Using Unsupervised Learning for Data-Driven Procurement Demand Aggregation

Eran Shaham,[1] Adam Westerski,[2] Rajaraman Kanagasabai,[3] Amudha Narayanan,[4] Samuel Ong,[5] Jiayu Wong,[6] Manjeet Singh[7]

Institute for Infocomm Research, A*STAR, Singapore,[1,2,3,4] ComfortDelGro, Singapore,[5]
A*STAR Procurement Office, A*STAR, Singapore,[6,7]

eran-shaham@i2r.a-star.edu.sg, adam-westerski@i2r.a-star.edu.sg, kanagasa@i2r.a-star.edu.sg, naraa@i2r.a-star.edu.sg, samuelong@comfortdelgro.com, wong_jiayu@hq.a-star.edu.sg, manjeet_singh@hq.a-star.edu.sg

Abstract

Procurement is an essential operation of every organization regardless of its size or domain. As such, aggregating the demands could lead to better value-for-money due to: (1) lower bulk prices; (2) larger vendor tendering; (3) lower shipping and handling fees; and (4) reduced legal and administration overheads. This paper describes our experience in developing an AI solution for demand aggregation and deploying it in A*STAR, a large governmental research organization in Singapore with procurement expenditure to the scale of hundreds of millions of dollars annually. We formulate the demand aggregation problem using a bipartite graph model depicting the relationship between procured items and target vendors, and show that identifying maximal edge bicliques within that graph would reveal potential demand aggregation patterns. We propose an unsupervised learning methodology for efficiently mining such bicliques using a novel Monte Carlo subspace clustering approach. Based on this, a proof-of-concept prototype was developed and tested with the end users during 2017, and later trialed and iteratively refined, before being rolled out in 2019. The final performance was 71% of past cases transformed into bulk tenders correctly detected by the engine; for new opportunities pointed out by the engine 81% were deemed useful for potential bulk tender contracts in the future. Additionally, per each valid pattern identified, the engine achieved 100% precision (all aggregated purchase orders were correct), and 79% recall (the engine correctly identified 79% of orders that should have been put into the bulk tenders). Overall, the cost savings from the true positive contracts spotted so far are estimated to be S$7 million annually.

Introduction

Procurement is an essential operation of every organization regardless of its size, business domain, and sector (i.e., private or public). A typical procurement budget can be a significant portion of total expenditure, up to 60% of revenues (Bartolini 2011). Knowing its magnitude, companies increasingly invest in optimizing procurement operations by reorganizing their practices to root out any inefficiencies or problems. This paper describes our experience in developing an AI solution for demand aggregation and deploying it in A*STAR, a large governmental research organization in Singapore with procurement expenditure in the scale of hundreds of millions of dollars annually. We were presented with a set of digitalized procurement data and had to work with certain constraints, such as noisy, incomplete and unlabeled data scattered across multiple legacy data management systems. This is in contrast to academic literature which discusses different aspects of procurement management based on theoretical frameworks and strong assumptions on data scale, labeling and consistency.

Large enterprises like A*STAR spend millions on purchases of goods and services. Demand aggregation is the process of aggregating the demands for better value-for-money due to: (1) lower bulk prices; (2) larger vendor tendering; (3) lower shipping and handling fees; and (4) reduced legal and administration overheads.

Our key insight is in formulating the demand aggregation problem using a bipartite graph model which depicts the relationship between procured items and target vendors. The need for an AI solution is mainly due to the dataset size. While small datasets could settle for eyeballing, datasets of moderate size are already hard to handle. In addition, employing simple techniques (e.g., clustering) will not be adequate due to the "curse of dimensionality" (Bellman 1961; Beyer et al. 1999; Kriegel, Kröger, and Zimek 2009). Thus, for a solution to be successfully deployed on an unlabeled large dataset, and to successfully scale (the dataset might increase in size over time), unsupervised learning techniques

are required. We show that identifying maximal edge bicliques within a bipartite graph would reveal potential demand aggregation patterns. We propose an unsupervised learning methodology for efficiently mining such bicliques using a novel Monte Carlo subspace clustering approach.

We begin by describing the organizational context and the problem description.

Problem Description

A*STAR is a large governmental research organization in Singapore comprising over 17 research entities and 5000 staff. Procurement spending of the organization can run into hundreds of millions of dollars annually. Given their scale, the procurement operations are handled in a decentralized manner by individual entities through an online workflow comprising several steps (see Figure 1), and are governed and audited by a centralized unit called the A*STAR Procurement Office (A*PO) through a predominantly manual process.

A*PO approached us in 2014 to create a data-driven framework towards transforming the manual process of detecting potential lapses, enhancing procurement compliance, and optimizing procurement spend. We embarked on the A*STAR Procurement Analytics initiative to develop an AI platform for tackling three major challenges: (1) procurement fraud detection; (2) procurement demand forecasting; and (3) procurement demand aggregation. Our research on the former two problems is reported elsewhere (Westerski et.al. 2015; Westerski, Kangasabai, and Sim 2017).

This paper focuses on the procurement demand aggregation problem and reports on our experience in developing an AI solution for tackling the problem and deploying it A*STAR-wide. Broadly, this was done in two phases: (1) a proof-of-concept phase in 2017, where the core method was developed by back testing on historical data; and (2) a pilot trial and iterative refinement until final roll-out in 2019. A high-level description of the algorithm developed during the proof-of-concept phase has been filed as a patent (Shaham, Westerski, and Kangasabai 2019). In this paper, we provide a complete account of both the phases as part of sharing our experiences.

Demand aggregation is the process of aggregating the demands for goods and services to achieve better value-for-money in terms of: (1) lower bulk prices; (2) larger vendor tendering; (3) lower shipping and handling fees; and (4) reduced legal and administration overheads. Large enterprises such as A*STAR spend millions of dollars on purchases of goods and services. Thus, even a small change in these four fields could lead to substantial savings.

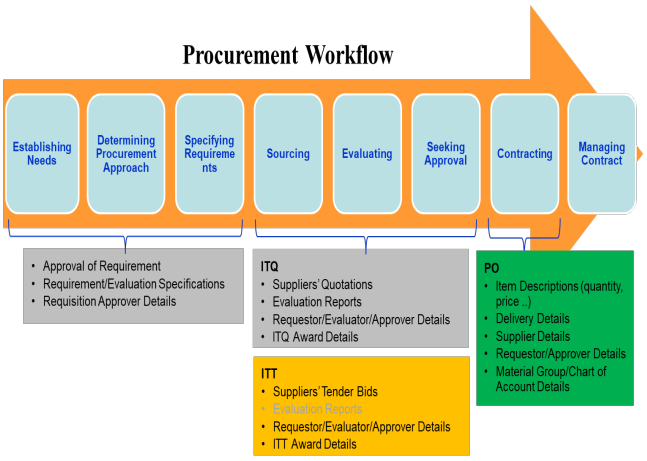

Figure 1. Procurement workflow and steps.

To fully appreciate the application, we first describe the procurement data structure.

Data Structure

The presented application was developed based on procurement transactional data produced over the course of 8 years (from 2009 to 2016, inclusive). The dataset was refreshed in subsequent stages of the project to include consecutive years. However, to present a consistent and focused evaluation scenario, including comparison of accuracy between different project stages, the experiments reported here relate to the initial 8 years only.

Within this dataset, the key elements that comprise a purchase are related to procedural stages of the procurement process as presented in Figure 1: (a) Invitation to Tender (ITT) or Invitation to Quotation (ITQ); (b) bid placement and approval of selected supplier; and (c) issuance of Purchase Order (PO). The data for typical final stages of procurement process related to delivery of goods and invoicing are generally treated as post-procurement process and are therefore not included in the discussion in this article. Also, as demand aggregation applies only to orders that were successfully placed, we will focus on the PO data.

Each PO would point to an employee of the organization fulfilling the role of buyer (*requesting officer*) and a hierarchy of *approving officers* (in charge of approving the purchase request). It would also have a *creation date* (day on which requesting officer submitted the information to the system) and an *approval date*. A PO would further consist of *purchase order items*, each of which can relate to a different item or service and contain further details, such as: textual *item description*, *quantity* of items bought and *unit price* per single item. Similar structure of purchases split into items with details of pricing, quantities, descriptions

etc. would also be present during earlier stages of (a) and (b). It is worth noting that values at all stages can differ – starting from expected purchases, to what suppliers offer, to what was finally negotiated and approved.

Throughout the experiments, we learned that the key features that most influenced the capabilities of our algorithm were the following five aspects of the procurement data: who (*requester*); what (*item description*); what type (*material group*); from who (*vendor*); and when (*creation/approval date*).

In total, the dataset comprised 1,032,275 POs with 660,162 distinct items, and 14,834 unique vendors. On average, a single PO had 2.2 items attached. However, 59% of the POs had only 1 order item attached, and 97% had 10 or less. Within the remaining 3%, the maximal recorded amount of order items per PO was 164. This reflects the overall behavior of the organization employees and the policies in place, which focused on simple orders typically related to one type of good.

Application Description

Aggregating and analyzing the demand using the PO dataset is not straightforward, due to the following challenges:

- Item description was a key field, but it was free text with no standard terminology. Also, it was of short length (<128 characters) and got truncated if it exceeded.
- Vendor name was another key field, which had standard names for vendors located in Singapore (the names followed the official ACRA-registered ones), but not for those located overseas.
- The only labeled data that we could possibly use were the 53 bulk contracts currently in operation. However, the contract descriptions had only a general description of the items covered, and the vendor names had poor matches with those in the dataset.
- The scale of the data, although not huge, was large enough to make many state-of-the-art unsupervised learning algorithms (as we later required) infeasible.

A naïve aggregation strategy could be implemented via clustering of PO items (and then by vendors) by suitably defining a text similarity measure (Chew 2017; Chowdhary et al. 2011; Wang and Miller 2005). However, such a one-dimensional clustering approach has two issues:

1. It is not adequate for large-scale datasets due to the "curse of dimensionality" (Bellman 1961; Beyer et al. 1999; Kriegel, Kröger, and Zimek 2009). Furthermore, demand aggregation requires an algorithm which can simultaneously group subsets of items which relate to subsets of vendors (see Figure 2).
2. On smaller or medium-sized data, its accuracy will be affected by the variability in text descriptions and effectively the similarity measure is fine tuned to tolerate it.

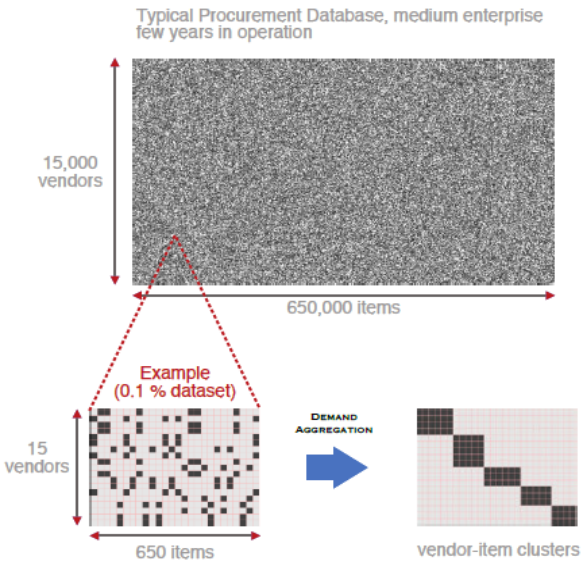

Figure 2. Simplified example of a procurement database containing hidden demand aggregation patterns (patterns can overlap).

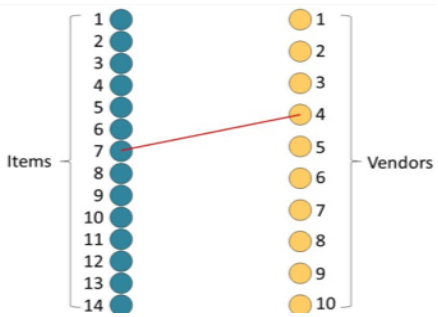

Figure 3. Procurement bipartite graph comprising 14 items, 10 vendors, and the relationship between item 7 and vendor 4 (item 7 was purchased from vendor 4).

The relationship between items and vendors could be naturally modeled by a bipartite graph (see example in Figure 3). Such graphs have been proven useful in modeling a wide range of relationship networks (Kunegis 2013; Shaham, Yu, and Li 2016). A simultaneous grouping of items and vendors within a bipartite graph is called a biclique (see examples in Figure 4). Note that a biclique does *not* require all the items (or vendors) to be lexically similar.

Biclique detection is a well-known problem in graph theory and data mining, with numerous real-world applications across different domains (Ben-Dor et al. 2003; Cheng and Church 2000; Dawande et al. 2001; Ganter and Wille 1999; Kunegis 2013; Melkman and Shaham 2004; Mishra, Ron, and Swaminathan 2003; Mushlin et al. 2007; Nussbaum et al. 2010; Sanderson et al. 2003; Swaminathan and Tayur 1998; Zhang et al. 2014). Given a bipartite graph and its cor-

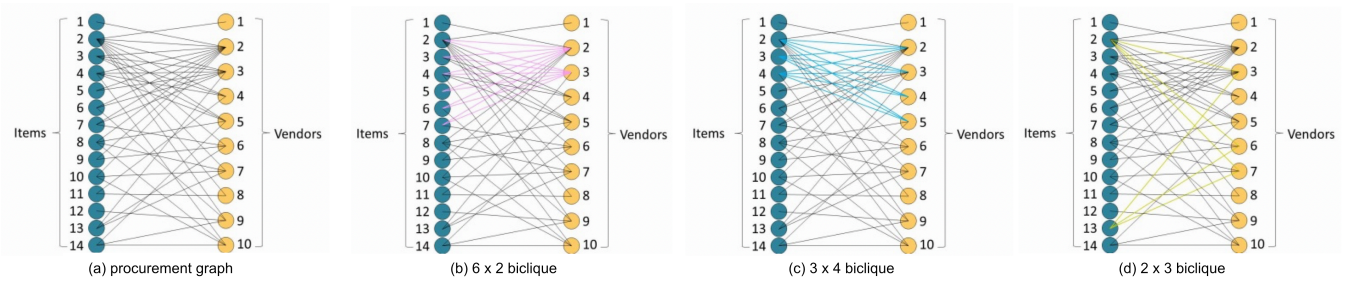

Figure 4. Example of (a) procurement bipartite graph comprising: (b) biclique of 6 items x 2 vendors ($\{i_2, i_3, i_4, i_5, i_6, i_7\} \times \{v_2, v_3\}$); (c) biclique of 3 items x 4 vendors ($\{i_2, i_3, i_4\} \times \{v_2, v_3, v_4, v_5\}$); and (d) biclique of 2 items x 3 vendors ($\{i_2, i_{13}\} \times \{v_3, v_6, v_7\}$).

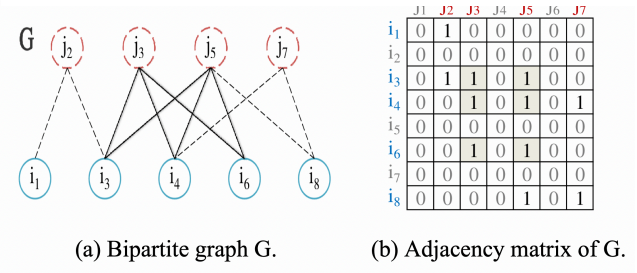

Figure 5. (a) Bipartite graph G; and (b) its corresponding adjacency matrix, comprising the maximum edge biclique ($\{i_3, i_4, i_6\}$, $\{j_3, j_5\}$) of size 6 edges and 5 vertices.

responding partition into two disjoint sets of vertices, a biclique is a complete bipartite subgraph such that every vertex of the first partition is connected to every vertex of the second partition (see example in Figure 5, where a vertex set $\{i_3, i_4, i_6\}$ and a vertex set $\{j_3, j_5\}$ form a biclique). Mathematically, the notion of biclique is defined as follows.

Definition 1. Let G = (U ∪ V, E) be a bipartite graph, where U and V are two disjoint sets of vertices, and E is an edge set such that ∀(i,j) ∈ E, i ∈ U, j ∈ V. A biclique within G is a couple (set pair) (I, J) such that I ⊆ U, J ⊆ V and ∀i ∈ I, j ∈ J, (i, j) ∈ E.

The computational complexity of finding the maximum biclique depends on the exact objective function used. In contrast to the well-known maximum clique problem (Makino and Uno 2004; Tomita, Tanaka, and Takahashi 2006), the maximum biclique problem has three distinct variants, with the following objective function μ(I, J):

1. μ(I, J) = |I| × |J| — known as the MAXIMUM EDGE BICLIQUE problem. The problem was proved to be NP-complete (Lonardi, Szpankowski, and Yang 2006; Peeters 2003), and challenging to approximate (Ambühl, Mastrolilli, and Svensson 2011; Feige 2002; Feige and Kogan 2004; Goerdt and Lanka 2004; Peeters 2003).
2. μ(I, J) = |I| , where |I| = |J| — known as the BALANCED COMPLETE BIPARTITE SUBGRAPH problem (also known as the balanced biclique problem). The problem was proved to be NP-complete (Garey and Johnson 1979).
3. μ(I, J) = |I| + |J| — known as the MAXIMUM VERTEX BICLIQUE problem. The problem can be solved in polynomial time using a minimum cut algorithm (Hochbaum 1998; Garey and Johnson 1979).

To achieve better value-for-money, demand aggregation aims to encapsulate the largest possible number of purchasing orders, and to "replace" them with one order. i.e., replace |I|×|J| individual purchasing orders (|I| items bought from |J| vendors) with one purchasing order (which includes the |I| items from e.g., the cheapest vendor). As such, we focus on the problem of finding the set of maximal edge bicliques (potentially overlapping). Each such maximal edge biclique will serve as a potential demand aggregation. We propose an efficient Subspace Biclique Clustering for Procurement (SBCP) algorithm to tackle this challenging problem. Extensive experimentations on artificial and real-world procurement datasets demonstrate the superiority of our proposed SBCP algorithm over state-of-the-art techniques.

Use of AI Technology

We are now ready to present the SBCP algorithm. Firstly, we describe a Monte Carlo algorithm for extracting a list of maximal bicliques. Next, we prove that the list contains optimal bicliques. Finally, we present the run-time analysis of the algorithm.

Finding Maximal Bicliques

For ease of readability, we adopt the graph's adjacency matrix representation, defined as follows (see the example in Figure 5b, which is the adjacency matrix representation of the bipartite graph G in Figure 5a).

Definition 2. Let G = (U ∪ V, E) be a bipartite graph such that |U| = m, and |V| = n. The adjacency matrix X of graph G is a [m × n] matrix such that $X_{i,j} = 1$ if (i,j) ∈ E and $X_{i,j} = 0$ otherwise.

The input of the SBCP algorithm is therefore an adjacency matrix X of a given bipartite graph G, consisting of only boolean numbers, namely 0 and 1. The output of the SBCP algorithm is a list of maximal bicliques, i.e., a list of submatrices of ones, representing maximal bicliques within G (the graph may contain multiple, possibly overlapping, maximal bicliques). The SBCP algorithm itself uses a subspace clustering approach (Lonardi, Szpankowski, and Yang 2006; Procopiuc et al. 2002; Shaham, Yu, and Li 2016). This common technique uses iterative random projection (i.e., a Monte Carlo strategy) to obtain the biclique's seed, which is later expanded into a maximal biclique.

The SBCP Algorithm

Algorithm 1 presents the SBCP algorithm. As in the case of many Monte Carlo algorithms, the structure of the SBCP algorithm is very simple, and can be divided into the following stages:
(i) Seeding (lines 2–4): a random selection of a set of rows to serve as a seed of the maximal biclique.
(ii) Addition of rows and columns (lines 5–20): interleaved accumulation of rows (lines 9–14) and columns (lines 15–20), which comply with the rows and columns already accumulated.
(iii) Polynomial repetition (line 1): repetition of the above two steps provides a probabilistic guarantee of acquiring a set of maximal bicliques.

Remark 1. The Monte Carlo nature of the SBCP algorithm is revealed in phase (i) where random seeds are generated. The subspace clustering nature of the SBCP algorithm is revealed in phases (ii), where the seed of phase (i) is expanded to form a maximal subset of rows over a maximal subset of columns, i.e., a maximal biclique.

Remark 2. To ease readability, lines 10 and 16 use the short notations of: $X_{i,J} = 1$ and $X_{I,j} = 1$, respectively, which have the meaning of: $\forall j \in J$, $X_{i,j} = 1$ and $\forall i \in I$, $X_{i,j} = 1$, respectively.

Remark 3. The SBCP algorithm has an inherent ability to mine multiple, possibly overlapping, bicliques by utilizing the independent random projection on each repetitive run, to reveal columns and rows relevant only to a specific biclique.

Remark 4. The SBCP algorithm is not designed for the enumeration of all maximal bicliques, which may be exponential in size (Eppstein 1994; Zhang et al. 2014). The algorithm has a polynomial number of iterations, and thus, the size of the return list is also polynomial. However, we next prove that the returned list contains, with a fixed probability, optimal bicliques.

Algorithm 1: **SBCP** algorithm for extracting a list of maximal bicliques.

Input: X, a [m × n] matrix of boolean numbers.
Output: List of maximal bicliques.
Initialization: Setting of N, $|P|$ and $|S|$ is discussed in the following section.

```
1: loop N times
2:    // Seeding phase
3:    choose a subset of rows P uniformly at random;
4:    set I ← P, J ← ∅;
5:    // Interleaving row and column addition phase
6:    set isAddRow ← False;
7:    set row i ← 1, column j ← 1;
8:    while row i ≤ m or column j ≤ n do
9:       if isAddRow then // row addition
10:         if X_{i,J} = 1 then
11:            add i to I;
12:         i ← i + 1;
13:         if j ≤ n then
14:            isAddRow ← !isAddRow
15:      else // column addition
16:         if X_{I,j} = 1 then
17:            add j to J;
18:         j ← j + 1;
19:         if i ≤ m ∧ (|J| ≥ |S| ∨ j > n) then
20:            isAddRow ← !isAddRow
21: return list of (I, J);
```

Optimality of the Algorithm

Clearly, the proposed SBCP algorithm can be viewed as a heuristic method. Next, we prove that there are solid theoretical reasons for this efficacy.

The SBCP algorithm derives inspiration from the BSC algorithm (Shaham, Yu, and Li 2016). The motivation to enhance the BSC algorithm is to avoid its tendency to be stuck in local maxima, which results in mining degenerated bicliques, i.e., bicliques that have large number rows but small number of columns (or the other way around, i.e., small number of rows and large number of columns). Such degenerated bicliques are less applicable, particularly in an industrial scenario usage such as demand aggregation. Next, we outline the intuition behind the SBCP algorithm. For detailed argumentations, proofs, run-time analysis, and comparisons to existing algorithms, we refer the reader to Shaham, Yu, and Li (2016).

The intuition behind the BSC algorithm is that once we successfully draw a discriminating column set (subset of the rows), we can use it to collect the biclique's columns, and only then, use the collected columns in order to collect the

biclique's rows. The mechanism behind the SBCP algorithm is similar. Unlike the BSC algorithm which collects all of the biclique's columns and **only then** collects all of the biclique's rows, the SBCP algorithm collects the biclique's rows and columns in an alternating fashion. The intuition is that an alternating expansion of the initial discriminating set would result in an **ever-growing discriminating set**. This promises better discriminating results (see Theorem 3.1 and Experiment I (Shaham, Yu, and Li 2016)), which in turn leads to a better detection probability (see Theorem 3.2 (Shaham, Yu, and Li 2016)), which results in an overall reduced run-time (see Subsection 3.3 (Shaham, Yu, and Li 2016)).

Run-time

The run-time is polynomial: $mn^{O(1)}$ (see Subsection 3.3 (Shaham, Yu, and Li 2016)).

Application Use and Payoff

The implementation of the SBCP algorithm to create a practical procurement solution for A*PO was an iterative process. Prior to achieving its final version, the solution underwent three major rounds of evaluation. Each evaluation ran the refined algorithm over a dataset of Purchasing Orders (POs), passing the mined Demand Aggregation (DA) patterns, with their related POs, to a procurement officer for evaluation. The quality of the algorithm was assessed using the following metrics: (1) *precision*; (2) *recall*; (3) *professional assessment* of usefulness for newly detected and unknown DA patterns according to A*PO expertise; and (4) number of patterns matching with existing DAs, i.e., 33 previously known DAs curated by A*PO legacy methods. Formally, those are defined are follows:

1. $Precision = \dfrac{valid\ POs}{all\ mined\ POs}$
2. $Recall = \dfrac{valid\ POs}{relevant\ POs}$
3. $Professional\ assessment = \dfrac{new\ valid\ DAs}{all\ new\ mined\ DAs}$
4. $Matching\ with\ existing\ DAs = \dfrac{detected\ past\ DAs}{total\ past\ DAs}$

The above metrics were picked in order to reflect the relevant aspects of A*PO operations. Metrics (1) and (2) were calculated with respect to POs within each detected pattern, i.e., determining the quality of each pattern regardless to how many patterns were detected in total. Furthermore, the procurement officer would treat POs listed by the algorithm as a starting point, and further investigate using traditional A*PO techniques whether other orders should have been included. Metrics (3) and (4) focus on the assessment of the

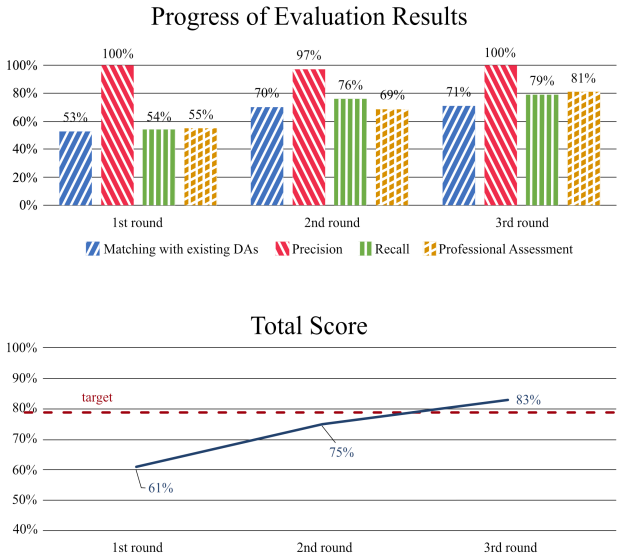

Figure 6. Evaluation progress.

mined DAs from a past and future business perspective. The evaluation of *precision*, *recall* and *professional assessment* employed an additional constraint: each was calculated based on 30 randomly picked DAs out of all returned by the algorithm. Such methodology was necessary as the evaluation of each pattern took considerable time and it was impossible for A*PO staff to look through all detected patterns during each evaluation round.

For the first round of evaluation, we began with running the SBCP algorithm on the entire A*STAR procurement database for the years 2009 to 2016 (inclusive). To recall from Section 2, the database comprises 660,162 items, 14,834 vendors, and 1,032,275 POs.

The evaluation by A*PO found various issues (e.g., price depreciation, coarse/grained clusters). Following A*PO feedback, we added domain constraints (encapsulating A*PO best practices) in the next two evaluation rounds:

1. Use of only the most recent 3 years (i.e., years 2014–2016). This resulted in a database comprising 271,219 items, 7,319 vendors, and 391,671 POs.
2. Filter out patterns with less than 20 POs a year.
3. Filter out patterns with a value less than S$70,000 a year.
4. Filter out patterns with a descending price trend (trend calculated as linear regression).
5. Allow patterns to fail once on the above filters.

After incorporating these new constraints, the SBCP miner found 643 patterns. The final performance (see Figure 6) achieved 71% detected legacy DAs (the existing DAs that were not identified by the engine were A*PO manually crafted DAs, which violated A*PO-imposed domain constraints). The performance for new DAs mined by the engine was 81% (i.e., 81% of the newly identified DAs were

deemed by A*PO staff as useful cases for potential future bulk tender contracts, to be thoroughly investigated by A*PO). Furthermore, all POs assigned to the valid DAs were correctly mined (100% precision); at the same time, the valid DAs were determined 79% complete in terms of listed POs in comparison to what the procurement officer thought should have been reported (79% recall).

A*PO calculated the estimation of the incurred savings resulting from the usage of DAs, as follows. First, from a chosen DA, a potentially discounted price is calculated as the cheapest item price among an item subcluster. This is repeated for all subclusters and subsequently for all DAs. In the second step, the discounted price is computed by comparing with actual prices for items in the last 3 years. The total savings were estimated to be > S$7 million, after accounting for cost appreciation. Although this is an approximation, note that additional discounts from buying in bulk and potential cost savings from reduced legal and administration overheads were not included. Thus, we believe the estimate is quite conservative.

Application Implementation and Deployment

In practice, in a production scenario, new demand aggregation patterns need a rather lengthy time period in order to surface. Therefore, at deployment stage the algorithm did not need to run constantly processing the stream of incoming purchase orders in real-time. Instead, A*PO aimed to use the system as a decision-supporting mechanism for their annual management reports. At bootstrap, our system was used to reveal past patterns from historical data which were unknown to A*PO. Moving forward, as new purchases are made and new data comes in, the system's purpose is to give suggestions of new/modified patterns to the procurement officer; those patterns would subsequently be reviewed manually by the officer and, if useful, recommend to management for new contracting/tendering decisions. To that end, the deployed application had the following characteristics as described below.

Implementation

The SBCP algorithm was implemented in Java 8, and deployed on a multi-core server running CentOS, using common libraries such as Apache 2.0 + LGPL, Commons Math, FastUtil and JavaCSV.

Input/Output

The input was extracted out of A*PO systems as a CSV format comprising the following attributes: (1) requester name; (2) item description; (3) material group; (4) vendor name; and (5) creation/approval date.

The output was also in the form of a CSV report format, accompanied by a dashboard analytics tool (see Figure 7).

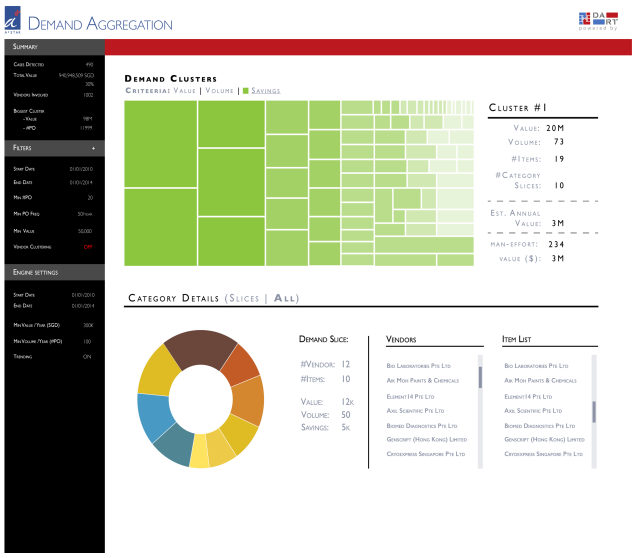

Figure 7. Dashboard analytics view.

Deployment Tweaks and Lessons Learned

As part of our partnership, A*PO personnel were deeply involved in all stages of the project. Algorithmic decisions, such as to model the procurement dataset as a bipartite relationship graph and to use bicliques to represent demand aggregation patterns, were explained to our collaborators. Consequently, A*PO procurement officers were able to give us active feedback on refinements, which iteratively helped to improve the algorithm's performance. Although a number of these tweaks are hard to quantify, they were equally important for A*PO in terms of the deployment and practical use of the solution:

1. Adjusting output to preference and analysis capability and capacity of the end users: during the first evaluation, we discovered that our end users were overwhelmed by the amount of potential demand aggregation patterns. The SBCP algorithm, as other biclique mining algorithms, strives to deliver a complete set of all possible bicliques. However, in practical terms, this abundance turns out to be counterproductive to the end users, as it is too much to digest. A constant communication of the results to the users led to incorporating business rules, which helped to trim and consolidate the demand aggregation patterns list, and to meet the end users' needs and capacity.
2. The classic evaluation via precision and recall measurements did not turn out to be particularly relevant from a business standpoint of A*PO: the end users were not interested in patterns which in their understanding seemed obvious (i.e., patterns which could be established by a skilled procurement officer even without data examination or with little effort via BI tools). This resulted in a ranking mechanism on top of the SBCP algorithm, which up-votes "interesting" and "non-obvious" patterns. The criteria for those were discussed throughout the evaluations and involved incorporating additional data fields

(e.g., item categories, requesting officers, approving officers, historical purchasing trends), in combination with various thresholds and rules.

3. Redefining requirements for the validity of DA patterns: previous patterns curated manually by A*PO considered the aggregation of purchases across different material groups as valid. However, this turned out to be less valuable when using our algorithm, as it resulted in a greatly increased number of patterns and generated patterns with weak business logic. Therefore, we enabled the output system to break down the mined DAs into their material groups' components.

Maintenance

Future maintenance was an important criterion during the application design. With a lean IT team supporting A*PO, we designed the system with loosely coupled data application and UI layers, where the data exchange between the layers happens via CSV files in pre-defined formats. Thus, the end users and supporting IT team have full flexibility in updating the data feed into the application in view of future database upgrades and in maintaining the Tableau UI should there be new requirements.

Support for the application layer is currently provided by our team. Our future plan is to license the technology to a local company to handle the technical support and maintenance, and also productize it and commercialize further.

Related Work

The MAXIMUM EDGE BICLIQUE problem has received much attention in academic research in recent years due to its wide range of applications in areas such as bioinformatics (Ben-Dor et al. 2003; Cheng and Church 2000; Sanderson et al. 2003), epidemiology (Mushlin et al. 2007), formal concept analysis (Ganter and Wille 1999), manufacturing problems (Dawande et al. 2001), molecular biology (Nussbaum et al. 2010), machine learning (Mishra, Ron, and Swaminathan 2003), management science (Swaminathan and Tayur 1998), database tiling (Geerts, Goethals, and Mielikäinen 2004), and conjunctive clustering (Mishra, Ron, and Swaminathan 2003).

Most of the algorithms tackling the problem can be divided into the following three main categories: (i) exploitation of some class of graph; (ii) relaxation of the problem by bounding a characteristic of a graph; and (iii) reduction to other domains. Among the algorithms in the first category we can find: limitation to chordal bipartite graphs (Kloks and Kratsch 1995), and limitation to convex bipartite graphs (Alexe et al. 2004; Kloks and Kratsch 1995; Nussbaum et al. 2010). Among the algorithms in the second category we can find: bounding the graph's vertices degree (Alexe et al. 2004; Liu et al. 2006), bounding the graph's arboricity (Eppstein 1994), and bounding the biclique's size (Liu et al. 2006; Sanderson et al. 2003). Among the algorithms in the last category we can find: reduction to maximal clique (Makino and Uno 2004; Tomita, Tanaka, and Takahashi 2006), and reduction to frequent itemsets (Li et al. 2007; Uno, Kiyomi, and Arimura 2004; Zaki and Hsiao 2002).

The reduction of the problem to finding a maximal clique has the benefit of a well-researched field with an abundance of heuristics and approximation algorithms (recall that the problem is NP-complete (Peeters 2003)). However, in order to perform such reduction, an inflation of the bipartite graph is needed, which makes the problem computationally impractical due to the large number of edges it adds.

The reduction of the problem to frequent itemsets benefits, as above, by relying on a rich field of research. However, it contains other difficulties. Literature has shown (Zaki and Ogihara 1998) that a transactional database corresponds to a bipartite graph $G = (U \cup V, E)$, where U is the set of items (itemsets), V is the set of transactions (tids), and E is the set of pairs (item, transaction), i.e., an edge in the bipartite graph represents a transaction comprising the item. Take for example a set of products offered by a supermarket. A transaction would be a subset of products (itemset) purchased by a customer. Therefore, finding frequent itemsets corresponds to finding bicliques, and finding a maximal frequent closed itemset corresponds to finding a maximal biclique. However, transforming a frequent itemset to a biclique, although requiring a trivial post-processing step, can be highly time consuming (Li et al. 2007). Therefore, these reductions to other domains may present practical and scalability problems (Zhang et al. 2014) and may not fully utilize the special characteristics of the bipartite graph.

Out of other approximation work (Geerts, Goethals, and Mielikäinen 2014), it is worth noting the problem of finding the number of edges to be deleted such that the resulting graph is a 2–approximation maximum edge biclique (Hochbaum 1998), and the finding of near complete bicliques (Li et al. 2008; Liu, Li, and Wang 2008; Mishra, Ron, and Swaminathan 2003; Mishra, Ron, and Swaminathan 2004; Sim et al. 2006; Sim et al. 2009; Sim et al. 2011).

Zhang et al. (2014) introduced the iMBEA algorithm for the enumeration of maximal bicliques in a bipartite graph. The algorithm uses an efficient branch-and-bound technique to prune away non-maximal subtrees of the search tree. Despite the theoretical complexity of an exponential run-time, the algorithm outperforms the previous state-of-the-art algorithms: (i) MICA (Alexe et al. 2004) – the best known general graph algorithm; and (ii) LCM-MBC (Li et al. 2007) – a prime frequent closed itemsets algorithm (an improvement of LCM (Uno, Kiyomi, and Arimura 2004) algorithm).

Recent work by Shaham, Yu, and Li (2016) used a subspace clustering approach for finding the maximum edge biclique in a bipartite graph. The algorithm, named BSC (Biclique Subspace Clustering), uses random projections to

obtain seed patterns, which are later grown into maximal edge bicliques. The repetitive projections ensure a probabilistic guarantee on the finding of the maximum edge biclique. The algorithm is reported to be at least four orders of magnitude faster than the iMBEA algorithm on both artificial and real-world datasets.

To the best of our knowledge, only a few papers have been published in the specific field of demand aggregation for procurement (Chew 2017; Chowdhary et al. 2011; Wang and Miller 2005). These papers use a naïve one-dimension aggregation strategy, which suffers from the "curse of dimensionality" (Bellman 1961; Beyer et al. 1999; Kriegel, Kröger, and Zimek 2009), and from sensitivity to text descriptions variability and the effectiveness of the similarity measure used.

Conclusions and Future Work

Aggregating procurement demands could lead to better value-for-money and substantial cost savings for large organizations. This paper describes our experience in developing an AI solution for demand aggregation and deploying it in A*STAR, a large governmental research organization in Singapore with procurement expenditure in the scale of hundreds of millions of dollars annually.

We formulate the demand aggregation problem using a bipartite graph model depicting the relationship between procured items and target vendors. We then show that identifying maximal edge bicliques within that graph would reveal potential demand aggregation patterns. We propose an unsupervised learning methodology for efficiently mining such bicliques using a novel Monte Carlo subspace clustering approach. While the method can be viewed as an effective heuristic, we show that there is a strong theoretical base for its efficacy. The algorithm enhances the BSC algorithm. As such, not only it inherits the theoretical analysis, but also achieves lower complexity bounds and higher detection probability.

A proof of concept prototype was developed and tested with the end users during 2017, and later trialed and iteratively refined, before being rolled out in 2019. The final performance achieved on past cases benchmark was: 71% of past DAs transformed into bulk tenders were correctly detected by the engine. The performance for new opportunities pointed out by the engine was 81% (i.e., 81% of the newly identified cases were deemed useful cases for potential bulk tender contracts in the future). Additionally, per each valid DA identified, the engine achieved (in terms of POs) 100% precision (all aggregated POs identified by the engine were correct), and 79% recall (the engine correctly identified 79% of POs that should have been put into the DAs). Overall, the direct cost savings from the true positive contracts spotted so far are estimated to be S$7 million annually (this in addition to potential savings due to reduced legal and administration overheads).

As part of future work, we are exploring further improvements in accuracy and scalability of the core algorithm. In addition, we plan to license the technology to a local company to handle the technical support and maintenance, and productize and commercialize it further.

References

Alexe, G.; Alexe, S.; Crama, Y.; Foldes. S.; Hammer, P. L.; and Simeone, B. 2004. Consensus Algorithms for the Generation of All Maximal Bicliques. *Discrete Applied Mathematics* 145(1): 11–21.

Ambühl, C.; Mastrolilli, M.; and Svensson, O. 2011. Inapproximability Results for Maximum Edge Biclique, Minimum Linear Arrangement, and Sparsest Cut. *SIAM Journal on Computing* 40(2): 567–596.

Bartolini, A. 2011. Innovative Ideas for the Decade, Technical Report. Ardent Partners Research.

Bellman, R. 1961. *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton University Press.

Ben-Dor, A.; Chor, B.; Karp R.; and Yakhini, Z. 2003. Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem. *Journal of Computational Biology* 10 (3-4): 373–384.

Beyer, K.; Goldstein, J.; Ramakrishnan, R.; and Shaft, U. 1999. When is "Nearest Neighbor" Meaningful?. In Proceedings of the International Conference on Database Theory, 217–235. Berlin, Heidelberg: Springer Berlin Heidelberg.

Cheng Y., and Church G. M. 2000. Biclustering of Expression Data. In Proceedings of the International Conference on Intelligent Systems for Molecular Biology, 93–103. Palo Alto, CA: AAAI Press.

Chew, T. H. 2017. Procurement Demand Aggregation. In *Business Analytics: Progress on Applications in Asia Pacific*, edited by J. L. C. Sanz, 664–684. Singapore: World Scientific.

Chowdhary, P.; Ettl, M.; Dhurandhar, A.; Ghosh, S.; Maniachari, G.; Graves, B.; Schaefer, B.; and Tang, Y. 2011. Managing Procurement Spend Using Advanced Compliance Analytics. In Proceedings of the IEEE International Conference on e-Business Engineering, 139–144.

Dawande, M.; Keskinocak, P.; Swaminathan, J. M.; and Tayur, S. 2001. On Bipartite and Multipartite Clique Problems. *Journal of Algorithms* 41(2): 388–403.

Eppstein, D. 1994. Arboricity and Bipartite Subgraph Listing Algorithms. *Information Processing Letters* 51(4): 207–211.

Feige, U. 2002. Relations Between Average Case Complexity and Approximation Complexity. In Proceedings of the ACM Symposium on Theory of Computing, 534–543. New York, NY: Association for Computing Machinery.

Feige, U.; and Kogan, S. 2004. Hardness of Approximation of the Balanced Complete Bipartite Subgraph Problem, Technical Report MCS04-04. The Weizmann Institute of Science.

Ganter B., and Wille, R. 1999. *Formal Concept Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Garey, M. R, and Johnson D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Geerts, F.; Goethals, B.; and Mielikäinen, T. 2004. Tiling Databases. In Proceedings of the International Conference on Discovery Science, 278–289. Berlin, Heidelberg: Springer Berlin Heidelberg.

Goerdt, A., and Lanka, A. 2004. An Approximation Hardness Result for Bipartite Clique. Technical Report 48. Electronic Colloquium on Computation Complexity.

Hochbaum, D. S. 1998, Approximating Clique and Biclique Problems. *Journal of Algorithms* 29(1): 174–200.

Kloks, T., and Kratsch, D. 1995. Computing a Perfect Edge Without Vertex Elimination Ordering of a Chordal Bipartite Graph. *Information Processing Letters* 55(1): 11–16.

Kriegel, H. P.; Kröger, P. and Zimek, A. 2009. Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering. *ACM Transactions on Knowledge Discovery from Data* 3(1): 1–58.

Kunegis, J. 2013. KONECT – The Koblenz Network Collection. In Proceedings of the International Conference on World Wide Web Companion, 1343–1350. New York, NY: Association for Computing Machinery

Li, J.; Liu, G.; Li, H.; and Wong, L. 2007. Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A One-to-One Correspondence and Mining Algorithms. *Transactions on Knowledge and Data Engineering* 19(12): 1625–1637.

Li, J.; Sim, K.; Liu, G.; and Wong, L. 2008. Maximal Quasi-Bicliques with Balanced Noise Tolerance: Concepts and Co-Clustering Applications. In Proceedings of the SIAM International Conference on Data Mining, 72–83.

Liu, G.; Sim, K.; and Li, J. 2006. Efficient Mining of Large Maximal Bicliques. In Proceedings of the International Conference on Data Warehousing and Knowledge Discovery, 437–448.

Liu, X.; Li, J.; and Wang, L. 2008. Quasi-Bicliques: Complexity and Binding Pairs. In Proceedings of the International Computing and Combinatorics Conference, 255–264.

Lonardi, S.; Szpankowski, W.; and Yang, Q. 2006. Finding Biclusters by Random Projections. *Theoretical Computer Science* 368(3): 217–230.

Makino, K., and Uno, T. 2004. New Algorithms for Enumerating All Maximal Cliques. In Proceedings of the Scandinavian Workshop on Algorithm Theory, 260–272.

Melkman, A. A., and Shaham, E. 2004. Sleeved CoClustering. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 635–640.

Mishra, N.; Ron, D.; and Swaminathan, R. 2003. On Finding Large Conjunctive Clusters. In Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop, 448–462.

Mishra, N.; Ron D.; and Swaminathan, R. 2004. A New Conceptual Clustering Framework. *Machine Learning* 56: 115–151.

Mushlin, R. A.; Kershenbaum, A.; Gallagher, S. T.; and Rebbeck, T. R. 2007. A Graph-Theoretical Approach for Pattern Discovery in Epidemiological Research. *IBM Systems Journal* 46(1): 135–149.

Nussbaum, D.; Pu, S.; Sack, J. R.; Uno, T.; and Zarrabi-Zadeh, H. 2010. Finding Maximum Edge Bicliques in Convex Bipartite Graphs. In Proceedings of the International Computing and Combinatorics Conference, 140–149.

Peeters, R. 2003. The Maximum Edge Biclique Problem is NP-Complete. *Discrete Applied Mathematics* 131(3): 651–654.

Procopiuc, C. M.; Jones, M.; Agarwal, P. K.; and Murali, T. 2002. A Monte Carlo Algorithm for Fast Projective Clustering. In Proceedings of the International Conference on Management of Data, 418–427.

Sanderson, M. J.; Driskell, A. C.; Ree, R. H.; Eulenstein, O.; and Langley, S. 2003. Obtaining Maximal Concatenated Phylogenetic Data Sets from Large Sequence Databases. *Molecular Biology and Evolution* 20(7): 1036–1042.

Shaham, E.; Yu, H.; and Li, X. L. 2016. On Finding the Maximum Edge Biclique in a Bipartite Graph: A Subspace Clustering Approach. In Proceedings of the SIAM International Conference on Data Mining, 315–323.

Shaham, E.; Westerski, A.; and Kangasabai, R. 2019. Method and Apparatus for Procurement Demand Aggregation. International Patent No. WO2019231390A1.

Sim, K.; Li, J.; Gopalkrishnan, V.; and Liu, G. 2006. Mining Maximal Quasi-Bicliques to Co-Cluster Stocks and Financial Ratios for Value Investment. In Proceedings of the International Conference on Data Mining, 1059–1063.

Sim, K.; Li, J.; Gopalkrishnan, V.; and Liu, G. 2009. Mining Maximal Quasi-Bicliques: Novel Algorithm and Applications in the Stock Market and Protein Networks. *Statistical Analysis and Data Mining 2 (4)*: 255–273.

Sim, K.; Liu, G.; Gopalkrishnan, V.; and Li, J. 2011. A Case Study on Financial Ratios via Cross-Graph Quasi-Bicliques. *Information Sciences* 181(1): 201–216.

Swaminathan, J. M., and Tayur, S. R. 1998. Managing Broader Product Lines Through Delayed Differentiation Using Vanilla Boxes. *Management Science* 44(12): 161–172.

Tomita, E.; Tanaka, A.; and Takahashi, H. 2006. The Worst-Case Time Complexity for Generating All Maximal Cliques and Computational Experiments. *Theoretical Computer Science* 363(1): 28–42.

Uno, T.; Kiyomi, M.; and Arimura, H. 2004. LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets. In Proceedings of the ICDM Workshop on Frequent Itemset Mining Implementations 126.

Wang, G. and Miller, S. 2005. Intelligent Aggregation of Purchase Orders in E-Procurement. In Proceedings of the IEEE International EDOC Enterprise Computing Conference, 27–36.

Westerski, A.; Kangasabai, R.; Wong, J.; and Chang H. 2015. Prediction of Enterprise Purchases Using Markov Models in Procurement Analytics Applications. *Procedia Computer Science* 60: 1357-1366.

Westerski, A.; Kangasabai, R.; and Sim, K. 2017. Method of Detecting Fraud in Procurement and System Thereof. International Patent No. WO2017116311A1.

Zaki, M. J., and Hsiao, C. J. 2002. CHARM: An Efficient Algorithm for Closed Itemset Mining. In Proceedings of the International Conference on Data Mining, 457–473.

Zaki M. J., and Ogihara, M. 1998. Theoretical Foundations of Association Rules. In Proceedings of the Workshop on Research Issues on Data Mining and Knowledge Discovery, 71–78.

Zhang, Y.; Phillips, C. A.; Rogers, G. L.; Baker, E. J.; Chesler, E. J.; and Langston, M. A. 2014. On Finding Bicliques in Bipartite Graphs: A Novel Algorithm and its Application to the Integration of Diverse Biological Data Types. *BMC Bioinformatics* 15(1): 110–127.