmatched hypernodes $u \in \Gamma(v)$ are consider. The set $\Gamma(v) = \{u | \exists e \in E : v \in E \wedge u \in E\}$ defines all adjacent vertices of a hypernode v. Also a vertex $u \in \Gamma(v)$ is called neighbour of v. The hypernode u with the largest edge weight between v and u forms a matching pair (u, v). The weight of an edge between two hypernodes u and v is defined as the number of all common incident nets that u and v belongs to. Formally, the weight of an edge $\omega' : V \times V \rightarrow \mathbb{R}$ between to hypernodes u and v in a hypergraph H as follows:

  $$\omega'(u, v) = |I(u) \bigcap I(v)|$$

  I is defined as $I : V \rightarrow P(E)$, which maps a vertex v to all its incident nets.The contraction partner $u \in \Gamma(v)$ maximize the $\omega'(u, v)$. In this scheme, the hypernodes are visited randomly.

- **Heavy Connectivity Clustering (HCC)** is a method for finding higly connected clusters. At the beginning of these method all hypernodes are sigleton clusters $C_u = u$. In order to find a contraction partner for a sigleton cluster this method consider all incident sigleton and multinode clusters. This method choose the cluster $C_v = v$ as contraction partner for $C_u = u$ the cluster that maximizes the following rule:

  $$\omega''(C_u, C_v) = \frac{|I(u) \bigcap (\bigcup_{v \in C_v} I(v))|}{c(u \bigcap C_v)}$$

  With the division $c(u \bigcap C_v)$ very heavy clusters should be avoided.

- **Initial Partitioning Phase**: During the initial partitioning phase, as detailed and in hMetis algorithm 3.1, a bisection of the coarsened hypergraph is computed. PaToH impliments eleven different initial partition methods which can

be categories into random, hypergraph growing and greedy hypergraph growing variants. In this project, only the Greedy Hypergraph Growing (GHG) algorithm is be used, so for all the other different variants, we refer to PaToH manual [8]. In GHG, a cluster around a randomly selected vertex is growing. During the coarse of the algorithm, the selected and unselected vertices induce a bipartition on the coarsest hypergraph H_m. The unselected vertices connected to the growing cluster are inserted into a priority queue according to their FM gains. In this step, the gain of an unselected vertex corresponds to the decrease in the cut size of the current bipartition if the vertex moves to the growing cluster. Then, a vertex with the highest gain is selected from the priority queue. After a vertex moves to the growing cluster, the gains of its unselected adjacent vertices which are currently in the priority queue are updated and those not in the priority queue are inserted. This cluster growing operation continues until a predetermined bipartition balance criterion is reached. Because the quality of this algorithm is sensitive to the choice of the initial random vertex, PaToH runs GHG algorithm multiple times starting from different random vertices and select the best bipartition for refinement during the uncoarsening phase.

- **Uncoarsening and Refinement Phase**: During this phase, as detailed in [7], at each level i (for i = m, m-1, ..., 1), bipartition Π_i found on H_i is projecting back to bipartition Π_{i-1} on H_{i-1}. The constituent vertices of each multinode in H_{i-1} is assigned to the part of the respective vertex in H_i. Then this bipartition is refined by running a Boundary FM (BFM) hypergraph bipartitioning algorithm on H_{i-1} starting from the initial bipartition Π_{i-1}. BFM moves only the boundary vertices from the overload part to the underloaded part, where vertex is said to be a boundary vertex if it is connected to an at least on cut net. The refinement stops if no feasible move remains or $max(50, 0.001|V|)$ moves with no decrease into the cut are performed.

3.3 KaHyPar

KaHyPar [21] is a multilevel hypergraph partitioning framework for optimizing the net-cut of the hypergraph. As a multilevel algorithm, like hMetis, it consists of three phases: the coarsening phase, the initial partition phase, and the uncoarsening and refinement phase. KaHyPar, in contrast, the other two partitioning tools is an open source tool.

As a multivel algorithm, KaHyPar also consists of three main phases. The coarsening phase, the initial partitioning phase and the uncoarsening phase. A general overview of this hypergraph partitioning framework is summarised in Algorithm 2, and the details of the main phases will be described below.

Algorithm 2 Multilevel Hypergraph Partitioning

1: **procedure** MULTILEVEL HYPERGRAPH PARTITIONING
   Input: Hypergraph H, number of desired blocks k, balance parameter ϵ
   Output: ϵ-balanced k-way partition $\Pi = V_1, ..., V_k$
2: **while** H is not small enough **do** //coarsening phase
3: $(u, v) := argmax_{u \in V} score(u)$ // choose vertex pair with highest rating
4: $H := contract(H, u, v)$ // H := H$\setminus\{v\}$
5: $\Pi := partition(H, k, \epsilon)$ // initial partitioning phase
6: **while** H is not completely uncoarsened **do** //uncoarsening phase
7: $(H, \Pi, u, v) := uncontract(H, \Pi)$
8: $(H, \Pi) := refine(H, \Pi, u, v, k, \epsilon)$

- **Coarsening Phase**: During the hypergraph coarsening phase, as detailed and in [24], the main goal is to contract highly connected vertices such that the number of the remaining nets in the hypergraph and their size to be reduced. This approach leads to simpler instances for the initial partitioning and allow the FM-based local search algorithms to identify more moves that improve the quality of the partitioning solution based on small net sizes. In order to achieve this, Kahypar's coarsening algorithm prefers vertex pairs that have a large number of heavy nets with small size in common. This coarsening process is repeated until the number of remaining vertices is below a specific threshold

or the priority queue becomes empty.

- **Initial Partitioning Phase**: After the coarsening phase is finished, the hypergraph will be small enough to be initial partitioned by an initial partition algorithm. KaHyPar framework does not impliments a new algorithm, but instead uses the recursive bisection variant of hMetis for the initial partioning proccedure. Because this variant can be defined differently, KaHyPar, uses an imbalance parameter which defines as follows:

$$\epsilon' := 100((\frac{1+\epsilon}{k} + \frac{max_{v \in V} c(v)}{c(V)})^{\frac{1}{log_2(k)}} - 0.5)$$

for initial partioning with hMetis. The initial partioner will be called multiple times with different random seeds and KaHyPar will be use the best partition as the initial partioning of the coarsest graph.

- **Uncoarsening and Refinement Phase**: During this step, the initial solution produced in the previous step is transfered to the next finer level by performing a sigle uncontraction step. After that, KaHyPar uses a localized local search algorithm in order to improve further the solution quality. The local search algorithm used by KaHyPar follows ideas similar to the k-way FM-based algorithm proposed by Sanchis [20] and inspired further by the local search algorithm used by Sanders and Osipov [19]. Counter to Sanchis algorithm, KaHyPar reduce the number of priority queues to k, one queue P_i for each cluster V_i. Additionaly, they only consider to move a vertex to adjacent blocks rather that calculationg and maintaining gains for moves to all blocks, as in the case of Sanchis. The local search pass started by performing a highly localized local search starting with only the represenative and the just uncontracted vertex. The search procedure then expands around this vertex pair by successively inserting moves for neighboring vertices into the queues.

3.4 Fitness & Connectivity Measures

As proposed in [11, 10] it is important to eliminate bad clusters and to remove vertices that are not highly connected to the rest of the vertices of the remaining clusters. This job is important in order to achieve better overall results in our stock prediction.

3.4.1 Fitness function

Once, the overall hypergraph has been partitioned into k parts (clusters), the cluster fitness criterion is been used in order to eliminate bad clusters. The fitness function measures the ratio of weights of edges that are within the partition and weights of edges involving any vertex of this partition. Let e be a set of vertices representing a hyperedge and C be a set of vertices representing a partition. Then the fitness function that measures how good the partition C is defined as follow:

$$fitness(C) = \frac{\sum_{e \subseteq C} Weight(e)}{\sum_{|e \cap C|>0} Weight(e)}$$

High fitness value suggests that the partition has more weights fot the edges connecting vertices within the partition. The partitions with fitness measure greater than a given threshold value are consider to be good clusters and will be used for stock prediction.

3.4.2 Connectivity measure function

Once good partitions are found, using the fitness measure algorithm described above, each good partition is examined to filter out vertices that are not highly connected to the rest of the vertices of the partition. The connectivity measures the percentage of edges that each vertex is associated with. The connectivity function of vertex u in C is defined as follow:

$$connectivity(u, C) = \frac{|\{e|e \subseteq C, u \in e\}|}{|\{e|e \subseteq C\}|}$$

High connectivity value suggests that the vertex has many edges connecting good proportion of the vertices in the partition. The vertices with connectivity measure greater than a give threshold value are considered to belong to the partition, and the remaining vertices are dropped from the partition.

3.5 Prediction

As mentioned before in 1.3 prediction of stock's price trend only based on historical data may be difficult due to stock's inherent indeterminacy. However, observing the closely related stocks in the same cluster which determined by their previous synchronicity we can analyze a stock. In order to do that, as proposed in [18, 22], after the hypergraph have been partitioned in k clustersfirstly we must find an overall parameter of rise or fall for every cluster. Let, S be the set of vertices in a cluster, and v a vertex in S. Then, we can calculate the overall parameter $f(S)$ as follows:

$$f(S) = \sum_{v \in S} T(v)$$

Where, $T(v)$ stands for the average of the gains of a stock in m days. $T(v)$ will be positive if stock rises in m days, or negative otherwise.

Then the $T'(u)$, which stands for inter-stocks global movement for the past m days, must be calculated. $T'(u)$ calculated as follows:

$$T'(u) = \sum_{i=1}^{m} \frac{i * T_i(u)}{\sum_{j=1}^{m} j}$$

Where, $T_i(u)$ means how much gains the stock u, i days ago. $T_i(u)$ will be positive if

stock rises, or negative otherwise.

Now having the $f(s)$ and the $T'(u)$ parameters we can run our prediction according to $R(u)$. $R(u)$ reflects if the stock u will rise the day after the last of our m days used in $f(s)$, or if it falls by judging if it is positive or negative. $R(u)$ calculated as follows:

$$R(u) = 0.7 * T'(u) + 0.3 * f(S)$$

3.5.1 Prediction Algorithm

All the above logic for stock prediction will summarise in the following algorithm.

Algorithm 3 Prediction Algorithm

```
1: procedure PREDICT
   Input: Transactions list, clusters
   Output: Predictions
2:     L = {Clusters};
3:     P = {};
4:     for all clusters S ∈ L do
5:         f(S) = CalculateCluserGlobalVariable;
6:         for all stocks u ∈ C do
7:             T'(u) = CalculateStock;
8:             R(u) = 0.7 * T'(u) + 0.3 * f(S);
9:             if R(u) > 0 then
10:                P.append(u, RISE);
11:            else
12:                P.append(u, FALL);
13:    Answer = P
```

```
1: f(S) = 0;
2: for all stocks v ∈ S do
3:     T(v) = sum(daily_movements(v))
4:     f(s)+ = T(v)
```

```
1: T'(u) = 0;
2: K = sum([j in range(1, m]))
3: for i in range(1,m) do //m = prior known days
4:     T'(u)+ = i * T(u)/K
```

Chapter 4

Experimental Results

In this chapter, the experimental results of this thesis will be presented. In Section 4.1 the test environment for our experiments will be detailed. After that, in Sections 4.2, 4.3, 4.4 the results of each partition algorithm will represented. In Appendix B you can find the full list of our results which produced from the most promising technique.

4.1 Test Environment

4.1.1 Instances

For our experiments we used two data sets from S&P 500 from two different time frames. The first dataset consists of stock data from 1st March 2016 to 31st March 2018, while the second one consists of stock data from 1st September 2015 to 30st September 2017. For both time frames, we predict the stock movements of 10th May 2018 based on their movements for 3, 7, and 21 days ago.

4.1.2 System

All the experiments were done in a MacBook Pro laptop running macOS 10.13 (High Sierra), which contains an Intel Core i5 2,6 GHz processor with two cores. Further-

more, the system has two L2-Caches, one per core, of size 256KB, L3-Cache of size 3MB and the main memory consists of two 1600MHz DDR3 chips with total size 8GB. For the three partition algorithms, we are using the official packages provided by their authors, which are available free of charge for non-commercial and research use.

4.1.3 Methology

Firtly, a bash script which also running a simple python script is been used for extracting the data from S&P 500. After this step, the Apriori algoithm is used for creating the two hypergraphs, one for each time period. We set the minimum support value of Apriori algorithm to 0.04 which means that all stocks in a frequent item set must have moved together at least on 21 days. After creating our hypergraphs, for each hypergraph we run the partioning algorithms with their default settings (and set the desired partitions to 40) to find our stock clusters and for each result we run additionally the fitness and connectivity criterions. Once we have all our results, we run our prediction algorithm to predict stock movements and evaluate each partioning technique.

4.2 hMetis results

During our experiments we found that hMetis make the partitions of our hypergraphs in 0 minutes and 21 seconds and 3 minutes and 47 seconds respectively. During the hypergraph creation we found that our hypergraphs consists of 352 verticies and 16118 hyperedges for the first one (starting from March 2016), and 480 verticies and 210709 hyperedges for the second one. Note that the number of verticies in both hypergraphs is considerably smaller than the original distinct items in our data sets. This is because some stocks do not move very frequently, hence the corresponding items do not have sufficient support.

After running hMetis for both hypergraphs, 40 clusters for each hypergraph was

generated. Out of this partitions, in both cases after running the fitness measure function and the connectivity function, only 21 partitions was remaining and the number of verticies was reduced again. Some of the results are shown in Table 4.1 and the complete list of the best scenario found by our experiments is available in Appendix B. In Table 4.2 and 4.3 some of our prediction results are shown for both hypergraphs. Finally, Fig. 4-1 shows the report of the prediction accuracy for both hypergraphs and for all available clusters.

Cluster ID	Cluster Items	Movement
1	LYB↓ ,KMX↓ ,NWSA↓ ,NWS↓ ,FLR↓	DOWN
2	**FLR↓ ,NEM↓ ,PNR↓ ,FLS↓ ,FOX↓ , FOXA↓ ,COTY↓ ,ALGN↓ ,XLNX↓**	**UP**
3	UNM↓ ,LNC↓ ,MET↓ ,AMP↓ ,PFG↓ , PBCT↓ ,PRU↓	UP
4	STX↓ ,AAL↓ ,UAL↓ ,ALK↓ ,LUV↓ , DAL↓	UP

(a) Original clusters produced by hMetis

Cluster ID	Cluster Items	Movement
1	**LYB↓ ,KMX↓** ,NWSA↓ ,NWS↓ ,FLR↓	**DOWN**
3	UNM↓ ,LNC↓ ,MET↓ ,AMP↓ ,PFG↓ , PBCT↓ ,PRU↓	UP
4	STX↓ ,AAL↓ ,UAL↓ ,ALK↓ ,LUV↓ , DAL↓	UP

(b) Clusters satisfying the fitness function (Cluster 2 is been removed)

Cluster ID	Cluster Items	Movement
1	NWSA↓ ,NWS↓ ,FLR↓	**DOWN**
3	UNM↓ ,LNC↓ ,MET↓ ,AMP↓ ,PFG↓ , PBCT↓ ,PRU↓	UP
4	STX↓ ,AAL↓ ,UAL↓ ,ALK↓ ,LUV↓ , DAL↓	UP

(c) Clusters after applying both fitness and connectivity functions (Verticies LYB, NEM was removed based on connectivity function)

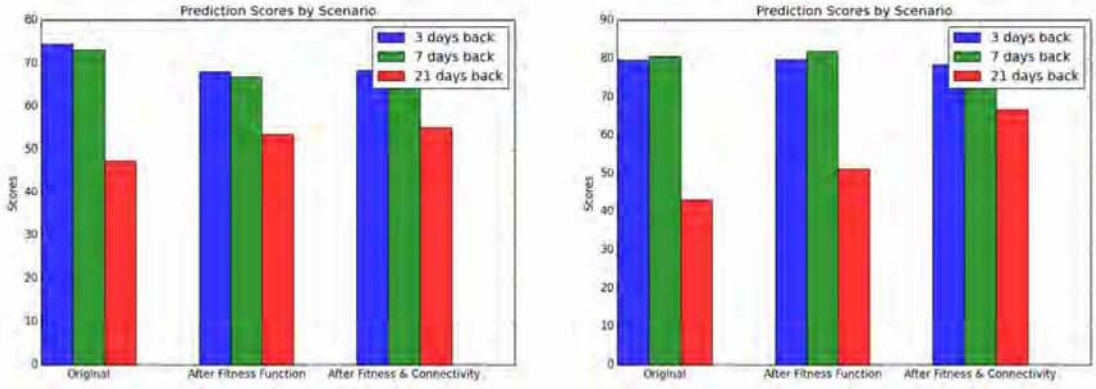
Table 4.1: Sample of clusters produced from hMetis partioning our first hypergraph

Stock	Actually	Prediction	Result
GOOGL	Rise	Rise	True
GOOG	Rise	Rise	True
OXY	Rise	Rise	True
ROK	Rise	Rise	True
NWS	Rise	Rise	True
NWSA	Rise	Rise	True
FLR	Rise	Rise	True
NEM	Rise	Rise	True
PNR	Fall	Rise	False
FLS	Rise	Rise	True

Table 4.2: Sample of prediction results for 10th May 2018 using the original clusters produced by hMetis on our first hypergraph and using real for the previous 3 days.

Stock	Actually	Prediction	Result
GOOGL	Rise	Rise	True
GOOG	Rise	Rise	True
OXY	Rise	Rise	True
ROK	Rise	Rise	True
NWS	Rise	Rise	True
NWSA	Rise	Rise	True
FLR	Rise	Rise	True
NEM	Rise	Fall	False
PNR	Fall	Rise	False
FLS	Rise	Rise	True

Table 4.3: Prediction results for 10th May 2018 using the original clusters produced by hMetis on our second hypergraph and using real for the previous 3 days.



(a) 1a (b) 1b

Figure 4-1: hMetis report of the prediction accuracy for both hypergraphs and for all available clusters. (1a) shows the prediction scores for various scenarios over our first hypergraph, while the (1b) is using the second hypergraph

4.3 PaToH results

During our experiments we found that PaToH make the partitions of our hypergraphs in 0 minutes and 9 seconds and 1 minute and 15 seconds respectively. During the hypergraph creation we found that our hypergraphs consists of 352 verticies and 16118 hyperedges for the first one (starting from March 2016), and 480 verticies and 210709 hyperedges for the second one. Note that the number of verticies in both hypergraphs is considerably smaller than the original distinct items in our data sets. This is because some stocks do not move very frequently, hence the corresponding items do not have sufficient support.

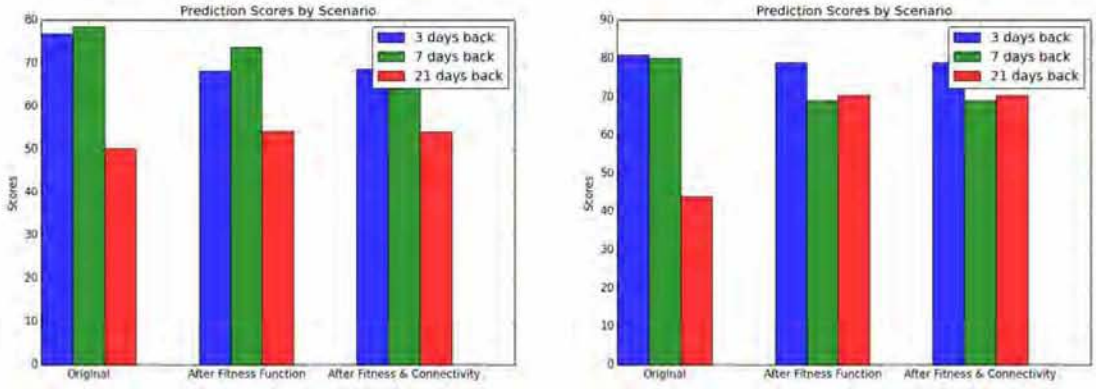
After running PaToH for both hypergraphs, 40 clusters for each hypergraph was generated. Out of this partitions, in both cases after running the fitness measure function and the connectivity function, only 18 partitions was remaining for the first hypergraph and only 6 for the second one. The number of verticies was reduced again. Some of the results are shown in Table 4.4 and the complete list of the best scenario found by our experiments is available in Appendix B. In Table 4.5 and 4.6 some of our prediction results are shown for both hypergraphs. Finally, Fig. 4-2 shows the report of the prediction accuracy for both hypergraphs and for all available clusters.

Stock	Actually	Prediction	Result
GOOGL	Rise	Rise	True
GOOG	Rise	Rise	True
OXY	Rise	Rise	True
ROK	Rise	Rise	True
NWS	Rise	Rise	True
NWSA	Rise	Rise	True
FLR	Rise	Rise	True
NEM	Rise	Rise	True
PNR	Fall	Rise	False
FLS	Rise	Rise	True

Table 4.5: Prediction results for 10th May 2018 using the original clusters produced by PaToH on our first hypergraph and using real for the previous 7 days.

Stock	Actually	Prediction	Result
GOOGL	Rise	Rise	True
GOOG	Rise	Rise	True
OXY	Rise	Rise	True
ROK	Rise	Rise	True
NWS	Rise	Rise	True
NWSA	Rise	Rise	True
FLR	Rise	Rise	True
NEM	Rise	Rise	True
PNR	Fall	Rise	False
FLS	Rise	Rise	True

Table 4.6: Prediction results for 10th May 2018 using the original clusters produced by PaToH on our second hypergraph and using real for the previous 7 days.



(a) 1a (b) 1b

Figure 4-2: PaToH report of the prediction accuracy for both hypergraphs and for all available clusters. (1a) shows the prediction scores for various scenarios over our first hypergraph, while the (1b) is using the second hypergraph

4.4 KaHyPar results

During our experiments we found that KaHyPar make the partitions of our hypergraphs in 0 minutes and 21 seconds and 5 minutes and 20 seconds respectively. During the hypergraph creation we found that our hypergraphs consists of 352 verticies and 16118 hyperedges for the first one (starting from March 2016), and 480 verticies and 210709 hyperedges for the second one. Note that the number of verticies in both hypergraphs is considerably smaller than the original distinct items in our data sets. This is because some stocks do not move very frequently, hence the corresponding items do not have sufficient support.

After running KaHyPar for both hypergraphs, 40 clusters for each hypergraph was generated. Out of this partitions, in both cases after running the fitness measure function and the connectivity function, only 22 partitions was remaining for the first hypergraph and only 8 for the second one. The number of verticies was reduced again. Some of the results are shown in Table 4.7 and the complete list of the best scenario found by our experiments is available in Appendix B. In Table 4.8 and 4.9 some of our prediction results are shown for both hypergraphs. As you can see in this tables some of stocks dont have any prediction. This heppens because after applying the

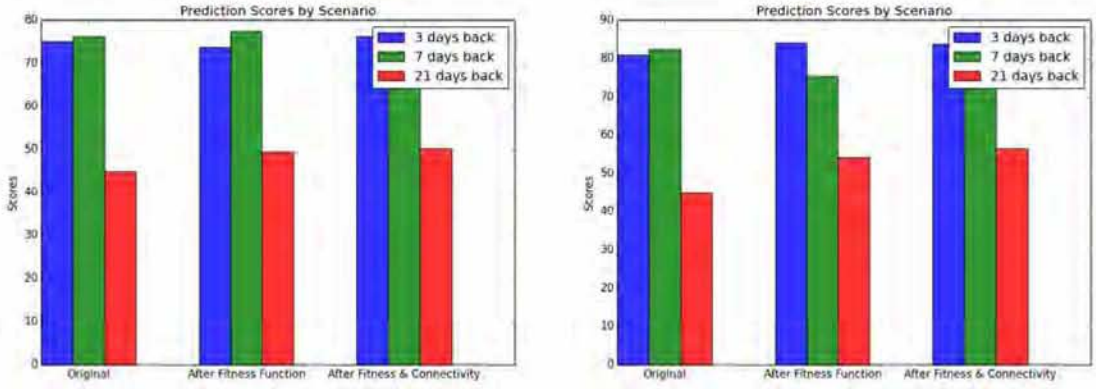
fitness and connectivity function the number of stocks remaining in clusters reduced significantly, so the number of stocks that we can predict their future movement reduced as well. Finally, Fig. 4-3 shows the report of the prediction accuracy for both hypergraphs and for all available clusters.

Stock	Actually	Prediction	Result
GOOGL	Rise	Rise	True
GOOG	Rise	Rise	True
OXY	Rise	Rise	True
ROK	Rise	Rise	True
NWS	Rise	Rise	True
NWSA	Rise	Rise	True
FLR	Rise	Rise	True
NEM	-	-	-
PNR	-	-	-
FLS	-	-	-

Table 4.8: Prediction results for 10th May 2018 using the clusters produced by KaHyPar after running the fitness and connectivity functions on our second hypergraph and using real for the previous 7 days.

Stock	Actually	Prediction	Result
GOOGL	Rise	Rise	True
GOOG	Rise	Rise	True
OXY	-	-	-
ROK	-	-	-
NWS	-	-	-
NWSA	-	-	-
FLR	-	-	-

Table 4.9: Prediction results for 10th May 2018 using the clusters produced by KaHyPar after running the fitness and connectivity functions on our second hypergraph and using real for the previous 7 days.

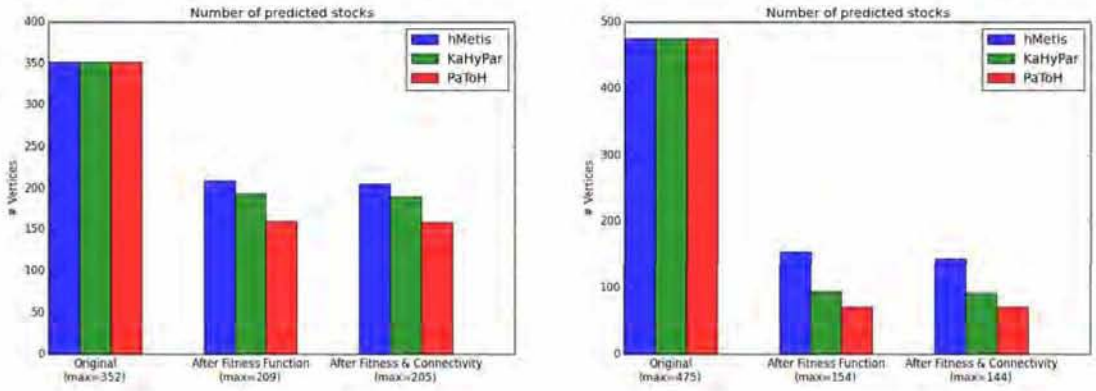


(a) 1a (b) 1b

Figure 4-3: KaHyPar report of the prediction accuracy for both hypergraphs and for all available clusters. (1a) shows the prediction scores for various scenarios over our first hypergraph, while the (1b) is using the second hypergraph

4.5 Comparison

After running all our experiments, we found that the partitions produced by the partioning tools, without applying neither fitness or connectivity functions can lead to better prediction results. Furthermore, as you can see in Fig. 4-4 the number of vertices-stocks that remain after applying this functions are considerable smaller than the original.



(a) 1a (b) 1b

Figure 4-4: Number of stocks that will be predicted in our scenarios. (1a) is referring to our first hypergraph, while (1b) is produced by our second hypergraph.

So considering all the above information, we are going to summarize and compare the three partioning tools used in this project, in the aspect of how well they can used to predict future stock movements for both hypergraphs and without applying any additional function in the result clusters.

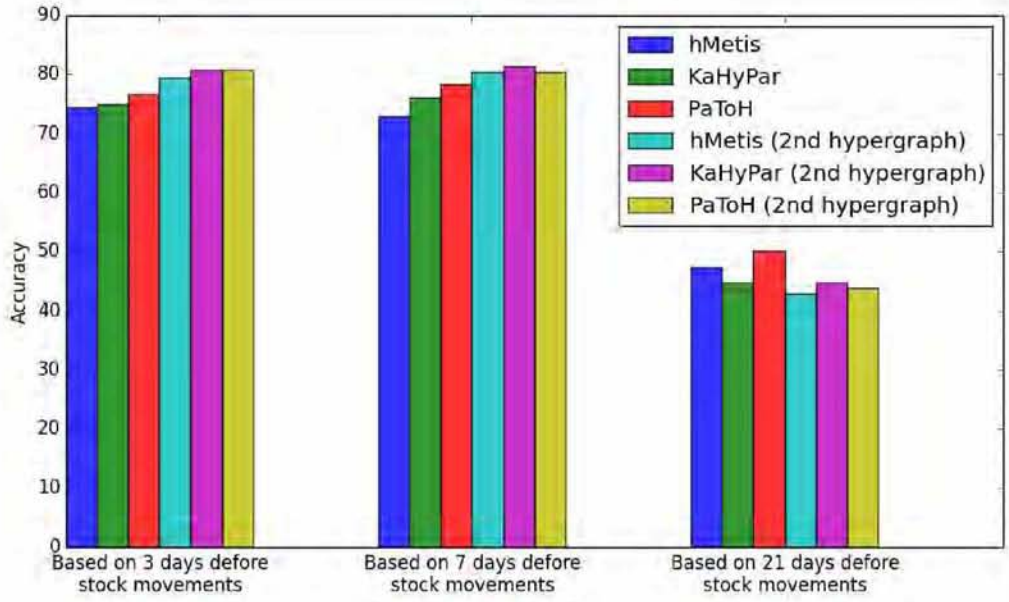


Figure 4-5: Comparison of all scenarios based on their prediction accuracy

As you can see in Fig. 4-5, the predictions produced using the stock movements for the past 21 days prior of the desired day, which is the number of days used in Apriori to create our hypergraphs, are considerable lower than the other two scenarios. For the other two scenarios, the accuracy for each partitioning tool is almost the same for either 3 days prior knowledge of stocks movement or 7 days. In our experiments, we saw that PaToH system has the leading accuracy comparing with the other two systems, and although the results of the second hypergraph are slightly better, the time you need to run Apriori is bigger enough to choose the second hypergraph as the best. Apriori took only 1 minutes and 7 seconds to build the first hypergraph, while for the second it took 2 hours, 44 minutes and 7 seconds.

Cluster ID	Cluster Items	Movement
1	**ALK↓ ,AAL↓ ,UAL↓ ,LUV↓ ,NCLH↓ , GPN↓ ,NFLX↓ ,DAL↓ ,RCL↓**	**DOWN**
2	AYI↓ ,PWR↓ ,ABC↓ ,MCK↓ ,O↓ , CAH↓ ,NWSA↓ ,NWS↓	DOWN
3	AAL↑ ,WDC↑ ,MU↑ ,UAL↑ ,ALK↑ , LUV↑ ,HPE↑ ,DAL↑ ,HPQ↑	UP
4	GPS↓ ,RL↓ ,JWN↓ ,LB↓ ,KSS↓ , HBI↓ ,M↓ ,BBY↓ ,FL↓	DOWN

(a) Original clusters produced by PaToH

Cluster ID	Cluster Items	Movement
2	**AYI↓** ,PWR↓ ,ABC↓ ,MCK↓ ,O↓ , CAH↓ ,NWSA↓ ,NWS↓	DOWN
3	AAL↑ ,WDC↑ ,MU↑ ,UAL↑ ,ALK↑ , LUV↑ ,HPE↑ ,DAL↑ ,HPQ↑	UP
4	GPS↓ ,RL↓ ,JWN↓ ,LB↓ ,KSS↓ , HBI↓ ,M↓ ,BBY↓ ,FL↓	DOWN

(b) Clusters satisfying the fitness function (Cluster 2 is been removed)

Cluster ID	Cluster Items	Movement
2	PWR↓ ,ABC↓ ,MCK↓ ,O↓ , CAH↓ ,NWSA↓ ,NWS↓	DOWN
3	AAL↑ ,WDC↑ ,MU↑ ,UAL↑ ,ALK↑ , LUV↑ ,HPE↑ ,DAL↑ ,HPQ↑	UP
4	GPS↓ ,RL↓ ,JWN↓ ,LB↓ ,KSS↓ , HBI↓ ,M↓ ,BBY↓ ,FL↓	DOWN

(c) Clusters after applying both fitness and connectivity functions (Verticies LYB, NEM was removed based on connectivity function)

Table 4.4: Sample of clusters produced from PaToH partioning our first hypergraph

Cluster ID	Cluster Items	Movement
1	**MPC↓ ,VLO↓ ,NEM↓ ,F↓ ,FL↓ , HOG↓ ,ALB↓ ,ALGN↓ ,NWL↓**	**DOWN**
2	MTB↑ ,GOOGL↓ ,GOOG↓ ,RHI↑ ,AYI↑ , ROK↑ ,ADBE↑ ,XLNX↑	UP
3	HBI↓ ,COTY↓ ,ABC↓ ,BBY↓ ,BHGE↓ , MCK↓ ,CMI↓ ,CNC↓ ,CAH↓	DOWN
4	LEN↑ ,DHI↑ ,PHM↑ ,JEC↑ ,CTL↑ , PH↑ ,NWS↑ ,NWSA↑ ,ALGN↑	UP

(a) Original clusters produced by KaHyPar

Cluster ID	Cluster Items	Movement
2	MTB↑ ,GOOGL↓ ,GOOG↓ ,RHI↑ ,AYI↑ , ROK↑ ,ADBE↑ ,XLNX↑	UP
3	**HBI↓ ,COTY↓ ,**ABC↓ ,BBY↓ ,BHGE↓ , MCK↓ ,CMI↓ ,CNC↓ ,CAH↓	**DOWN**
4	LEN↑ ,DHI↑ ,PHM↑ ,JEC↑ ,CTL↑ , PH↑ ,NWS↑ ,NWSA↑ ,ALGN↑	UP

(b) Clusters satisfying the fitness function (Cluster 2 is been removed)

Cluster ID	Cluster Items	Movement
2	MTB↑ ,GOOGL↓ ,GOOG↓ ,RHI↑ ,AYI↑ , ROK↑ ,ADBE↑ ,XLNX↑	UP
3	ABC↓ ,BBY↓ ,BHGE↓ , MCK↓ ,CMI↓ , CNC↓ ,CAH↓	DOWN
4	LEN↑ ,DHI↑ ,PHM↑ ,JEC↑ ,CTL↑ , PH↑ ,NWS↑ ,NWSA↑ ,ALGN↑	UP

(c) Clusters after applying both fitness and connectivity functions (Verticies LYB, NEM was removed based on connectivity function)

Table 4.7: Sample of clusters produced from KaHyPar partioning our first hypergraph

Chapter 5

Conclusion

Determine the future movements of stocks is a rather difficult process, and in many cases you need to have a lot of info. But as we shown, you can predict future stock movements of several stocks at a good rate, only having the knowledge of historical stock data.

Using the Apriori algorithm, we can construct a good hypergraph of the stock market, but as the dataset become larger or the data become more closely related the time of execution increased rapidly. After our experiments, we find that all the three partiotion tools have approximately the same rate, but PaToH has the leading both execution time and to produce more accurate results.

5.1 Future Work

As a future step at the end of the project, further research needs to be done to optimize the number of partitions produced by the partitiong tools and find the optimal number of days that stocks move together. Also new association rules algorithms must be developed in order to find related item-sets in less time and to include additional information of the stocks eg. the current market needs.

Appendix A

Algorithms

In this section, the pseudo-code of two algoritms (FM, and Kernighan-Lin) used in multilevel-partition algorithm will be presenting.

A.1 FM Algorithm

Algorithm 4 Fiduccia-Matheyses

1: **procedure** FM
Input: Hypergraph H = (V,E,w), ε and a bipartition $P = (V_1, V_2)$ with $V_i \leq \lceil \frac{c(V)}{2} \rceil (1+\varepsilon) i \in 0, 1)$
Output: Improved partition $P' = (V_1', V_2')$
2: $b_0 \leftarrow$ initialize with gain values from V_1, if we move those hypernodes from V_1 to V_2;
3: $b_1 \leftarrow$ initialize with gain values from V_2, if we move those hypernodes from V_2 to V_1;
4: $V_1' \leftarrow V_1$;
5: $V_2' \leftarrow V_2$;
6: **repeat**
7: $X_1 \leftarrow V_1'$;
8: $X_2 \leftarrow V_2'$;
9: **for** $i = 1$ until $|V| \vee (b_o.empty() \wedge b_1.empty())$ **do**
10: **repeat**
11: $p_i \leftarrow argmax_{j \in \{0,1\}} b_{j,max}$;
12: $g_i \leftarrow b_{p_i,max}$;
13: $v_i \leftarrow b_{p_i,max}.node()$;
14: $b_{p_i}.remove(v_i)$;
15: **until** $c(V_p) + c(u_i) \geq \lceil \frac{c(V)}{2} \rceil (1+\varepsilon)$
16: $X_{p_i} \leftarrow X_{p_i} \setminus \{v_i\}$;
17: $X_{1-p_i} \leftarrow X_{1-p_i} \cup \{v_i\}$;
18: $lock(v_i)$;
19: update gain of b_0 and b_1;
20: $k \leftarrow argmax_{1 \leqslant k \leqslant |V|} \sum_{i=1}^{k} g_i$;
21: $g \leftarrow \sum_{i=1}^{k} g_i$;
22: **if** $g > 0$ **then**
23: **for** $i = 1$ until k **do**
24: $V_{p_i}' \leftarrow V_{p_i}' \setminus \{v_i\}$;
25: $V_{1-p_i}' \leftarrow V_{1-p_i}' \setminus \{v_i\}$;
26: **until** g > 0
27: Return $= P' = (A', B')$;

A.2 Kernighan-Lin Algorithm

Algorithm 5 Kernighan-Lin

1: **procedure** KL
Input: Hypergraph H = (V,E,w) and perfect balanced partition P =(A, B)
Output: Improved partition $P' = (A', B')$
2: initialize all D values;
3: $A' \leftarrow A$;
4: $B' \leftarrow B$;
5: **repeat**
6: $X \leftarrow V'_1$;
7: $Y \leftarrow V'_2$;
8: **for** $i = 1$ until $\frac{|V|}{2}$ **do**
9: $(a^i, b^i) \leftarrow argmax_{a^i \in X, b^i \in Y} g(a^i, b^i)$
10: $g_i \leftarrow b_{p_i,max}$
11: $v_i \leftarrow b_{p_i,max}.node()$
12: $b_{p_i}.remove(v_i)$
13: $X_{p_i} \leftarrow X_{p_i} \setminus \{v_i\}$
14: $X_{1-p_i} \leftarrow X_{1-p_i} \cup \{v_i\}$
15: $lock(v_i)$
16: update gain of b_0 and b_1
17: $k \leftarrow argmax_{1 \leqslant k \leqslant |V|} \sum_{i=1}^{k} g_i$
18: $g \leftarrow \sum_{i=1}^{k} g_i$
19: **if** $g > 0$ **then**
20: **for** $i = 1$ until k **do**
21: $V'_{p_i} \leftarrow V'_{p_i} \setminus \{v_i\}$
22: $V'_{1-p_i} \leftarrow V'_{1-p_i} \setminus \{v_i\}$
23: **until** g > 0
24: Return = $P' = (A', B')$

Appendix B

Tables

In this section the complete results of our best scenario will be presenting. After our experiments the best scenario produced by running the PaToH partioning tool over our first hypergraph and make our predictions based on stock movements for 7 days prior the 10th May 2018. Also, neither fitness function neither connectivity function applied in the result clusters.

Cluster ID	Cluster Items (stocks)
0	QRVO↑, STX↑, AMD↑, COG↑, ARNC↑, LYB↑, CF↑, INCY↑, FLS↑
1	MPC↑, KSS↑, UAA↑, UA↑, KMI↑, M↑, GPS↑, JWN↑, SIG↑
2	URI↑, WMB↑, OKE↑, MOS↑, EQT↑, PWR↑, FTI↑, NRG↑, SLB↑
3	FCX↑, HAL↑, APA↑, EOG↑, HP↑, XEC↑, NOV↑, NFX↑, PXD↑
4	CHK↑, HES↑, MRO↑, CXO↑, DVN↑, RRC↑, APC↑, NBL↑, COP↑
5	LRCX↑, AMAT↑, NVDA↑, SWKS↑, NTAP↑, ADSK↑, AVGO↑, MCHP↑, KLAC↑
6	AAP↑, LB↑, BBY↑, FL↑, ORLY↑, COTY↑, AZO↑, CMG↑, XLNX↑
7	FLR↑, PNR↑, VLO↑, TGT↑, XRX↑, JEC↑, CTL↑, ROK↑, ALGN↑
8	ANDV↑, CNC↑, HBI↑, WRK↑, PCAR↑, OXY↑, AYI↑, PH↑, HOG↑
9	DOV↑, NEM↑, KORS↑, CAT↑, HST↑, CMI↑, SYMC↑, AES↑
10	ILMN↑, REGN↑, ALXN↑, VRTX↑, NFLX↑, TRIP↑, MYL↑, BIIB↑, CELG↑
11	WYNN↑, MGM↑, NUE↑, MLM↑, VMC↑, FOX↑, FOXA↑, FOX↓, FOXA↓
12	BWA↑, RCL↑, NCLH↑, LEN↑, ATVI↑, DHI↑, PHM↑, RHI↑, EA↑
13	AAL↑, WDC↑, MU↑, UAL↑, ALK↑, LUV↑, HPE↑, DAL↑, HPQ↑
14	ALB↑, FMC↑, VIAB↑, DISCK↑, DISCA↑, PYPL↑, ADI↑, ADBE↑
15	AMG↑, KMX↑, STT↑, GS↑, AMP↑, ETFC↑, SYF↑, BEN↑, BK↑
16	NTRS↑, BBT↑, PNC↑, MTB↑, WFC↑, GOOG↓, NSC↑, BLK↑, NWSA↑
17	ZION↑, NAVI↑, RJF↑, IVZ↑, MET↑, PFG↑, LUK↑, GOOGL↓, NWS↑
18	LNC↑, BAC↑, FITB↑, RF↑, CMA↑, HBAN↑, CFG↑, KEY↑, STI↑
19	COF↑, CSX↑, C↑, PRU↑, SCHW↑, UNM↑, JPM↑, MS↑
20	ADSK↓, NAVI↓, MU↓, VIAB↓, VRTX↓, WDC↓, URI↓, DISCA↓, DISCK↓
21	UA↓, SIG↓, UAA↓, COG↓, EQT↓, KMI↓, ANDV↓, NRG↓, OKE↓
22	CF↓, ARNC↓, AMD↓, WMB↓, MOS↓, FLS↓, NOV↓, FTI↓, TRIP↓
23	XEC↓, NFX↓, CHK↓, DVN↓, APA↓, HES↓, APC↓, MRO↓, PXD↓
24	NBL↓, RRC↓, HP↓, FCX↓, CXO↓, EOG↓, HAL↓, COP↓
25	GPS↓, RL↓, JWN↓, LB↓, KSS↓, HBI↓, M↓, BBY↓, FL↓
26	DISH↓, CTL↓, CMG↓, HRB↓, WYNN↓, ALB↓, JNPR↓, TSCO↓, KORS↓
27	ILMN↓, HPQ↓, NEM↓, STX↓, F↓, FMC↓, CMI↓, ALGN↓, FLR↓
28	SLB↓, NUE↓, NTAP↓, PNR↓, LYB↓, AMP↓, HOG↓, KMX↓, ADBE↓
29	MPC↓, MAT↓, AKAM↓, COTY↓, CAT↓, BHGE↓, CNC↓, NWL↓
30	MAC↓, GGP↓, HCP↓, KIM↓, REG↓, FRT↓, SPG↓, VTR↓, HCN↓
31	VLO↓, AYI↓, PWR↓, ABC↓, MCK↓, O↓, CAH↓, NWSA↓, NWS↓
32	AMG↓, MGM↓, BEN↓, BWA↓, VMC↓, MLM↓, ADS↓, IVZ↓, GM↓
33	INCY↓, MYL↓, PRGO↓, ALXN↓, REGN↓, SYF↓, AGN↓, BIIB↓, CELG↓
34	ATVI↓, WRK↓, IP↓, KLAC↓, EA↓, AAP↓, IDXX↓, ORLY↓
35	AVGO↓, NVDA↓, LRCX↓, SWKS↓, QRVO↓, AMAT↓, ADI↓, XLNX↓, MCHP↓
36	ALK↓, AAL↓, UAL↓, LUV↓, NCLH↓, GPN↓, NFLX↓, DAL↓, RCL↓
37	SCHW↓, RJF↓, UNM↓, LNC↓, MET↓, PFG↓, LUK↓, PRU↓, TXN↓
38	RF↓, ZION↓, FITB↓, PNC↓, CFG↓, HBAN↓, CMA↓, KEY↓, STI↓
39	BAC↓, MS↓, COF↓, ETFC↓, MTB↓, C↓, PBCT↓, GS↓

Table B.1: Complete clustering of S&P 500 stock data, using Apriori and PaToH partioning tool

Stock Name	Prediction	Actual	Result	Stock Name	Prediction	Actual	Result	Stock Name	Prediction	Actual	Result
QRVO	Rise	Rise	True	STX	Rise	Rise	True	AMD	Rise	Rise	True
COG	Rise	Fall	False	ARNC	Rise	Rise	True	LYB	Rise	Rise	True
CF	Rise	Rise	True	INCY	Rise	Rise	True	FLS	Rise	Rise	True
MPC	Rise	Rise	True	KSS	Fall	Fall	True	UAA	Fall	Rise	False
UA	Fall	Rise	False	KMI	Fall	Rise	False	M	Fall	Fall	True
GPS	Fall	Fall	True	JWN	Fall	Fall	True	SIG	Fall	Rise	False
URI	Rise	Rise	True	WMB	Rise	Rise	True	OKE	Rise	Rise	True
MOS	Rise	Rise	True	EQT	Rise	Rise	True	PWR	Rise	Rise	True
FTI	Rise	Fall	False	NRG	Rise	Rise	True	SLB	Rise	Rise	True
FCX	Rise	Rise	True	HAL	Rise	Rise	True	APA	Rise	Rise	True
EOG	Rise	Rise	True	HP	Rise	Rise	True	XEC	Rise	Rise	True
NOV	Rise	Rise	True	NFX	Rise	Fall	False	PXD	Rise	Rise	True
CHK	Rise	Rise	True	HES	Rise	Rise	True	MRO	Rise	Fall	False
CXO	Rise	Fall	False	DVN	Rise	Rise	True	RRC	Rise	Fall	False
APC	Rise	Rise	True	NBL	Rise	Rise	True	COP	Rise	Rise	True
LRCX	Rise	Rise	True	AMAT	Rise	Rise	True	NVDA	Rise	Rise	True
SWKS	Rise	Rise	True	NTAP	Rise	Rise	True	ADSK	Rise	Rise	True
AVGO	Rise	Rise	True	MCHP	Rise	Rise	True	KLAC	Rise	Rise	True
AAP	Rise	Rise	True	LB	Rise	Fall	False	BBY	Rise	Rise	True
FL	Rise	Rise	True	ORLY	Rise	Fall	False	COTY	Rise	Fall	False
AZO	Rise	Rise	True	CMG	Rise	Fall	False	XLNX	Rise	Rise	True
FLR	Fall	Rise	False	PNR	Fall	Fall	True	VLO	Fall	Fall	True
TGT	Fall	Rise	False	XRX	Fall	Rise	False	JEC	Fall	Rise	False
CTL	Fall	Rise	False	ROK	Rise	Rise	True	ALGN	Rise	Rise	True
ANDV	Rise	Rise	True	CNC	Rise	Rise	True	HBI	Rise	Rise	True
WRK	Rise	Rise	True	PCAR	Rise	Fall	False	OXY	Rise	Rise	True
AYI	Rise	Fall	False	PH	Rise	Fall	False	HOG	Rise	Rise	True
DOV	Rise	Fall	False	NEM	Rise	Rise	True	KORS	Rise	Rise	True
CAT	Rise	Rise	True	HST	Rise	Rise	True	CMI	Rise	Fall	False
SYMC	Rise	Rise	True	AES	Rise	Rise	True	ILMN	Rise	Rise	True
REGN	Rise	Rise	True	ALXN	Rise	Fall	False	VRTX	Rise	Rise	True
NFLX	Rise	Fall	False	TRIP	Rise	Rise	True	MYL	Rise	Rise	True
BIIB	Rise	Fall	False	CELG	Rise	Fall	False	WYNN	Rise	Rise	True
MGM	Rise	Rise	True	NUE	Rise	Rise	True	MLM	Rise	Fall	False
VMC	Rise	Fall	False	FOX	Rise	Rise	True	FOXA	Rise	Rise	True
FOX	Rise	Rise	True	FOXA	Rise	Rise	True	BWA	Rise	Rise	True
RCL	Rise	Rise	True	NCLH	Rise	Rise	True	LEN	Rise	Rise	True
ATVI	Rise	Rise	True	DHI	Rise	Rise	True	PHM	Rise	Rise	True
RHI	Rise	Fall	False	EA	Rise	Rise	True	AAL	Fall	Rise	False
WDC	Rise	Rise	True	MU	Rise	Rise	True	UAL	Rise	Rise	True

Table B.2: Complete of predictions of S&P 500, using Apriori and PaToH partioning tool, for 10th May 2018 (part 1)

Stock Name	Prediction	Actual	Result	Stock Name	Prediction	Actual	Result	Stock Name	Prediction	Actual	Result
ALK	Rise	Rise	True	LUV	Rise	Rise	True	HPE	Rise	Rise	True
DAL	Rise	Rise	True	HPQ	Rise	Rise	True	ALB	Rise	Fall	False
FMC	Rise	Rise	True	VIAB	Rise	Rise	True	DISCK	Rise	Rise	True
DISCA	Rise	Rise	True	PYPL	Rise	Rise	True	ADI	Rise	Rise	True
ADBE	Rise	Rise	True	AMG	Rise	Rise	True	KMX	Rise	Fall	False
STT	Rise	Rise	True	GS	Rise	Rise	True	AMP	Rise	Rise	True
ETFC	Rise	Rise	True	SYF	Rise	Rise	True	BEN	Rise	Rise	True
BK	Rise	Rise	True	NTRS	Rise	Rise	True	BBT	Rise	Rise	True
PNC	Rise	Rise	True	MTB	Rise	Rise	True	WFC	Rise	Rise	True
GOOG	Rise	Rise	True	NSC	Rise	Rise	True	BLK	Rise	Rise	True
NWSA	Rise	Rise	True	ZION	Rise	Rise	True	NAVI	Rise	Rise	True
RJF	Rise	Rise	True	IVZ	Rise	Rise	True	MET	Rise	Rise	True
PFG	Rise	Rise	True	LUK	Rise	Rise	True	GOOGL	Rise	Rise	True
NWS	Rise	Rise	True	LNC	Rise	Rise	True	BAC	Rise	Rise	True
FITB	Rise	Rise	True	RF	Rise	Rise	True	CMA	Rise	Rise	True
HBAN	Rise	Rise	True	CFG	Rise	Rise	True	KEY	Rise	Rise	True
STI	Rise	Rise	True	COF	Rise	Rise	True	CSX	Rise	Rise	True
C	Rise	Rise	True	PRU	Rise	Rise	True	SCHW	Rise	Rise	True
UNM	Rise	Rise	True	JPM	Rise	Rise	True	MS	Rise	Rise	True
ADSK	Rise	Rise	True	NAVI	Rise	Rise	True	MU	Rise	Rise	True
VIAB	Rise	Rise	True	VRTX	Rise	Rise	True	WDC	Rise	Rise	True
URI	Rise	Rise	True	DISCA	Rise	Rise	True	DISCK	Rise	Rise	True
UA	Rise	Rise	True	SIG	Rise	Rise	True	UAA	Rise	Rise	True
COG	Rise	Fall	False	EQT	Rise	Rise	True	KMI	Rise	Rise	True
ANDV	Rise	Rise	True	NRG	Rise	Rise	True	OKE	Rise	Rise	True
CF	Rise	Rise	True	ARNC	Rise	Rise	True	AMD	Rise	Rise	True
WMB	Rise	Rise	True	MOS	Rise	Rise	True	FLS	Rise	Rise	True
NOV	Rise	Rise	True	FTI	Rise	Fall	False	TRIP	Rise	Rise	True
XEC	Rise	Rise	True	NFX	Rise	Fall	False	CHK	Rise	Rise	True
DVN	Rise	Rise	True	APA	Rise	Rise	True	HES	Rise	Rise	True
APC	Rise	Rise	True	MRO	Rise	Fall	False	PXD	Rise	Rise	True
NBL	Rise	Rise	True	RRC	Rise	Fall	False	HP	Rise	Rise	True
FCX	Rise	Rise	True	CXO	Rise	Fall	False	EOG	Rise	Rise	True
HAL	Rise	Rise	True	COP	Rise	Rise	True	GPS	Fall	Fall	True
RL	Rise	Fall	False	JWN	Rise	Fall	False	LB	Rise	Fall	False
KSS	Fall	Fall	True	HBI	Fall	Rise	False	M	Fall	Fall	True
BBY	Fall	Rise	False	FL	Fall	Rise	False	DISH	Rise	Rise	True
CTL	Rise	Rise	True	CMG	Rise	Fall	False	HRB	Rise	Rise	True
WYNN	Rise	Rise	True	ALB	Rise	Fall	False	JNPR	Rise	Rise	True
TSCO	Rise	Rise	True	KORS	Rise	Rise	True	ILMN	Rise	Rise	True

Table B.3: Complete of predictions of S&P 500, using Apriori and PaToH partioning tool, for 10th May 2018 (part 2)

Stock Name	Prediction	Actual	Result	Stock Name	Prediction	Actual	Result	Stock Name	Prediction	Actual	Result
HPQ	Rise	Rise	True	NEM	Rise	Rise	True	STX	Rise	Rise	True
F	Rise	Rise	True	FMC	Rise	Rise	True	CMI	Rise	Fall	False
ALGN	Rise	Rise	True	FLR	Rise	Rise	True	SLB	Rise	Rise	True
NUE	Rise	Rise	True	NTAP	Rise	Rise	True	PNR	Rise	Fall	False
LYB	Rise	Rise	True	AMP	Rise	Rise	True	HOG	Rise	Rise	True
KMX	Rise	Fall	False	ADBE	Rise	Rise	True	MPC	Rise	Rise	True
MAT	Rise	Rise	True	AKAM	Rise	Fall	False	COTY	Rise	Fall	False
CAT	Rise	Rise	True	BHGE	Rise	Fall	False	CNC	Rise	Rise	True
NWL	Rise	Rise	True	MAC	Rise	Rise	True	GGP	Rise	Rise	True
HCP	Rise	Rise	True	KIM	Rise	Rise	True	REG	Rise	Fall	False
FRT	Rise	Rise	True	SPG	Rise	Rise	True	VTR	Rise	Rise	True
VLO	Fall	Fall	True	AYI	Fall	Fall	True	PWR	Fall	Rise	False
ABC	Fall	Rise	False	MCK	Fall	Rise	False	O	Fall	Rise	False
CAH	Fall	Rise	False	NWSA	Fall	Rise	False	NWS	Fall	Rise	False
AMG	Rise	Rise	True	MGM	Rise	Rise	True	BEN	Rise	Rise	True
BWA	Rise	Rise	True	VMC	Rise	Fall	False	MLM	Rise	Fall	False
ADS	Rise	Rise	True	IVZ	Rise	Rise	True	GM	Rise	Rise	True
INCY	Fall	Rise	False	MYL	Fall	Rise	False	PRGO	Fall	Rise	False
ALXN	Fall	Fall	True	REGN	Fall	Rise	False	SYF	Fall	Rise	False
AGN	Fall	Rise	False	BIIB	Fall	Fall	True	CELG	Fall	Fall	True
ATVI	Rise	Rise	True	WRK	Rise	Rise	True	IP	Rise	Rise	True
KLAC	Rise	Rise	True	EA	Rise	Rise	True	AAP	Rise	Rise	True
IDXX	Rise	Rise	True	ORLY	Rise	Fall	False	AVGO	Rise	Rise	True
NVDA	Rise	Rise	True	LRCX	Rise	Rise	True	SWKS	Rise	Rise	True
QRVO	Rise	Rise	True	AMAT	Rise	Rise	True	ADI	Rise	Rise	True
XLNX	Rise	Rise	True	MCHP	Rise	Rise	True	ALK	Rise	Rise	True
AAL	Rise	Rise	True	UAL	Fall	Rise	False	LUV	Fall	Rise	False
NCLH	Fall	Rise	False	GPN	Fall	Rise	False	NFLX	Rise	Fall	False
DAL	Rise	Rise	True	RCL	Rise	Rise	True	SCHW	Rise	Rise	True
RJF	Rise	Rise	True	UNM	Rise	Rise	True	LNC	Rise	Rise	True
MET	Rise	Rise	True	PFG	Rise	Rise	True	LUK	Rise	Rise	True
PRU	Rise	Rise	True	TXN	Rise	Rise	True	RF	Rise	Rise	True
ZION	Rise	Rise	True	FITB	Rise	Rise	True	PNC	Rise	Rise	True
CFG	Rise	Rise	True	HBAN	Rise	Rise	True	CMA	Rise	Rise	True
KEY	Rise	Rise	True	STI	Rise	Rise	True	BAC	Rise	Rise	True
MS	Rise	Rise	True	COF	Rise	Rise	True	ETFC	Rise	Rise	True
MTB	Rise	Rise	True	C	Rise	Rise	True	PBCT	Rise	Rise	True
GS	Rise	Rise	True								

Table B.4: Complete of predictions of S&P 500, using Apriori and PaToH partioning tool, for 10th May 2018 (part 3)

Bibliography

[1] `https://www.standardandpoors.com/`.

[2] `http://pandas-datareader.readthedocs.io/en/latest/`.

[3] `https://www.python.org/`.

[4] `https://t4tutorials.com/apriori-algorithm-in-data-mining-with-examples`.

[5] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. "Mining association rules between sets of items in large databases." In *In Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington, D.C., May 1993.

[6] Rakesh Agrawal and Ramakrishnan Srikant. "Fast Algorithms for Mining Association Rules." In *In Proc. of the 20th Int'l Conference on Very Large Databases*, Santiago, Chile, September 1994.

[7] Umit V Catalyurek and Cevdet Aykanat. "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication." In *Parallel and Distributed Systems, IEEE Transactions*, pages 673–693, 1999.

[8] Umit V. Catalyurek and Cevdet Aykanat. "Patoh (partitioning tool for hypergraphs).", 2011.

[9] C. M. Fiduccia and R. M. Mattheyses. "A linear time heuristic for improving network partitions." In *In Proc. 19th IEEE Design Automation Conference*, pages 175–181, 1982.

[10] Eui-Hong (Sam) Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. "Clustering based on association rule hypergraphs (position paper)". In *In Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 9–13, Tucson, Arizona, 1997.

[11] Eui-Hong (Sam) Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. "Hypergraph Based Clustering in High-Dimensional Data Sets: A summary of Results". In *IEEE Bulletin of the Technical Committee on Data Engineering*, 1998.

[12] George Karypis and Vipin Kumar. "Multilevel k-way Hypergraph Partitioning." In *36th Design Automation Conference*, pages 343–348, 1990.

[13] George Karypis and Vipin Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs". In *SIAM Journal on Scientific Computing*, 1998.

[14] George Karypis and Vipin Kumar. "Multilevel algorithms for multi-constraint graph partitioning". In *Journal Parallel and Distributed Computing*, volume 48, 1998.

[15] George Karypis and Vipin Kumar. "Multilevel Hypergraph Partitioning: Applications in VLSI Domain." In *IEEE Transactions on VLSI Systems*, volume 7, pages 69–79, 1999.

[16] George Karypis, Vipin Kumar, Rajat Aggarwal, and Shashi Shekhar. "Multilevel Hypergraph Partitioning: Applications in VLSI Domain". In *34th Design and Automation Conference*, pages 526–529, 1997.

[17] B. W. Kernighan and S. Lin. " An efficient heuristic procedure for partitioning graphs." In *The Bell System Technical Journal*, pages 291–307, 1970.

[18] Yongen Luo, Jicheng Hu, Xiaofeng Wei, Dongjian Fang, and Heng Shao. "Stock Trends Prediction based on Hypergraph Modeling Clustering Algorithm)". In *IEEE International Conference on Progress in Informatics and Computing*, 2014.

[19] Vitaly Osipov and Peter Sanders. n-level graph partitioning. In *European Symposium on Algorithms*, pages 278–289. Springer, 2010.

[20] Laura A Sanchis. Multiple-way network partitioning. *IEEE Transactions on Computers*, (1):62–81, 1989.

[21] Sebastian Schlag, Vitali Henne, Tobias Heuer, Henning Meyerhenke, Peter Sanders, and Christian Schulz. "Basics of Hypergraph Theory". In *18th Workshop on Algorithm Engineering and Experiments, (ALENEX 2016)*, pages 53–67, 2016.

[22] Yang Shen, Jicheng Hu, Yanan Lu, and Xiaofeng Wang. "Stock Trends Prediction by Hypergraph Modeling". In *IEEE International Conference on Computer Science and Automation Engineering*, 2012.

[23] Hongliang Zhang, Lingyang Song, Zhu Han, and Yingjun Zhang. "k-way Hypergraph Partitioning via n-Level Recursive Bisection". In *Hypergraph Theory in Wireless Communication Networks, (Springer 2018)*, pages 1–10, 2018.

[24] Florian Ziegler. *n-Level Hypergraph Partitioning.* PhD thesis, Citeseer, 2012.