中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-17-005

Efficient Randomized Algorithms for Large-scaled Exact Matchings with Multiple Controls: Implementation and Applications

Hung-Jui Chang, Yu-Hsuan Hsu, Chih-Wen Hsueh,
and Tsan-sheng Hsu

Dec. 13, 2017 || Technical Report No. TR-IIS-17-005
http://www.iis.sinica.edu.tw/page/library/TechReport/tr2017/tr17.html

Efficient Randomized Algorithms for Large-scaled Exact Matchings with Multiple Controls: Implementation and Applications

Hung-Jui Chang[1,2], Yu-Hsuan Hsu[1], Chih-Wen Hsueh[1], and Tsan-sheng Hsu*[2]

[1]Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
{r03922013, cwhsueh}@csie.ntu.edu.tw
[2]Institute of Information Science, Academia Sinica, Taipei, Taiwan
{chj, tshsu}@iis.sinica.edu.tw

Abstract

Background: Extracting sequential patterns, in which an event α happens before another event β with high probability, from a time-stamped event database is an important task in data science. In cohort studies, which are frequently conducted on data in such databases, it is essential to find a matching between the chosen cases and the control candidates to identify the causality when the control candidates that cases can match are different. The hidden causality is investigated by observing several statistical indicators in the returned matched results.
Methods: We present an efficient exact matching method that matches each case with exactly K controls such that a control candidate can be matched with at most one case, where K is an input constant.
Results: Our experiment results show that the method is fast and memory efficient, even though the database contains a large number of events. Moreover, the quality of matchings found is high, i.e., a control candidate is matched with all matchable cases with roughly equal probability when multiple matchings are performed. We posit that to obtain stable statistical results, K should be set to be around, but not larger than, the maximum matching ratio, which is the largest integer L. Then each case can be matched with exactly K controls rather than a fixed constant as proposed in previous studies. In addition, to ensure that the results are stable, several such matchings should be performed to find the average, instead of performing the matching just once.

*Corresponding Author

Keywords Case-control studies, randomized matching, cohort study, exact matching

1 Introduction

Finding association rules in which two events, α and β, will happen together with high probability, is one of the most important problems in data mining. Most approaches are based on the Apriori algorithm [2, 15, 23] and its variations, which can be applied in traditional market management [2, 15], as well as in medical research [18, 24]. When data is time relevant, the basic association rule can not fully express the relation between events. A sequential pattern, in which an event α has a high probability of being followed by an event β, is a generalization over the association rule. Such a pattern not only represents the association between events, but also shows the order of their occurrence [18, 23].

1.1 Data Source

It has been shown that a good medical record database can provide a great deal of useful information to help researchers find underlying sequential patterns, i.e., causal relations, between diseases [4]. In this paper, we use Taiwan's National Health Insurance Research Database (NHIR Database or NHIRD) [1] as the data source. Every resident in Taiwan must register with the National Health Insurance (NHI) program, which covers more than 95% of the medical treatments provided nation-wide. NHIRD stores the claim records of all patients who have received treatment. A claim record contains detailed information, such as disease diagnosis records as ICD-9 codes, the drugs prescribed, and any surgical procedures performed. That is, the event sets are heterogeneous. Basically, a record can be treated as a three-tuple, (P, E, D), comprised of the person P, the corresponding event E, and the record date D. NHIRD also contains personal data, such as demographic and financial information about each person. Traditionally, medical studies have used the NHIRD records to identify potential risk factors of particular diseases [5, 17, 27] by determining the causality between events in NHIRD. The corresponding sequential pattern may span different event sets, e.g., from medical treatment to disease diagnosis, or from particular drugs used to another disease diagnosis.

1.2 Cases and Controls

To determine a plausible causality in a time-stamped events database, it is often necessary to conduct a *randomized controlled trial*. The first step is to identify the *study cases*. Given two events, α and β, we want to check whether the occurrence probability of event β is affected by event α happening previously among all persons in NHIRD. In this case, α is the control variable, which separates all individuals in NHIRD into two groups; G_α, those with α; and $G_{\neg\alpha}$, those without α. For an individual h in group G_α, the first occurrence day of

	β	$\neg\beta$	Row sum
α	s	t	$s+t$
$\neg\alpha$	u	v	$u+v$
Col. Sum	$s+u$	$t+v$	N

Table 1: The four groups of a matching.

the event α is called the α *index day*. If we consider the β index day of an individual h, there are three possible cases: 1) the β index day is earlier than the α index day; 2) the β index day is later than the α index day; or 3) the event β is not included in h's record. Note that the third case is a special instance of the second case; that is, the β index day is infinity. We are interested in cases where α leads to β being more likely to happen in the future. Therefore, if the β index day is earlier than the α index day, i.e., the first case, then this sequence is meaningless. Consequently, we only focus on the second and third cases whose α index dates are earlier than their corresponding β index dates. These two cases form the *study cases*. All persons in group $G_{\neg\alpha}$ are the *control candidates*.

In a cohort study, to determine meaningful causality with litter statistical bias, each study case must be matched with K different control candidates, where K is a given integer [25]. The output matching needs to satisfy some quality criteria in terms of randomness. That is, a control candidate has roughly equal chance of being matched with any of the matchable study cases. Based on some pre-determined constraints, a case can only be matched with a subset of persons in $G_{\neg\alpha}$. For example, the individuals in $G_{\neg\alpha}$ can be matched with a case of the same birth year and gender. Further constraints are possible depending on the reason event pairs α and β are checked for. The constant K is called the *matching ratio*. To obtain a matching, a control candidate can be matched to a study case whose α index day is later than his β index day. Note that in our definition, a control candidate with no β can be matched with all study cases. If there are 100 study cases and $K = 10$, then each study case can be matched with 10 control candidates such that a control candidate can be matched with at most one case. Hence, there will be 1,000 matched controls if the matching is successful. The set of matched control candidates is called the *control group*. In some studies, if a case does not have K matched candidates, it is dropped to avoid the situation where a few difficult-to-match cases cause the study to fail. The goal of matching is to find as many controls on non-dropped cases as possible [10]. Some matching methods assign a propensity score to each matched pair [20]. The goal is to return a matching with the best total score. In practice, the verification of where an event actually happens, (called phenotyping), may require extra computation [14]. It also necessary to consider excluding cases of persons with α who have some pre-existing or co-existing conditions, i.e., when α is a disease, its comorbidities should be considered [20]. These are interesting practical issues, but they are beyond the scope of this study.

	β	$\neg\beta$	Row sum
α	s	t	$s+t$
$\neg\alpha$	$\sim a \cdot u$	$\sim a \cdot v$	$\sim a \cdot (u+v)$
Col. Sum	$\sim s + a \cdot u$	$\sim t + a \cdot v$	N′

Table 2: The four groups of another matching with a matching ratio that is a times larger than that of the matching in Table 1.

1.3 Causality Inference

In the following example, we calculate how the occurrence of α affects the chances of β happening later. After matching, we obtain a study group and a control group, each of which we divide into two parts based on whether the individuals have event β or not. In Table 1, the number of persons in the four subgroups, $\alpha \to \beta$, $\alpha \to \neg\beta$, $\neg\alpha \to \beta$ and $\neg\alpha \to \neg\beta$ is denoted by s, t, u and v respectively; and the total number of persons in G_α and $G_{\neg\alpha}$ is denoted by $s+t$ and $u+v$ respectively. Let $N = s+t+u+v$. For each group, we calculate the *occurrence rate* of β. In the study group, the *risk factor* R_1 is defined as $s/(s+t)$; and in the control group, the *risk factor* R_2 is defined as $u/(u+v)$. The *Relative Risk factor*, RR factor [22], which is defined as $R_1/R_2 = s(u+v)/u(s+t)$, is used to measure the relative risk of having β with and without α occurring previously. RR will be roughly equal to 1 if α does not affect the probability of β occurring later. If RR is much larger than 1 ($R_1 >> R_2$), then α will have a positive influence on the probability of β occurring later. Conversely, if RR is much less than 1 ($R_1 << R_2$), α will have a negative influence on the probability of β occurring later. Other statistical indicators, such as the *Odds Ratio* (OR), may be used instead of RR[22]. Another important statistical indicator is called the *p-value*, which determines whether the statistical conclusion is *meaningful* or *significant*. The calculation of the p-value is based on Fisher's exact test [7], where

$$p\text{-value}(s,t,u,v) = \frac{(s+t)!(u+v)!(s+u)!(t+v)!}{N!s!t!u!v!}. \quad (1)$$

In the test, when the column sum and row sum are given and assuming each case happens in a random manner, the probability that a random distribution will generates the specific observed result can be calculated. When the p-value is less than a threshold, say 0.001, it is regarded as *statistically significant*, or unlikely to happen at random. Other measurements for statistical significance can also be used, e.g., the Z-value [28] or the tail probability [19].

1.4 Research Issues

In the above process, the matching ratio K is a constant. Intuitively, when K is small, the matching result will be unstable because the percentage of matched control cases over all control candidates is very small. Hence, setting K as large as possible can reduce the statistical bias. However when K is larger,

the matching result is often over-estimated as the control group is similar to the whole population (from which $G_{\neg\alpha}$ is drawn), rather than the study group. Furthermore if K is too large, it may not be possible to find any matching. The following issue may also arise if K too large. Let $K^{'}/K = a$ be a value > 1. Given s, t, u and v, let $RR(s,t,u,v)$ be its RR factor. Assume we find another matching in the same study and control group with the matching ratio $K^{'}$. In this scenario, it is expected that the group's matched results will be similar to those listed in Table 2 when the value of a is not too large. Then, the p-value of the matching found with the matching ratio, $K^{'}$ tends to be smaller than the case where K is the matching ratio assuming the same RR value is observed, because p-value$(s,t,u,v) > p$-value$(s,t,a\cdot u, a\cdot v)$ though $RR(s,t,u,v) = RR(s,t,a\cdot u, a\cdot v)$. To derive a meaningful result, it is important to determine a suitable matching ratio.

Various methods can be used to solve the above matching problem, e.g., "Simple Match" [16] in SAS, "MatchIt" [12, 13] in R and "Optmatch" [11] in R. However when K is large, these methods require a substantial amount of time or memory for computation. When K is large, some methods based on simple approaches, such as "Simple Match", fail to obtain a matching even though such a matching exists. It has been suggested that K should be set at 4 or 5 [**?**, 26], but some methods [11, 12, 13, 16] are based on simple greedy algorithms. As a result, if the matching is only performed once with a small value of K, the result may not be stable in the sense that different matchings may yield different RR values, as shown by our experiment results. To obtain a reliable result, it is better to match multiple times, and then take an average of all the outcomes. Hence, the randomness quality of the derived matching needs to be high. However, when the input size is large, some packages are too resource consuming to compute even once. To resolve the above problems, we propose an efficient matching algorithm with a high quality of randomness. We also introduce a method to determine the best value of K.

The remainder of this paper is organized as follows. In Section 2, we describe the notations and methodology; and in Section 3, we compare the performance of the proposed algorithm with that of existing algorithms. Section 4 contains our concluding remarks.

2 Methods

We begin by formally defining the matching problem studied in this paper. Then, we introduce the naive approach used by previous studies; explain how to transform the matching problem into a graph-theoretical maximum flow problem; and present our maximum flow problem for the transformed flow problem.

2.1 Problem Formulation

K-Regular Matching Problem	
Input:	1) study cases S, 2) control candidates C, and 3) a matching ratio K which is a positive integer.
Output:	$\forall s \in S$, Matched$[s] \subset C$, $\lvert$Matched$[s]\rvert = K$ or 0, $\forall c \in$ Matched$[s]$, CanMatch(c, s) is *true* and Matched$[s_1] \cap$ Matched$[s_2] = \varnothing$ for all $s_1, s_2 \in S$

Procedure CanMatch(s, c)

Input : Study case s, Control candidate c
Output : c can be matched with s or not
Generate conditions $Q_1, Q_2, \ldots, Q_n$ of s;
if $\exists Q_i$ such that c does not satisfy **then**
  return *false;*
else
  return *true;*
end

A derived matching is called *a legal K-regular matching*, which is *full* if $\forall s \in S$ $\lvert$Matched$[s]\rvert = K$. If it is not full, it is a *partial* matching. A case s is *dropped* if $\lvert$Matched$[s]\rvert = \varnothing$. We will formally define the *quality* of a K-regular matching algorithm later. Intuitively, if we run the algorithm multiple times, then a control candidate should be matched with a matchable case with roughly equal probability in all of the derived matchings if the quality is high.

The procedure CanMatch checks whether the input pair (s, c) can be matched. The condition of CanMatch depends on the user's requirements, for example, the range of the birth-year, the gender and whether or not the study case s has been matched already. Note that the α index date of s must be earlier than the β index date of c.

2.2 A Naive Greedy Matching Algorithm

The most popular method, Simple Match, uses an algorithm similar to Algorithm 1, which comprises two procedures: RandomPermute and CanMatch. The RandomPermute procedure (Procedure RandomPermute) randomly permutes the order of the input set.

2.3 Using Graph Maximum Flow

We transform our K-regular matching problem into a variation of the well-known maximum flow problem in graph theory [8]. We first describe the original max flow problem and then introduce its variation.

Algorithm 1: A Naive Greedy matching Algorithm

Input : Study cases S, control candidates C, $\forall s \in S$, $c^{'} \subset C$ such that s can match each control in $c^{'}$, matching ratio K.
Output : A K-regular matching

```
R ⇐ ∅;
for each c ∈ C do
    mark c as unselected;
end
for each s ∈ S do
    Match[s]⇐ ∅;
end
RC ⇐ RandomPermute(C);
for each c ∈ RC such that c is not selected before do
    RS ⇐ RandomPermute(S);
    for each s ∈ RS do
        if CanMatch(s, c) and |Match[s]| < K then
            Match[s] = Match(s) ∪{c};
            mark c as selected;
            break;
        end
    end
end
for each s ∈ S do
    if Match[s].size < K then
        Match[s] = ∅;
    end
end
return ∪_{s∈S} Match[s];
```

Procedure RandomPermute(D)

Input : A list of data D
Output : D in random order

```
for i = 1 to |D| do
    j = UnifromRandom(i, |D|);
    swap(D_i, D_j);
end
return (D);
```

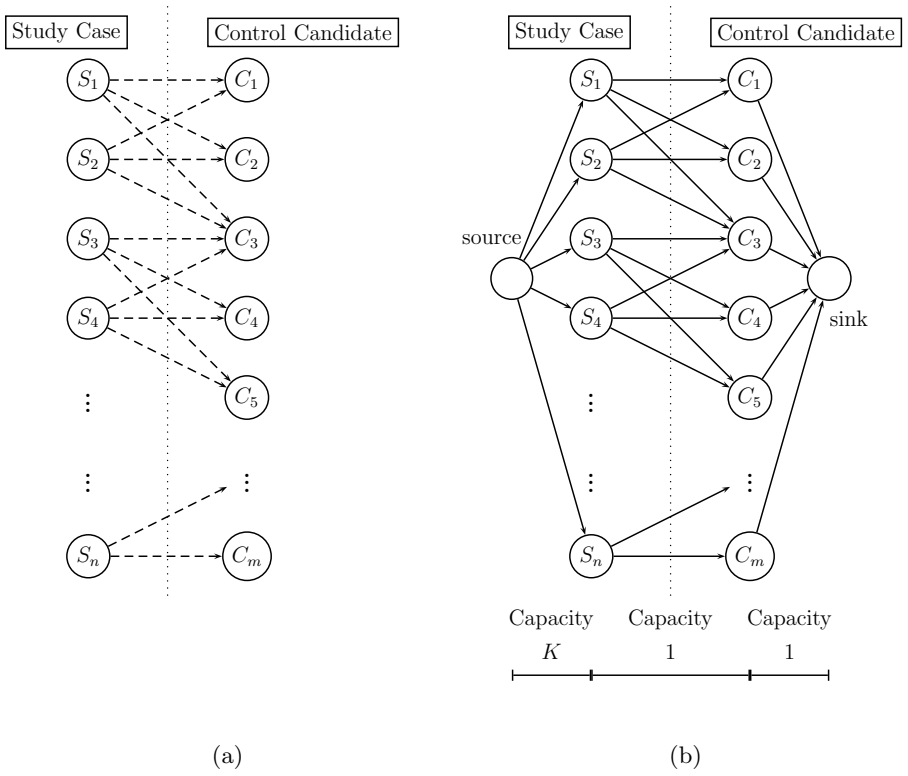


Figure 1: An example of transforming the matching problem into a flow problem.

2.3.1 Maximum Flow Problem

Definition 2.1. Let $G(V, E)$ be a *Directed Acyclic Graph* (DAG) with vertex set V and edge set E. There are two special nodes in G, namely, the *source* s and the *sink* t. The source node does not have an in-going edge and the sink node does not have an out-going edge. The capacity of an edge (u, v), $c_{u,v}$, is the maximum amount of flow that can pass through it. A *legal flow* F is a mapping $F : E \Rightarrow R^+$, where 1) for each edge $(u, v) \in E$, the amount of flow $f_{u,v} \leq c_{u,v}$; and 2) for each node $v \in V \backslash \{s, t\}$, $\sum f_{u,v} = \sum f_{v,w}$. The *value* of a flow F, $|F|$, is defined as $\sum_{u:(s,u)\in E} f_{s,u}$.

The *residual graph* of a legal flow F is the DAG with the capacity in each edge of F being substrated from its capacity. When the legal flow is a path from s to t then it is an *augmenting path*.

The maximum flow problem [8] is defined as follows.

Maximum Flow Problem	
Input:	A DAG $G(V, E)$ with source s and sink t, and each edge (u, v) has a capacity value $c_{u,v}$.
Output:	A legal flow F^* where $\|F^*\|$ is maximum among all legal flows.

The most famous algorithm for solving the maximum flow problem, Ford-Fulkerson algorithm [8], uses DFS on the residual graph of the current legal flow F, whose initial value is all zero, found to find an augmenting path P. F then is added with P and a new residual graph is generated. This procedure continues until the generated residual graph has no augmenting path. The time complexity is $O(|E||F^*|)$, where $|E|$ is the number of edges and $|F^*|$ is the value of the maximum flow. Note that either in theory or in practice [9], there are faster algorithms for solving the maximum flow problem. In practice, the Ford-Fulkerson algorithm is simple and good enough in terms of time and space for our application [8].

2.3.2 Problem Transformation

We use the following steps to transform the K-regular matching problem into the Maximum Flow problem.

1. We first build a DAG G with a source and sink, and assign capacity K to all edges from the source to the case nodes, and capacity 1 to all other edges, as described in Procedure GenerateGraph.

2. Find a maximum flow F^* in G.

3. Convert a flow into a K-regular matching by keeping only the case nodes with the flow values K from the source as described in procedure Flow-ToMatching.

Node S and Node C denote the source node and sink node in the flow network respectively. Figure 1a shows the original matching problem in the case-control study, and Figure 1b shows the transformed max-flow problem.

Procedure GenerateGraph(Node *source*, Nodes S, Nodes C, Node *sink*, Matching Ratio K)

Input : Nodes *source*, S, C, *sink*
Output : Graph (V, E)
V$\Leftarrow \{source, sink\} \cup S \cup C$;
E$\Leftarrow \varnothing$;
for each study case $s_i \in S$ **do**
  add an edge $(source, s_i)$ to E with the capacity K.
end
for each control case $c_j \in C$ **do**
  add an edge $(c_j, sink)$ to E with the capacity 1.
end
for each (s_i, c_j) pair in $\{s_i\} \times \{c_j\}$ **do**
  if *CanMatch*(s_i, c_j) **then**
    add an edge (s_i, c_j) to E with the capacity 1
  end
end

Note that to generate a matching with a high randomness quality, we preprocess the input graph to randomly shuffle it as shown in Algorithm 3.

Algorithm 2: Flow(S, C, K)

Input : Study cases S, control candidates C, Matching ratio K
Output : Max flow F of the input flow network defined by S, C and K
G$\Leftarrow$GenerateGraph(source, S, C, sink, K);
$i \Leftarrow 0$;
for *i from 1 to* $|S|$ **do**
  while *there is an augmenting path P from source to* s_i *to sink* **do**
    update the flow network by subtracting the least capacity used in edges of P;
  end
end

In the Procedure GenerateGraph, if we run the RandomPermute procedure for both set S and set C before we generate the edge set, the output graph of Procedure GenerateGraph becomes a *random shuffled* graph. In this case, for any deterministic flow algorithm, we can get a random matching by randomly shuffling the input graph. The matching method "optmatch" is different from what is described in Algorithm 3. This method does not contain any randomness. When the given matching ratio is less than the maximum matching ratio, "optmatch" always finds a maximum matching by the min-cost flow method. However, "optmatch" uses a deterministic method, so the matching result is always the same without randomness. The steps of the procedure are shown in Algorithm 3.

Algorithm 3: Flow1(S, C, K) Max Flow with Random Graph Shuffling

Input : Study cases S, Control candidates C, Matching Ratio K
Output : A K-regular matching
$RS \Leftarrow$ RandomPermute(S);
$RC \Leftarrow$ RandomPermute(C);
F$\Leftarrow$ Flow(RS, RC, K); `// Algorithm 2`
return *FlowToMatching(F);*

Procedure FlowToMatching(S, E, F, K)

Input : Study cases S, Edge sets E, A legal flow F, and Matching Ratio K
Output : Matching R transformed from F
for *each* $s \in S$ **do**
  if $f_{source,s} < K$ **then**
    remove the flow from source to S;
  end
end
for *each* $s \in S$ **do**
  Match[s] $\leftarrow \varnothing$;
end
for *each* $s \in S$ **do**
  for *each edge (s, c)* $\in E$ **do**
    Match[s] = Match[s] $\cup\{c\}$;
  end
end
return R;

Using the graph generated by Procedure GenerateGraph, we successfully transformed the matching problem into a maximum matching problem. If every edge (S, s_i) in the resulting flow is saturated, the flow represents a matching with matching ratio K. If some of the edges (S, s_i) in the resulting flow are not saturated, we cannot find a matching with matching ratio K. In this case, the maximum possible matching ratio without any dropped study case will be less than K.

2.4 The Proposed Matching Algorithm

The approach discussed in Section 2.3 is limited in terms of the quality of randomness. We will consider this issue with the experiment results in Section 3. Here, we describe our algorithm in details.

Algorithm 4: Flow2(S, C, K) /* Our Matching Algorithm */

Input : Study case S, control candidates C and matching ratio K
Output : A K-regular matching
begin
  E $\Leftarrow$ GenerateGraph(source, S, C, sink, K);
  path $\Leftarrow \varnothing$;
  while *RDFS(source, path)* **do**
    `// There is still a path from source to sink, pick a random one`
    update the flow network by subtracting the least capacity used in edges of *path*;
    path $\Leftarrow \varnothing$;
  end
  return *FlowToMatching(F);*
end

2.4.1 Adding a random augmenting flow

In Algorithm 4, the matching algorithm is implemented with a *Random Depth-First Search* (RDFS) function that finds an augmenting path at random. For a directed edge e, (e_x, e_y), e_x and e_y represent the head and tail of the edge respectively; while c_e, c_{e_x,e_y}, denotes the current capacity of the edge. In RDFS, the RandomChooseEdge function randomly selects an edge according to the current available capacity. For example, in Figure 1a, if the matching ratio is K, then at the beginning of the matching, each edge (S, s_i) has capacity $c_{S,s_i} = K$. Hence, the probability that edge (S, s_i) will be chosen as the out-edge of node S in the RDFS is equal to $c_{S,s_i} / \sum_{\forall i} c_{S,s_i} = K/(Kn) = 1/n$; i.e., the current capacity of edge (S, s_i) over the total capacity of all out-of S edges. If edge (S, s_1) has been chosen once, the probability that it will be chosen in the next round becomes $c_{S,s_i} / \sum_{\forall i} c_{S,s_i} = (K-1)/(nK-1)$.

Procedure RDFS(x, *path*)

Input : Current node x, Current *path* from the source vertex to x
Output : Return a source-x path in random, $\varnothing$ if there is no path from source to sink

```
if x is sink then
    return path;
end
for each edge (x, y) do
    Visit[y]:=false;
end
while (x,y)=RandomChooseEdge(x) do
    if Visit[y]=true then
        continue;
    else
        Visit[y]:=true;
    end
    path.push((x, y));
    if RDFS(y, path)≠ ∅ then
        return path;
    else
        path.pop;
    end
end
return false;
```

Procedure RandomChooseEdge(x)

Input : Node x
Output : An x out-going edge with a non-zero capacity

```
W = ∑_{∀edge(x,u)} c_{x,u};
D ⇐ ∅;
for each edge (x, u) with c_{x,u} ≠ 0 do
    p_{x,u} = c_{x,u}/W;
    D∪{(x, u), p_{x,u}};
end
Randomly choose an edge e = (x, u) from D with the probability p_{x,u};
return e;
```

2.4.2 Merging Edges

The main factor that affects the running time of Algorithm 4 is the size of the edge set. The size of the original graph is $O(|S| + |C| + |S||C|)$. We use the following edge combining method to reduce the size to $O(|S| + |C| + |S|r)$, where r is the total number of types of control candidates.

In the Procedure GenerateGraph, we check each (s, c) pair individually. Then, for each pair that passes the CanMatch function, we add one edge to the edge set of the output graph. The number of edges between S and C is roughly equal to $O(|S||C|)$. However, if either of the two control cases c_1 and c_2 have the same *characteristics*, which are defined by the applications, any study case s that can be matched with c_1 can be also be matched with c_2. That is, we can group the control cases with the same characteristics together. If there are only r different characteristics, then after grouping the control cases with the same characteristics into one node, the edge size between S and C shrinks from $O(|S||C|)$ to $O(|S|r)$; and the total number of edges in the new graph becomes $O(|S| + |C| + |S|r)$.

2.4.3 Finding the Maximum Matching Ratio

With Algorithm 4, for any given matching ratio K, we can always find a legal full K-regular matching with the given matching ratio if such a matching exists. Next, we consider the *maximum matching ratio problem*, which is defined formally as follows:

Maximum Matching Ratio Problem	
Input:	Study cases S, Control cases C
Output:	Maximum Matching Ratio K^* such there is a full legal K^*-regular matching from S to C

When the given matching ratio K is less than the maximum matching ratio, all the study cases can be matched with K control cases. That is, no study case will be dropped. However, when K is larger than the maximum matching ratio, some study cases will be dropped.

We can use an open binary search algorithm to find the optimal matching ratio. Using the maximum matching ratio as a starting point, we run Algorithm 4 with different matching ratios close to the maximum matching ratio. Then, by examining the derived matching ratios, we determine that the optimal ratio is the one with the lowest standard deviation. Our experiment results show that this approach achieves a very good randomness quality.

2.4.4 Efficient implementation of RandomChooseEdge

To ensure that Algorithm 4 has a good randomness quality, we use the procedure RandomChooseEdge to select a random control candidate. The last statement in RandomChooseEdge takes $O(|D|)$ time and space using a naive linear selection approach, where D is the number of outgoing edges of the input nodes. To improve the procedure, we propose two efficient implementations. In the first,

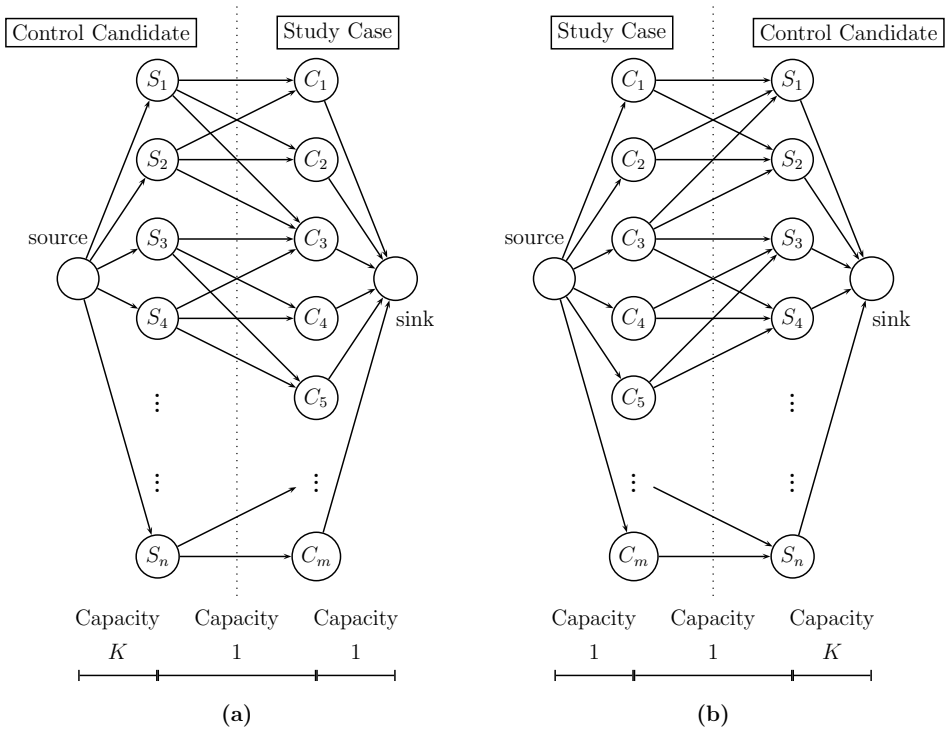


Figure 2: Original flow network and its reversed flow network.

each node uses a segment tree data structure [6] to store the capacity of each of its out-going edges. This method requires $O(|D| \log |D|)$ pre-processing time and $O(\log |D|)$ time for the selection and deletion of each chosen edge. The second modification uses the trick of *the reversed flow network* (shown in Figure 2) to reverse the direction of each edge in the residual network. In the resulting graph, the control candidates can be efficiently matched with the study cases by random selection. Note that in the reversed flow network, only edges from study cases to the sink node have non-binary capacities. The capacities of all the other edges are binary, i.e., 0 or 1. All edges with non-zero capacities can be stored in an array. It then takes $O(1)$ time to randomly select an edge when the capacity of all edges is binary. Hence, we can select an edge efficiently with a prescribed probability in Procedure RandomChooseEdge(x).

3 Experiment Results and Discussion

In this section, we present the experiment results and give discussion.

Table 3: Summary of the original result of testing event pairs.

	Event Pair I	Event Pair II	Event Pair III
Study Cases	27,073	6,427	5,527
Control cases	135,365	32,135	5,527
Matching Ratio	5	5	1

Table 4: Summary of the basic attributes of testing event pairs.

	Event Pair I	Event Pair II	Event Pair III
Study Case	21,647	6,412	5,192
Control Candidates	463,754	619,902	9,102
Maximum Matching Ratio	11	53	0
Total edge	121,121,658	38,629,676	404,835
Average degree(Study)	5595.31	6024.59	77.97
Average degree(Control)	261.18	62.32	44.48

3.1 Experiment Setting

As a benchmark, we use the following three event pairs from the following two exist research: 1) Obstructive sleep apnea and depression whose positive bidirectional relationship has been demonstrated [3]; 2) Statin use and dementia in patients with stroke whose negative relationship has been discovered [1]. The basic information of these three event pairs are shown in Table 4.

In the original study of event pairs I, 27,073 study cases and 135,365 control cases were found and used with a matching ratio of 5. In our database, there are 21,647 study cases with 463,754 control candidates, and the total number of edges is 121,121,658. The average degree of the study cases is 5595.31 and that of the control candidates is 261.18. The degree distribution of the study cases and control candidates is shown in Figure 3a and Figure 3b. The maximum matching ratio of these event pairs is 11.

In the original study of event pairs II, 6,427 study cases and 32,135 control cases were found and used with a matching ratio of 5. In our database, there are 6,412 study cases with 619,902 control candidates, and the total number of edges is 38,629,676. The average degree of the study cases is 6,024.59 and that of the control candidates is 62.32. The degree distribution of the study cases and control candidates is shown in Figure 3c and Figure 3d. The maximum matching ratio of these event pairs is 53.

In the original study of event pairs III, 5,527 study case and 5,527 control cases were found and used propensity score with a matching ratio of 1. In the

[1]Data set from Ph.D. Mei-Lien Pan through private communication.

Table 5: Average execution times (in seconds) of each method.

Method	Language	Algorithm	Matching Ratio 1	5	10	15	20	25	30
MatchIt	R	1	182.81	928.57	1956.58	2890.76	3891.28	4821.20	5924.35
Simple Match	SAS	1	1221.00	959.00	894.00	859.00	2350.00	1228.00	1078.00
Simple Match	C	1	0.51	0.51	0.51	0.51	0.51	0.51	0.51
Optmatch	R	2	8473.28	9132.34	9676.67	9407.05	9605.70	9037.06	8399.11
Flow1	C	1	8.00	7.82	8.00	6.54	6.09	5.64	5.11
Flow2	C	1	6.60	11.91	18.52	20.91	19.10	16.47	13.85

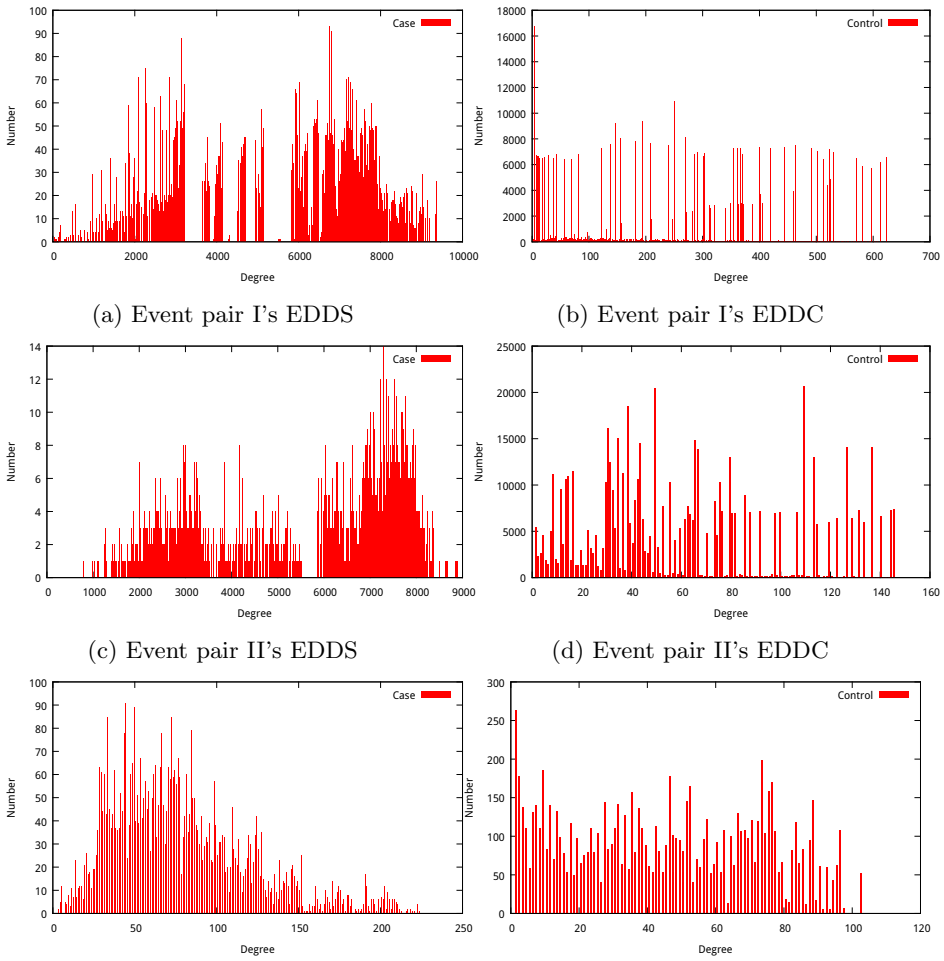
provided data set, there are 5,192 study cases with 9,102 control candidates, and the total number of edge is 404,835. The average degree of the study cases is 77.97 and that of control candidates is 44.48. The degree distribution of the study cases and control candidates is shown in Figure 3e and Figure 3f. The maximum matching ratio of these event pairs is 0.

We test these event 2-sequences with the following five software/programs: "Simple Match" [16] in SAS, "MatchIt" [12, 13] in R, "Optmatch" [11] in R, "Simple Match" in C, and our methods in C. "Simple Match" is based on Algorithm 1. "MatchIt" and "Optmatch" use variations of Algorithm 3 without randomly shuffling the input graph. For the fourth method, we implemented the "Simple Match" methods based on Algorithm 1 using C. Our method is based on Algorithm 4. The experiment assessed the average running time and the corresponding RR ratio with matching ratios from 1 to 30. For methods with some randomness in the output, we ran them 100 times and took the average. The experiments were performed on a Ubuntu 14.04 system with an Intel(R) Core(TM) i7-3770 CPU 3.40 GHz, and 16 Gbytes RAM.

3.2 Results of the Execution Time and Memory Usage

In order to measure the performance of execution time and memory usage, in this part of experiments, we choose Event-pair I, which has the most proper maximum matching ratio as the benchmark.

The average execution times of each method with different matching ratios are shown in Table 5. The first column shows the name of each method; the second column shows the basis algorithm of each method; and columns 3 to 9 summarize the average execution times (in second) of each method with matching ratios of 1, 5, 10, 15, 20, 25 and 30 respectively. Because of software limitations, programs and packages with software SAS and R are much slower than those programs in C. For programs in C, the time complexity of "Simple Match" (Algorithm 1) is $O(nm)$ and that of our method (Algorithm 4) is $O(n^2m)$, where n is the number of nodes and m is the number of edges in the graph. Therefore, it is expected that Algorithm 1 will be faster than Algorithm 4.



(a) Event pair I's EDDS (b) Event pair I's EDDC

(c) Event pair II's EDDS (d) Event pair II's EDDC

(e) Event pair III's EDDS (f) Event pair III's EDDC

Figure 3: Edge degree distribution of study cases (EDDS) and control candidates (EDDC) of all three event pairs.

Table 6: Memory usage of different implementations with matching ratio 1.

Programs/ Languages	"Simple" (SAS)	"Optmatch" (R)	"MatchIt" (R)	"Simple" (C)	Flow2 (C)	Flow2 with edge merging (C)
Memory usage	1.039 Gb	0.277 Gb	0.232 Gb	0.068 Gb	0.152 Gb	0.070 Gb

In Table 6, the memory use of "Simple Match (SAS)" is the largest, this is the language nature of SAS. "Optmatch" and "MatchIt" in R also use more memory than those methods which implements in C. For methods implement in C, Flow2 requires more memory than "Simple Match", but if edge merging technique is applied, these two methods use almost the same amount of memory.

In Figure 4, the running time of "MatchIt" is almost linear in the matching ratio, but the execution times of "Optmatch" and our method increase before the matching ratio reaches the maximum and decrease after it exceeds the maximum. The result suggests that the time consumption is highest when a matching ratio close to the maximum matching ratio is chosen. As the C implementation is more efficient and memory saving, we only make comparisons with C implementations hereafter.

3.3 Successful Matching Rates

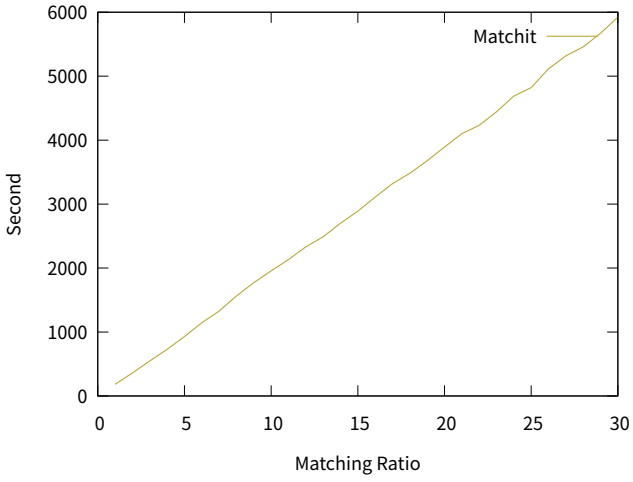
In this experiment, we analyze the *dropping ratio*, i.e., the percentage of cases that are dropped under different matching ratios. In Figure 5, the x-axis indicates the size of the matching; and the y-axis indicates the percentage of matched study cases that are not dropped, which is called the *successful matching rate*. Figure 5 (a) shows that when the matching ratio is less than the maximum matching ratio, "Optmatch" and Flow2 do not drop any cases. The successful matching rate of "Simple Match (C)" decreases smoothly as the matching ratio increases, and it starts to decrease before the matching ratio reaches the maximum. When the matching ratio is larger than the maximum matching ratio, the matching rates of the three methods are all less than 100%. Flow2 achieves the highest successful matching rate of the three methods.

Figure 5 (b) compares the successful matching rates of Flow2 and "Simple Match (C)". In the execution time experiment, "Simple Match (C)" was much faster than Flow2, so we compare the successful matching rate in terms of the number of executed trials. For "Simple Match (C)" we compute the average matching rate and the maximum matching rate for 100 trials. The results show that Flow2 still achieves the highest matching rate among all the matching ratios, even if we compare it with the best result of "Simple Match (C)" in 100 trials. Although "Simple Match (C)" is faster than Flow2, when the execution time is fixed, it cannot achieve the successful matching rate of Flow2.

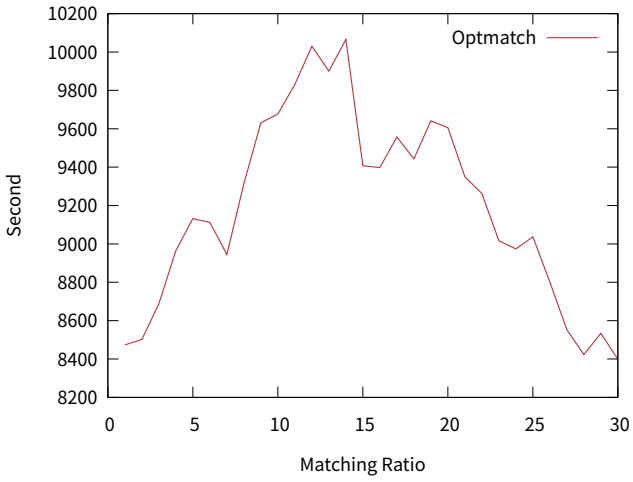
The results of Event Pair II and Event Pair III are shown in Figure 5(c)-(f). The behavior of all three methods are similar with the results of Event Pair I.

3.4 RR Ratio

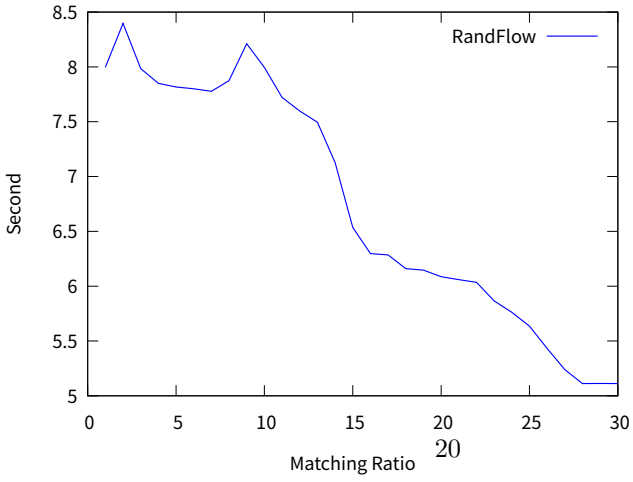
In this experiment, we compare the distribution of RR values in "Simple Match (C)" and Flow2. Figure 6 (a) shows the average RR ratio of "Simple Match (C)" and Flow2 using different matching ratios after performing the experiment 100 times. Both methods have a similar average RR value when the matching ratio is less than the maximum matching ratio 11. When the matching ratio is larger than the maximum matching ratio, the average RR value decreases sharply.



(a) Matchit

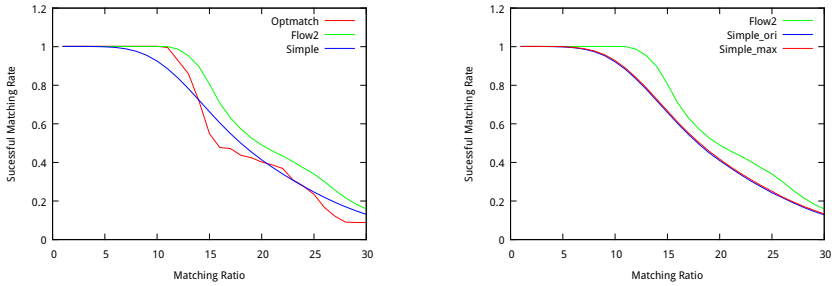


(b) Optmatch



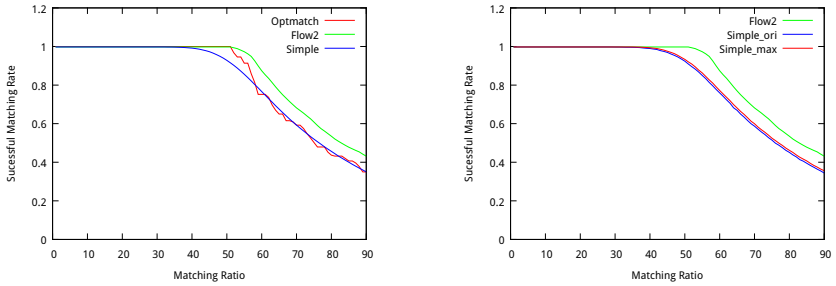
(c) Flow2

Figure 4: Execution times of some selected methods.



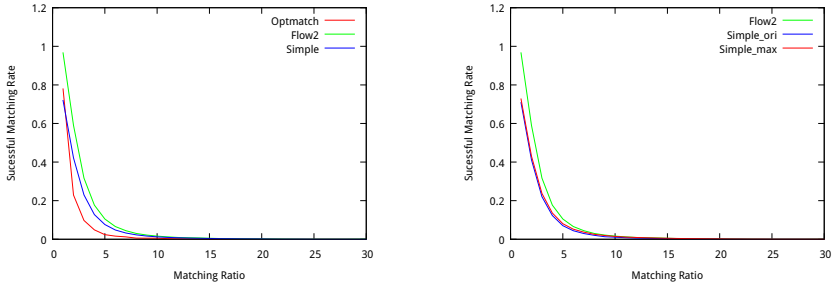
(a) Event Pair I

(b) Event Pair I



(c) Event Pair II

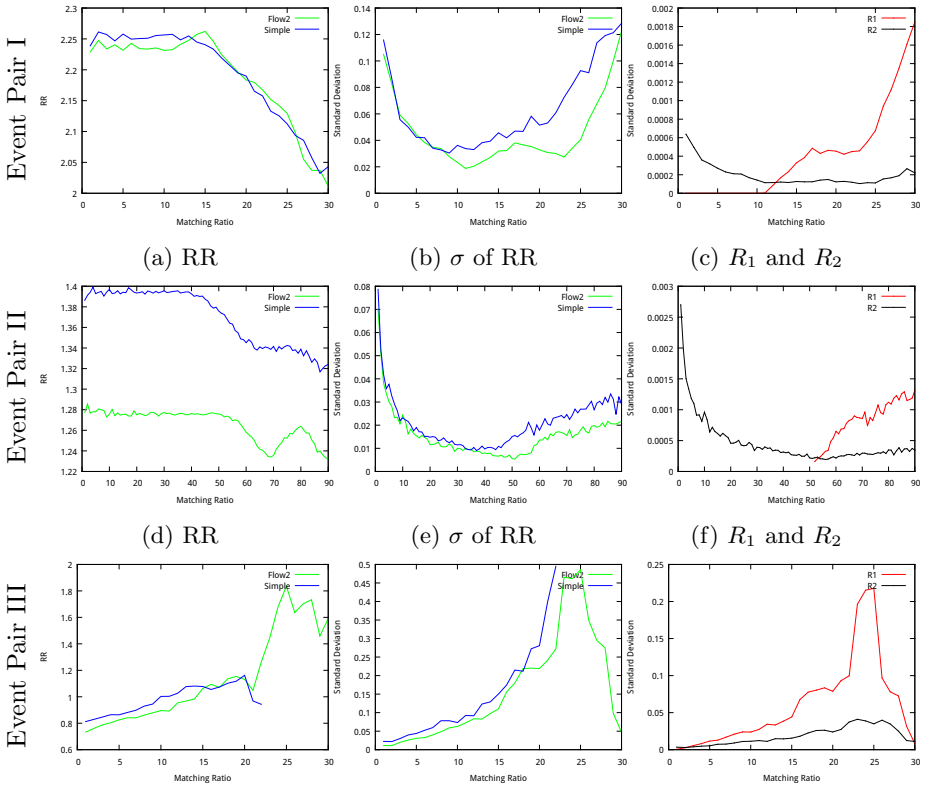
(d) Event Pair II



(e) Event Pair III

(f) Event Pair III

Figure 5: Successful matching rates of “Optmatch”, “Simple Match” and Flow2.



(a) RR (b) σ of RR (c) R_1 and R_2

(d) RR (e) σ of RR (f) R_1 and R_2

(g) RR (h) σ of RR (i) R_1 and R_2

Figure 6: Comparison of the R_1, R_2 and RR values of Algorithm 1 (Simple Match) and Algorithm 4 (Flow2).

In Figure 6 (b), we analyze the relationship between the average RR ratio and the matching ratio. Specifically, we compare the standard deviation of the RR ratios of "Simple Match (C)" and Flow2 with different matching ratios. When the matching ratio is 11, i.e., the maximum matching ratio, the average RR ratio has the minimum standard deviation for Flow2 and a relative smaller standard deviation for "Simple Match (C)".

By checking the standard deviations of R_1 and R_2, we believe the reason that RR has a smaller standard deviation when the matching ratio is close to the maximum is as follows. Figure 6 (c) shows the standard deviations of R_1 and R_2 against different matching ratios. As the maximum matching ratio is 11, no study case will be dropped when the matching ratio is less than 11; thus, the standard deviation of R_1 remains 0. When the matching ratio is larger than the maximum, the standard deviation of R_1 increases and becomes unstable. On the other hand, the standard deviation of R_2 decreases when the matching ratio increases until it reaches the maximum. After the matching ratio reaches the maximum, the standard deviation of R_2 becomes stable. If the size of the control group is small, the variation of the group will be large. However, when the size increases to a certain number, the variations of control group in different matching trails will become relatively small.

The same behaviors can be observed from event pairs II and III as shown in Figure 6 (d) through (i).

The result of this experiment shows that if we want to have a high quality and relatively stable matching result for the average RR value, the matching ratio should be chosen around the maximum matching ratio. Hence, it is very important to find out the maximum matching ratio before carrying out the matching.

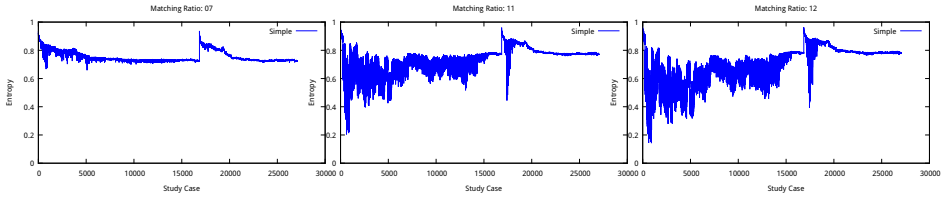
3.5 Quality of Randomness

In this experiment, we analyze the quality of the randomness of "Simple Match (C)" and Flow2 by calculating the entropy of the distribution of the matched control candidates.

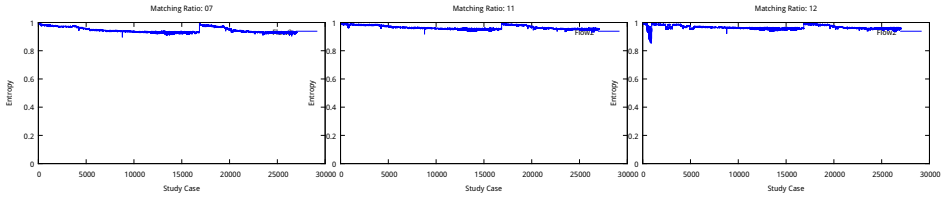
Definition 3.1. For a matching method with a matching ratio of 1 and a fixed study case s, the method is executed N times and we record the N control cases, c_1, c_2, ..., c_N that match with s in the i-th round. Then, the N control candidates form a *control case population*. The event set of a control case population comprises the unique cases over all c_i and the corresponding probability of each event is the occurrence frequency in the N matchings.

If a study case has 100 different control candidates in the control case population, then it has *the maximum possible entropy* $\sum -\frac{1}{100} \log \frac{1}{100}$ according to the entropy formula [21].

Definition 3.2. The *quality of randomness* is defined by the relative ratio of the entropy of the control cases distribution over the maximum possible entropy value.

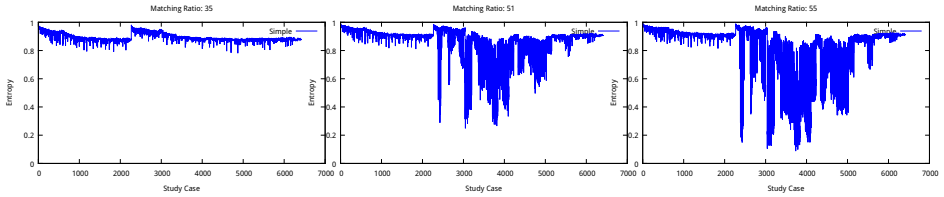


(a) Simple method (70%) (b) Simple method (100%) (c) Simple method (110%)

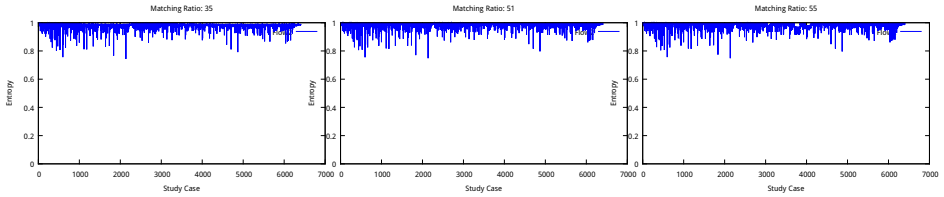


(d) Flow2 (70%) (e) Flow2 (100%) (f) Flow2 (110%)

Figure 7: Comparison of "Simple Match (C)" and Flow2's relative entropy with matching ratio as 70%, 100% and 110% of maximum matching number in Event Pair I.



(a) Simple method (70%) (b) Simple method (100%) (c) Simple method (110%)



(d) Flow2 (70%) (e) Flow2 (100%) (f) Flow2 (110%)

Figure 8: Comparison of "Simple Match (C)" and Flow2's relative entropy with matching ratio as 70%, 100% and 110% of maximum matching number in Event Pair II.

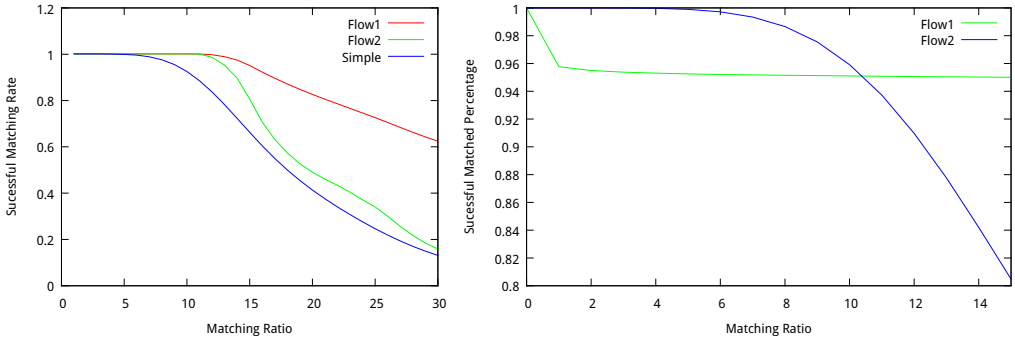
We compare the entropy between "Simple Match (C)" and Flow2 with three different matching ratios: 70%, 100% and 110% of the maximum matching ratio.

We start with 70% of maximum matching ratio, 7. In Figure 7 (a), the x-axis indicates each study case. There are 21,647 study cases. The y-axis represents the entropy of the distribution of matched control candidates. In the experiment, we execute "Simple Match (C)" 100 times. By calculating the probability that a control candidate will be chosen in 100 matches, we can compute its relative entropy. In Figure 7 (a), the entropy of most of the study cases is around 0.8, which means the quality of randomness of "Simple Match (C)" is not bad. However, in Figure 7 (d), the entropy of most of the study cases is large than 0.9, which suggests the quality of randomness of Flow2 is better than "Simple Method (C)". Note that some study cases have low entropy because they can only be matched to a very small set of candidates. As the set is relatively small, the corresponding entropy is also small.

Next, we compare the relative entropy between "Simple Match (C)" and Flow2 with 100% of maximum matching ratio, 11. In Figure 7 (b), most of the study cases have relative entropy less than 0.8, which means the quality of randomness of "Simple Match (C)" becomes worse as matching ratio increases. On the other hand, in figure 7 (e), the relative entropy of each study cases remain the same when matching ratio changes from 7 to 11. Now we increase the matching ratio to 110% of maximum matching ratio, 12, and compare the relative entropy between "Simple Match (C)" and Flow2. In Figure 7 (c), most of the study cases has the similar or smaller relative entropy comparing with Figure 7, which suggests that when matching ratio increases, the relative entropy of study cases in "Simple Match (c)" decreases consistently. In Figure 7 (f), the relative entropy of study cases also decreases as the matching ratio increases. But most of the study cases still have high relative entropy which is close to 1. That is, when matching ratio is greater than 100% of maximum matching ratio, the quality of randomness of Flow2 is still good.

In Figure 8, we show the relative entropy of "Simple Method (C)" and Flow2 with Event Pair II using the similar setting as Event Pair I in Figure 7, that is, 70%, 100% and 110% of maximum matching ratio. The conclusion is similar to Event Pair I. The relative entropy of each study cases in "Simple Methods (C)" starts to decrease when the value of matching ratio exceeds 70% of the maximum matching ratio, and consistently decreases as the value of matching ratio increases. And the relative entropy of each study cases in Flow2 is around 1 even if the value of matching ratio exceeds 110% of the maximum matching ratio. As a result, it is clearly that Flow2 has better quality of randomness than "Simple Method(C)" in all cases.

Now we compare the quality of randomness of Flow1 (Algorithm 3) and Flow2 (Algorithm 4). We use the result of Algorithm 1 as the baseline for comparison. In the matching phase of Algorithm 3, a study case is repeatedly matched with control candidates until the number of matched controls has reached the required number of matches. In Algorithm 3, the matching order of study cases is determined by randomly permuting the vertices of the input graph. The probability of a study case being successfully matched to the required number of



(a) Successful matching rate. (b) Cumulative successful matching rates.

Figure 9: Comparison of successful matching rate and cumulative successful matching rate.

control cases depends to a large extent on the matching order. That is, after the random permutation, study cases with a smaller index have a higher probability of matching the required number of control cases than study cases with a larger index. By contrast, in the reversed flow network with random DFS, a control candidate has equal probability of being matched with all study cases that it can be matched with. In Figure 9a, Flow1 has the most successful matching rate. However, as described above, the distribution of cases successfully matched by Algorithm 3 is not uniform. The successful matching rate of Algorithm 4 is lower than that of Algorithm 3, but the distribution of successfully matched study cases is more uniform. Hence, we conclude that the quality of randomness of Algorithm 4 is better than that of Algorithm 3.

In Figure 9b, the x-axis denotes the matching ratio and the y-axis denotes the percentage of study cases matched to more than K control cases. When the matching ratio is set at 15, if we compare the percentage of study cases whose successful matching number is 15, then Flow1 has a higher successful matching rate than Flow2. However, if we only count study cases with more than 10 matched control cases, then Flow2 has a higher successful matching rate than Flow1. Hence Flow2 tends to match a case with more types of control candidates when it is run multiple times. In this scenario, Flow2 has a better quality of randomness than Flow1.

4 Concluding Remarks

In this paper, we present an efficient algorithm with good quality of randomness to solve the K-regular matching problem. The computation time is much faster than that of existing programs. Moreover, the successful matching rate of the algorithm is 100% when the matching ratio is lower then the maximum matching ratio. When the matching ratio required is larger than the maximum matching

ratio, the successful matching rate of the proposed algorithm is higher than or equal to that of other approaches. Moreover, the algorithm has an RR ratio that is close to that obtained by Algorithm 3, but with a relatively smaller standard deviation. Hence, our algorithm produces relatively stable matching results. We have also shown that the randomness quality of our method is similar to that of "Simple Match (C)". Finally, the experiment results suggest that to make the matching results reliable, the size of the matching ratio should be set to be around the theoretical maximum matching ratio.

References

[1] National health insurance research database, Taiwan. URL: `\url{http://nhird.nhri.org.tw/en/index.htm}`.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[3] M.-L. Pan andH. M. Tsao, C.-C. Hsu, K.-M. Wu, T.-s Hsu, Y.-T. Wu, and G.-C. Hu. Bidirectional association between obstructive sleep apnea and depression: A population-based longitudinal study. *Medicine*, 95(37):e4833, 2016.

[4] Yu-Chun Chen, Jau-Ching Wu, Ingo Haschler, Azeem Majeed, Tzeng-Ji Chen, and Thomas Wetter. Academic impact of a public electronic health database: bibliometric analysis of studies using the general practice research database. *PloS one*, 6(6):e21404, 2011.

[5] S.D. Chung, C.C. Lin, J.D. Ho, J. Ting, H.C. Lin, and C.C. Hu. Increased risk of open-angle glaucoma following chronic rhinosinusitis: a population-based matched-cohort study. *Eye*, 28(2):225–230, 2013.

[6] P. M. Fenwick. A new data structure for cumulative frequency tables. *Software: Practice and Experience*, 24(3):327–336, 1994.

[7] R. A. Fisher. On the interpretation of χ^2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, 85(1):87–94, 1922.

[8] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[9] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[10] D. A. Grimes and K. F. Schulz. Compared to what? finding controls for case-control studies. *The Lancet*, 365(9468):1429–1433, 2005.

[11] B. B. Hansen. Full matching in an observational study of coaching for the sat. *Journal of the American Statistical Association*, 99(467):609–618, 2004.

[12] D. E. Ho, K. Imai, G. King, and E. A. Stuart. Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. *Political analysis*, 15(3):199–236, 2007.

[13] D. E. Ho, K. Imai, G. King, and E. A. Stuart. Matchit: Nonparametric preprocessing for parametric causal inference. *Journal of Statistical Software*, 2007.

[14] G. Hripcsak and D. J. Albers. Next-generation phenotyping of electronic health records. *Journal of the American Medical Informatics Association*, 20(1):117–121, 2013.

[15] Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In *ACM Sigmod Record*, volume 27, pages 85–93. ACM, 1998.

[16] H. Kawabata, M. Tran, and P. Hines. Using SAS® to match cases for case control studies. In *Proceeding of the Twenty-Ninth Annual SAS® Users Group International Conference*, volume 29, pages 173–29, 2004.

[17] Wei-Hung Lin, Chih-Hui Hsu, Hua-Fen Chen, Chi-Chu Liu, and Chung-Yi Li. Mortality of patients with type 2 diabetes in Taiwan: A 10-year nationwide follow-up study. *Diabetes research and clinical practice*, 2014.

[18] Debprakash Patnaik, Patrick Butler, Naren Ramakrishnan, Laxmi Parida, Benjamin J. Keller, and David A. Hanauer. Experiences with mining temporal event sequences from electronic medical records: initial successes and some challenges. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 360–368. ACM, 2011.

[19] Karl Pearson. *On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling*, pages 11–28. Springer New York, New York, NY, 1992.

[20] P. R. Rosenbaum and D. B. Rubin. Constructing a control group using multivariate matched sampling methods that incorporate the propensity score. *The American Statistician*, 39(1):33–38, 1985.

[21] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[22] C. L. Sistrom and C. W. Garvan. Proportions, odds, and risk. *Radiology*, 230(1):12–19, 2004.

[23] Ramakrishnan Srikant and Rakesh Agrawal. *Mining sequential patterns: Generalizations and performance improvements*. Springer, 1996.

[24] Chin-Hsiao Tseng. Diabetes but not insulin increases the risk of lung cancer: A Taiwanese population-based study. *PloS one*, 9(7):e101553, 2014.

[25] H. K. Ury. Efficiency of case-control studies with multiple controls per case: continuous or dichotomous data. *Biometrics*, pages 643–649, 1975.

[26] S. Wacholder, D. T. Silverman, J. K. McLaughlin, and J. S. Mandel. Selection of controls in case-control studies: Iii. design options. *American journal of epidemiology*, 135(9):1042–1050, 1992.

[27] Fang-Jen Wu, Shiow-Yunn Sheu, and Herng-Ching Lin. Osteoporosis is associated with antiepileptic drugs: a population-based study. *Epileptic Disorders*, 16(3):333–342, 2014.

[28] F. Yates. Contingency tables involving small numbers and the χ^2 test. *Supplement to the Journal of the Royal Statistical Society*, 1(2):217–235, 1934.