An Improved Method for Transaction Footprints Stripping with Bigraph System

Changyun Li \(^1,2\)

Email: lcy469@163.com

1School of Computer Science, National University of Defense Technology, Changsha 410073, China
2School of Computer and Communication, Hunan University of Technology, Zhuzhou 412008, China

Junfeng Man, Zhibing Wang

School of Computer and Communication, Hunan University of Technology, Zhuzhou 412008, China

Abstract—Multiple parallel transactions in new-type distributed software environment result in that the events produced by every transaction are randomly ranked. If the tokens of these events are incomplete or unavailable, it is difficult for software system to distinguish these events to actually belong to which transaction, corresponding transaction analysis and prediction can’t be executed. In this paper, the problem of stripping events with incomplete tokens is transferred into maximum-weight perfect matching of bigraph system. If the transition time among these events is independently and identically distributed, all possible states (events) are separated into multiple cutsets, every cutset composes a bigraph system. The maximum-weight perfect matching is used to finish respective matching, and then the results of independent matching of multiple bigraph systems are spliced to gain the most possible footprint sequences produced by multiple transactions, which is convenient for subsequent analysis and prediction. For implementing quick stripping for transaction footprints, the paper presents rank-maximal matching algorithm to improve matching efficiency. Simulation experiment confirms that the method presented in this paper can effectively implement transaction footprint stripping with incomplete tokens. Compared to other methods, the rank-maximal matching algorithm has higher matching efficiency and lower time cost.

Index Terms—incomplete token, transaction footprint, bigraph matching, maximum-likelihood rule, rank-maximal matching

I. INTRODUCTION

In new-type distributed software, a solution to end-to-end transaction monitoring and analyzing comprises of four pieces: (a) discovery of IT artifacts, such as servers and applications on which the transaction depends, (b) modeling of relationships among these IT artifacts in the context of the transaction, (c) monitoring of IT artifacts to draw conclusions regarding the status of a transaction, and (d) creditability analysis for transaction footprints [1]. Each of these pieces will pose different challenges depending on the degree of information and instrumentation available in the system. In general condition, we may use industry standards such as the open-group ARM instrumentation [2] to generate transaction correlators or tokens that may be used to track the flow of transactions. In the process of software interaction, the event tokens that some software entities produce are incomplete or unavailable. Unfortunately, only a small number of footprints may contain tokens, an instance is illustrated in Fig.1. Other than the footprints at state \(S_0\), none of the other footprints contain tokens. In such cases, it may not be possible to identify the unique source of each footprint with certainty. Except for simple cases such as a strictly ordered process scheduling like first-in-first-out (FIFO) or last-in-first-out (LIFO), the system may splice these events to compose a complete footprint sequence. In distributed multi-thread environment, multiple parallel transactions result in that the events produced by every of them are randomly ranked. If these event tokens are incomplete or unavailable, they can’t be distinguished to belong to which transaction, which results in that the system can’t analyze and predict transaction. Thus, an efficient method should be found to strip randomly ranked events, namely, mark these events, and then splice them to compose complete transaction footprints. For simple (without repeated sub-footprints) transaction footprints with incomplete tokens, it is a key that finds a kind of efficient and accurate stripping technology.

![Figure 1. The transition model instance with incomplete tokens.](image_url)
transaction footprints when monitoring them, (b) for multiple transaction footprint samples, analyze their correlation and construct transfer matrix, (c) by tracking a set of transaction instances, find the most likely sequence of states visited by each of these transaction instances and estimate the times spent in these states by every instance. We take a probabilistic approach by incorporating the available (statistical) information about transition time between different states. In our approach, optimal tracking refers to using the maximum-likelihood rule (MLR) that maximizes the probability that all the footprints are correctly matched to the transactions that generated them. For a two-state system, under Independent and Identically Distributed (I.I.D.) transition time, optimal MLR is reduced to a maximum-weight perfect matching of bipartite graph (or bigraph).

For maximum-weight perfect matching of bigraph, many researchers have studied according algorithm. The fastest algorithms have running time \( O(n(m + n \log n)) \) [3] and \( O(\sqrt{n}m \log nC) \) [4] where \( C \) is the maximum weight in an instance. The first is strongly- while the second weakly-polynomial. Simply applying the aforementioned algorithms to the rank-maximal matching problem does not result in efficient effects, either in time or in space requirements. The problem is that the edge weights are as large as \( n' \), which is non-polynomial in the input size. Both algorithms assume that arithmetic operations between numbers which are in \( O(C) \) can be performed in constant time. This is not true in this case, where arithmetic on numbers in \( O(n') \) takes time \( \Omega(r) \). Hence, the running times are \( O(rn(m + n \log n)) \) and \( O(r^2\sqrt{n}m \log n) \) respectively, both using \( O(rn) \) space. It is known, however, that the scaling algorithm for the weighted matching problem can be implemented such that all algorithms are performed on numbers with \( O(\log n) \) bits, independent of the edge weights. In this case the running time improves to \( O(r\sqrt{n}m \log n) \). In [5] the authors present a combinatorial algorithm which solves the rank-maximal matching problem in \( O(\min(n + r, r\sqrt{n})m) \) time using linear space. The algorithm identifies edges which cannot be part of a rank-maximal matching and deletes them. This approach, however, does not seem to generalize to the maximum cardinality rank-maximal or the fair matching problem. In an attempt to close the gap between the rank-maximal matching and its variants, we present an algorithm which solves the rank-maximal matching problem in the same running time and space as [6]. The main difference is that our algorithm is based on weight matching reduction. We believe that our algorithm is simpler and more intuitive.

II. TOKENS MODELING FOR TRANSACTION FOOTPRINTS

A. The Concept and Definition

Let \( f_x(x) \) be the probability density function (PDF) of a continuous random variable \( X \) and \( \bar{F}_x(x) := P[X > x] \) its complementary cumulative distribution function (CCDF). For a matrix \( A \), let \( A(i, j) \) denote the element in its \( i^{th} \) row and \( j^{th} \) column. Let \( \pi \) denote a permutation vector over \( \{1, ..., n\} \), \( \log x \) the natural logarithm of \( x \) and \( |A| \) the cardinality of a set \( A \). For sets \( A \) and \( B \), let \( A \setminus B = \{i, i \in A, i \notin B\} \).

For an undirected graph \( G(V, E) \), let \( (i, j) \) denote the edge between \( i \) and \( j \) and \( \deg(i) \) be the degree of node \( i \). For a bigraph \( G = (V, V \cup E) \), the edges are represented by a 0-1 biadjacency matrix \( A = [A(i, j)] \), here \( A(i, j) = 1 \) indicates an edge between \( V_i(i) \) and \( V_j(i) \). A matching \( M \subseteq E \) is a set of pairwise non-adjacent edges, i.e., no two edges share a common vertex. A maximum cardinality matching is a matching that contains the largest possible number of edges. A perfect matching is a matching where there is no unmatched vertex and a maximum-weight perfect matching maximizes the sum of the matched edge weights. For a directed graph (digraph), when there is an edge from \( i \) to \( j \), \( j \) is an immediate successor of \( i \), and \( i \) an immediate predecessor of \( j \). The set of all immediate successors of \( i \) is denoted by \( N(i) \), and the set of all immediate predecessors by \( P(j) \).

We now define a semi-Markov process (SMP). Let \( S_i, i = 0, ..., N \) denote the \( i^{th} \) state of the process and let \( T_{ij} \) denote the (random) time to transition from state \( S_i \) to state \( S_j \). A process is said to be semi-Markov if the sequence of states visited is a Markov chain, with transition probability matrix \( P = [P(i, j)] \) and each transition time \( T_{ij} \), is a random variable that depends only on the states \( S_i \) and \( S_j \) involved in the transition. We assume that each transaction progresses through the system according to a general SMP with each transition time \( T_{ij} \) drawn from a known PDF \( f_{T_{ij}} \) having a continuous interval. We make a simplifying assumption that the state transition digraph is directed acyclic graph (DAG). This ensures that all transactions are processed in one direction, and that no transaction can leave more than one footprint at a state. Furthermore, we assume that transition time is independent of system load.

B. Maximum-Likelihood Rule

A footprint is defined as a time-tamped entry created in the application log when a transaction enters a state in the model. In addition to the timestamp, a footprint may optionally contain a unique identifier or a token that ties it to the transaction instance. By convention, the footprints at the (unique) start state \( S_c \) are each assigned a token. We assume that no footprint is missing from the log records. We consider the general case where at the time of observation, transaction instances are still residing at different states of the system, and hence, all the footprints that these transactions will eventually generate are not yet available. Tracking transactions in such cases is affected by the assumption that the records appear in logs as soon as they are written by the applications, i.e., the writing is not buffered.

Any valid match can be represented by the set of permutation vectors \( \pi_i \), for each state \( S_i \) in the model. Let \( Y_j \) be the vector of the timestamps of the footprints at
state $S_k$. When the monitoring engine receives the footprints in the correct temporal order, $Y^S_i$ is in ascending order with the most recent footprint being the last entry. Let $Y^{nS}_i$ be the permutation of $Y^S_i$, according to the permutation vector $\pi_i$. By convention, we assign tokens to footprints $Y_k$ at the start state $S_0$, and hence, the permutation vector at the start states $S_0$ is set to identity ($\pi_0 = I$). In other words, we find the correspondence of all other footprints in the system with respect to the footprints at $S_0$. When the joint PDF of the transaction transition times $f_T$ is known, we can quantify the tracking performance as the probability that all the transaction instances are matched correctly to their footprints and this is maximized by MLR. Hence, for $N+1$ states of the system, the optimal MLR of transactions reduces to finding a set of $N$, number of permutation vectors,

$$\hat{\pi} = \arg \max_{\pi} P(Y_1^{nS}, ..., Y_N^{nS} \mid Y_0)$$

In general, solving (1) is NP-hard. The rest of the paper primarily deals with the special cases, starting with the two state system, where (1) can be solved efficiently.

III. TWO-STATE SYSTEM

We consider a two-state model, which will serve as a foundation for more elaborate models. For this model, we will show how optimal MLR reduces to a perfect matching of bigraph under I.I.D. transition time.

A. Preliminaries

A two-state model is shown in Fig.2, we will calculate permutation vector $\pi$ given $Y_k$ and $Y_i$. Here, the footprints $Y_k$ and $Y_i$ at states $S_k$ and $S_i$ are related through an unknown permutation vector $\pi$,

$$Y_{\pi}(j) = Y_i(\pi(j)) - T_j, \quad 1 \leq j \leq |Y_i|$$

where $T_j$ is the transition time of $Y_{\pi}(j)$, the $j^{th}$ footprint at $S_k$. $Y_i(\pi(j))$ is the $j^{th}$ element of the permuted $Y_i$, according to $\pi$. The footprints $Y_{\pi}(j)$ and $Y_i(\pi(j))$ are generated by the same transaction. We have $|Y_{\pi} \geq |Y_i|$, since some transactions may still be resident at $S_i$. The number of instances in state $S_0$ at the time of observation is

$$Cnt(S_0) = |Y_0| - |Y_i|$$

where the sum is over all the permutation vectors $\pi$ over $\{1,...,n\}$. Hence, the timestamps in the footprints reduce the number of valid matches. Since there is at least one perfect match, corresponding to the true transition pattern, we have $1 \leq \text{perm}(A) \leq n!$. The upper bound is achieved for a complete bigraph, i.e., when all the instance departures from $S_0$ occur after all the arrivals in the batch.

For a partial batch, some of the footprints at $S_i$ are not yet generated, and hence, a perfect bigraph matching is not feasible. For the case when the footprints may not arrive in the correct temporal order, any maximum cardinality bipartite matching is a valid match. However, when the footprints arrive in the correct order, we have additional information about the transactions which are still resident at $S_k$ and this changes the structure of the bigraph. Given that $|Y_i| = k < |Y_0| = n$, there are $n-k$ number of instances that have not yet made the transition and their departures occur after time $Y_k(k)$, the timestamp of the most recent footprint at $S_k$. This information is incorporated by adding $n-k$ number of identical copies of a dummy node, denoted by $V(\delta)$, to the bipartition $V$. Edges are added between $V(\delta)$ and any node $V_k(i)$ if $Y_k(k) - Y_k(i) < \Delta$, i.e., the deadline has not yet passed. Since all the dummy nodes are identical, some of the perfect matching in this bigraph are now equivalent, and the number of unique matching in a partial batch is

$$N_s(n,k,A) = \text{perm}(A) / (n-k)!$$

since the permutations among the copies of the added node $V(\delta)$ are equivalent. When $n=k$, it reduces to (4). It is NP-hard to compute $\text{perm}(A)$ in (5). Hence, we resort to approximations and bounds [7].

B. Optimal Tracking

When the joint PDF $f_T$ of the transaction transition times $T = [T(j)]$ is known in a two-state system, the ML matched in (1) reduces to $S_0$

$$\hat{\pi}^{ML}(Y_i,Y_i; f) := \arg \max_{\pi} P(Y_i^* \mid Y_i)$$

for a partial batch $Y_i$, the MLR now reduces to

$$\hat{\pi}^{ML}(Y_i,Y_i; f) := \arg \max_{\pi} \prod_{i \in \pi(1)} f_T[Y_i(\pi(i)) - Y_i(i)]$$

(7)

where $F_T(t) = P[T > t]$ is the CCDF. For the bigraph $G = V_0 \cup V_1, E >$, defined in the previous section with the added node $V(\delta)$ (henceforth, known as the CCDF node), we now assign a weight $W(i,j)$, for each edge $(i,j)$,
\[ W(i, j) = \begin{cases} -\log f_i[Y(i) - Y_v(i)], & j \leq k \\ -\log \bar{F}_i[Y_i(k) - Y_v(i)], & j = \delta \end{cases} \quad (8a) \]

\[ \pi^*(n, k, W) = \arg \max_{\pi} \sum_{i=1}^{n} W(i, \pi(i)) \quad (9) \]

An example of maximum weight matching is shown in Fig.3. MLR is simplified to weight matching, subfigure(a) is a complete batch and subfigure(b) is a partial batch. Maximum weight perfect matching can be performed in \(O(n(n + \log n))\) for a \(n\)-batch and \(m\) number of edges via Hungarian algorithm [8]. Hence, we see that the creation of batches leads to efficient implementation.

Above maximum-weight matching has exponential time complexity. When the values of \(n\) and \(m\) is very large, it is not suitable for footprint analysis with incomplete tokens that has very high requirement for time effectiveness. In section 4, an optimal maximum-weight matching algorithm is presented, which has effective \(O(\min(n + r, r\sqrt{n}m))\) time complexity in the case of linear space complexity. Assuming that there are 10000 nodes and 10000 edges in a bigraph, the partition of the edge set is 200. Adopting the method of literature [8], the consumption time of maximum weight matching is about \(1.02 \times 10^9\), however, according time is about \(1.02 \times 10^6\) with the method presented in the paper, the latter is only 1/10 of the former.

IV. THE IMPROVED ALGORITHM FOR RANK-MAXIMAL MATCHING

A. Preliminaries

Let \( V \mapsto \mathbb{R}_{\geq 0} \) be a potential function defined on the vertices of \( G \). For an edge \( e = (v, w) \in E \) denotes its weight by \( c(e) \) and defines its reduced weight with respect to \( \pi \) as \( \overline{\pi}(e) = \pi(v) + \pi(w) - c(e) \). Such an edge is tight if \( \overline{\pi}(e) = 0 \). We say that \( \pi \) is a feasible potential function for \( c \) if \( \overline{\pi}(e) \geq 0 \) for all edges \( e \in E \). We say that a feasible potential function \( \pi \) is optimal if there is a matching \( M \) in \( G \) such that \( \overline{\pi}(e) = 0 \) for each \( e \in E \) and \( \pi(v) = 0 \) for all \( v \in V \) which are free in \( M \).

The algorithm that we present uses a decomposition theorem by Kao et al. [9]. For an integer \( h \in [1, C] \), where \( C \) is the maximum-weight of an edge, divide \( G \) into two lighter subgraphs \( G_L \) and \( G_R \) as follows: (a) \( G_L \) is formed by edges \((u, v) \in G\) such that \( c(e) \in [C - h + 1, C] \). An edge \( e \in G_L \) has weight \( c(e) = c(e) - (C - h) \). b) let \( \pi_a \) be an optimal potential function (OPF) for \( G_L \). An edge \( e = (v, w) \in G \) belongs to \( G_a \) if \( \pi_a(u) + \pi_a(v) - c(e) < 0 \). In that case edge \( e \) has weight \( c_a(e) = c(e) - \pi_a(u) - \pi_a(v) \).

Theorem 2. Consider \( G_a \), \( G_L \), and \( G_R \), as defined above and let \( mwm(G) \) denote the weight of a maximum weight matching in \( G \). Then \( mwm(G) = mwm(G_L) + mwm(G_R) \).

If \( \pi_a \) and \( \pi_b \) are OPFs for \( G_a \) and \( G_b \) respectively, then \( \pi_a + \pi_b \) is an OPF for \( G \) [9].

B. The Decomposition of Edge

Consider an instance of the rank-maximal matching problem and the reduced instance of the weight matching problem. The edges of \( G \) have weights of the form \( 1, n, n^2, \ldots, n^{r-1} \). Using Theorem 2, we decompose the problem and solve it recursively. The base case of the recursion is a maximum cardinality matching computation.

Choose \( h = n^{r-1} - 1 \), then \( G_h \) contains the edges of \( G \) with weight in the range \([2, n^{r-1}]\). Each edge in \( G_h \) has weight \( c_h(e) = c(e) - 1 \). Thus, \( G_h \) contains all edges of \( G \) with rank at most \( r-1 \). These edges have weights \( n-1, n^2-1, \ldots, n^{r-1}-1 \). Assuming that \( \pi_h \) is an OPF for \( G_h \), \( G_h \) contains only the edges of \( G \) with negative reduced weight \( \pi_h \). Such edges fall into two categories: (a) Edges \( e = (v, w) \in G \) where \( c(e) = 1 \) and \( \pi_h(u) + \pi_h(w) = 0 \). Such edges have cost 1 in \( G_h \).
Edges $e = (v, w) \in G$, where $c(e) > 1$ and $\pi_0(u) + \pi_0(w) - c(e) < 0$. Due to the feasibility of $\pi_0$ in $G$, $\pi_0(u) + \pi_0(w) - c(e) \geq 0$, where $c(e) = c(e) - 1$. We conclude that $\pi_0(u) + \pi_0(w) - c(e) \geq -1$ and therefore all such edges also have cost 1 in $G'$. These edges are exactly the ones in $G'$ which are tight $\pi_0$.

The decomposition results in two subproblems (an example in Fig. 4). The subproblem in $G'$ is a maximum cardinality matching, since all edges have weight 1. On the other hand graph $G_h$ has edges with weights of the form $n-1, n^2-1, \ldots, n^{r-1}-1$. We will show that an optimally potential function for these weights is also optimal for the weights $1, n, n^2, \ldots, n^{r-2}$. Thus, the subproblem in $G_h$ is a rank-maximal matching computation with $r-1$ ranks, which is recursively solved in time $T(r-1)$.

The cost of the algorithm should be considered, since the potential function $\pi_0$ can take values up to $O(n^r)$. In this respect, we consider the following representation for an OPF $\pi$:

(a) A set of nodes $V^0$ containing all nodes $v \in V$ s.t. $\pi(v) = 0$. (b) A set of edges $E^0$ containing all edges $e = (v, w) \in E$ which are tight $\pi$, i.e., $\overline{\pi}(e) = \pi(u) + \pi(w) - c(e) = 0$.

The above two sets can guide the construction of a matching $M$ in $O(nmn)$ time such that any matched edge is tight and every free vertex has zero potential. The above implies that such a matching has the same cost as our OPF and, therefore, is itself optimal. In the algorithm we will manipulate these two sets and maintain the invariant that all such tuples will correspond to the representation of some OPF.

C. The Problem of Ranks

Let $G$ be a graph with edge weights $1, n, n^2, \ldots, n^{r-2}$ and let $V^0$ and $E^0$ be two sets representing an OPF. In this section we show that the same sets are a solution for the edge weights $n-1, n^2-1, \ldots, n^{r-1}-1$. More precisely, there exists an OPF such that these two sets are its representation.

Assume that we have solved the subproblem with edge costs $1, n, n^2, \ldots, n^{r-2}$ in time $T(r-1)$ and we have an OPF $\pi$ represented by $V^0$ and $E^0$. In order to obtain an OPF for the edge costs $n-1, n^2-1, \ldots, n^{r-1}-1$, the first step is to obtain an OPF for the edge costs $n, n^2, \ldots, n^{r-1}$. The following theorem illustrates it.

Theorem 3. Consider a graph $G$ with edge costs $1, n, n^2, \ldots, n^{r-2}$. Let $M$ be a maximum-weight matching of $G$ and $\pi : V \mapsto [0, \infty)$ be a potential function proving its optimality. Then the potential function $n\pi$ proves the optimality of $M$ for $G$ with edge costs $n, n^2, \ldots, n^{r-1}$.

Let $\pi$ be the potential function obtained by multiplying $\pi$ with $n$ in theorem 3. From the feasibility of $\pi$, the definition of $\pi$ and the integrality of the potential functions, we also get the following theorem.

Theorem 4. For any node $v \in V$ either $\pi(v) = 0$ or $\pi(v) \geq n$. Moreover, for any edge $e \in E$ either $\overline{\pi}(e) = \pi(u) + \pi(w) - c(e) = 0$ or $\overline{\pi}(e) \geq n$.

The same sets $V^0$ and $E^0$ are a representation of the new OPF $\pi$ for the new weights. The next step is the construction of a potential function $\pi'$ which will be optimal for the weight $n-1, n^2-1, \ldots, n^{r-1}-1$. The OPF algorithm [10] constructs such a potential function.

Definition 1: Equality Subgraph. For a graph $(G(V, E))$ with edge weights $c : E \mapsto [0, \infty)$ and a potential function $\pi : V \mapsto [0, \infty)$, let the equality subgraph $G_e(V, E_e)$ be the graph with edge set $E_e = \{e = (u, v) \in E : \overline{\pi}(e) = \pi(u) + \pi(v) - c(e) = 0\}$.

D. The Problem of Combination

Now, we are left with the following two subproblems:

(a) Graph $G'_n$ and an OPF $\pi'_n$ which was obtained by a maximum cardinality matching computation, and (b) graph $G_h$ and sets $V^0$ and $E^0$ representing an OPF $\pi_h$.

Combining the two solutions requires to add up the two potential functions, $\pi_n$ and $\pi_h$. The addition will be performed implicitly by changing $V^0$ and $E^0$ based on the potential function $\pi_h$. Updating $V^0$ requires checking for each $v \in V$ whether $\pi_n(v) = \pi_h(v) = 0$ which can be done by checking whether $\pi_n(v) = 0$ and $v \in V^0$. The process of updating $E^0$ is as follows:

(a) For an edge $e = (v, u)$ with $c(e) = 1$ in $G$, i.e. $e \in E_e$, we have to check whether $\pi_n(v) + \pi_n(u) + \pi_h(v) + \pi_h(u) = 1$. Recall from Section 4.3 that if a vertex $v \in V$ has $\pi_n(v) > 0$ then $\pi_n(v) > n-2$ and therefore it is enough to check: a) that $\pi_n(v) + \pi_n(u) = 1$ (an operation which takes constant time since $\pi_n$ is polynomially bounded) and b) that $\pi_h(v) = \pi_h(u) = 0$ which can be done by checking whether $(v, u) \in E^0$.

(b) For the rest of the edges, we have to check whether $\pi_n(v) + \pi_n(u) + \pi_h(v) + \pi_h(u) = c(e)$. By the feasibility of $\pi_n$ we know that $\pi_n(v) + \pi_n(u) \geq c(e) - 1$. For an edge, $\pi_n(v) + \pi_n(u) \neq c(e) - 1$, we know that $\pi_n(v) - 1 + \pi_h(u) - 1 - c(e) - 1 \geq n-1$ (in the worst case where both endpoints got their potential decreased by 1, when transforming to a new potential function for edge weights which are reduced by 1), and hence any such edge cannot be tight.

We conclude that if an edge has $\pi_n(v) + \pi_n(u) = c(e) - 1$ and therefore belongs already to $E^0$, then it will remain tight if $\pi_n(v) + \pi_n(u) = 1$. The final output of the algorithm is sets $V^0$ and $E^0$ which represent an OPF.

From these sets a rank-maximal matching can be constructed by performing one maximum cardinality matching.
matching computation. This is done in the following way. Let \( G(V, E') \) be the final equality subgraph. Create a new graph \( G_0^{eq} \), containing two copies of \( G^{eq} \), \( G_0(V, E') \) and \( G_0^b(V, E') \). For a vertex \( v \in V \), let \( v^a \) and \( v^b \) be the two copies in the new graph. Then, if \( v \in V_0 \) include the edge \((v^a, v^b)\). Finally find a maximum cardinality matching \( M \) in \( G_0^{eq} \). The matching \( M \cap E'' \) is then a maximum weight matching in the original graph.

**Algorithm 1**

The rank-maximal matching algorithm

**Input:** graph \( G \) with edge partition \( E_1, E_2, ..., E_r \)

**Output:** sets \( V^0, E^0 \)

If \( r = 1 \)

Compute maximum matching \( M \) of \( G \) and optimal \( \pi \); based on \( \pi \), Compute \( V^0 \) and \( E^0 \) and return them;

Else

Solve recursively instance for \( G(V, E \setminus E_r) \) and \( E_1, E_2, ..., E_{r-1} \);

Let \( V^0 \) and \( E^0 \) be the solution

Let \( E = \{e = (v, u) \in E : (v, u) \in V_0^0\} \)

Form unweighted \( G(V, E' \cup E^0) \) and find OPF \( \pi_\alpha \) in \( G_t \);

Set \( E^0 = \{e = (v, u) \in E' \cup E^0 : \pi_\alpha(v) + \pi_\alpha(u) = 1\} \);

Set \( V^0 = \{v \in V_0^0 : \pi_\alpha(v) = 0\} \)

Return \( V^0 \) and \( E^0 \);

Endif

End.

**V. SEMI-MARKOV PROCESS MODEL**

The two-state model studied in the previous section represents a high-level model where the only observable points are the system entry and exit points. When more system points are observable, e.g., the entry and exit points of sub-processes such as flight booking, train booking, hotel booking, online pay, ticket delivery with FedEx etc., the two-state model can be expanded to a multi-state model. Assuming that the transition times of each transaction form a multi-state semi-Markov process (SMP) and the transitions of different transactions are independent of one another, we consider ML matching of all the available footprints to the transactions.

We find the most-likely match between all the available footprints and recall that it is given by the ML-sequence of permutation vectors in (1),

\[
\bar{\pi}_{1_{1}}^{\text{ML}}, ..., \bar{\pi}_{s_{1}}^{\text{ML}} = \arg\max_{\pi_{1}, ..., \pi_{s_{1}}} P(Y_{1}^{s_{1}}, ..., Y_{s_{1}}^{s_{1}} | Y_{0}) \quad (11)
\]

By convention, \( \pi_{0} = I \). A brute-force search for the ML-sequence of permutation vectors is over all possible footprints paths from the start state \( S_0 \) to all the terminating states. It is unclear if this problem has a reduction to bigraph matching, as in the two-state system. However, the search can be simplified through semi-Markov property which states that the transition time only depends on current state and next state, and hence,

\[
P(Y_{1}^{s_{1}}, ..., Y_{s_{1}}^{s_{1}} | Y_{0}) = \prod_{m=1}^{S_{1}} P(Y_{m}^{s_{m}} | \bigcup_{j \notin P(m)} Y_{j}^{s_{j}}) \quad (12)
\]

Each term in the product has a structure similar to the two-state system. However, the set of states occurring in any two terms of (12) may not be disjoint, since the sets of immediate predecessors \( P(k) \) and \( P(l) \) of any two states \( S_k \) and \( S_l \) may not be disjoint. This implies that in general, we cannot independently match the footprints in each term in (12). Hence, we need to and construct high-level states, comprising of many model states that “localize” the movement of footprints, thereby enabling us to perform matching independently within these high-level states. To this end, define a partition of states \( (B_{m})_{m=0} \) such that states in any two sets in the partition do not share a common immediate predecessor,

\[
P(S_k) \cap P(S_l) = \emptyset \quad \forall S_k \in B_m, S_l \in B_j, m, j \neq 0 \quad (13)
\]

Let \( B_0 = S_0 \), the start state. Since we have assumed that the state-transition digraph is acyclic (DAG), the partition in (13) is well defined. Therefore, we can rewrite (12) as

\[
P(Y_{1}^{s_{1}}, ..., Y_{s_{1}}^{s_{1}} | Y_{0}) = \prod_{m=0}^{S_{1}} P \left( \bigcup_{S_k \in B_{m}} Y_{k}^{s_{k}} \bigcup_{S_l \in P(B_k)} Y_{l}^{s_{l}} \right) \quad (14)
\]

where each term in the product corresponds to a bigraph system \( (P(B_k), B_m) \), with the start state \( P(B_0) \) and the terminating state \( B_m \). These bigraph systems are disjoint; a state cannot occur in two systems in the same role, since by definition, \( B_m \) are all disjoint sets (partition), and in (14), we also require the sets \( P(B_k) \) to be disjoint. Hence, we can conduct decentralizing matching in these bigraph systems. This also implies that knowledge of the footprints and the model parameters (such as the transition-time PDF) is only required “locally” within each bigraph system. After undertaking matching in all the bigraph systems, the most likely sequence of footprints produced by each transaction are constructed by splicing together the results of matching in bigraph systems to obtain the set of permutation vectors \( \pi_{1_{1}}^{\text{ML}}, ..., \pi_{s_{1}}^{\text{ML}} \) in (1).

**Theorem 5** Properties of partition \((B_{m})_{m=0} \). For a semi-Markov process with acyclic transition digraph and a partition of states \((B_{m})_{m=0} \), the following properties hold for any finite number of recursions iff (13) is true:

\[
N(P...(N(P(B_{m})))) \subset B_{m}, \ m > 0 \quad (15)
\]

\[
P...(N(P(B_{m}))) \subset P(B_{m}), \ m > 0 \quad (16)
\]

The paper adopts bigraph states partition algorithm presented in literature [9].

We now specify the nodes and the edge weights for each bigraph system \((P(B_k), B_m) \). Along the formula (3), we define \( \text{Cnt}(P(B_k)) \) as the number of instances residing at \( P(B_k) \) at the time of observation, given by

\[
\text{Cnt}(P(B_k)) = |Y_{P(B_k)}| - |Y_{B_{m}}| \quad (17)
\]

A batch of footprints is defined between the successive zero-crossings of \( \text{Cnt}(P(B_k)) \). Along the formula (8), given the transition probability matrix \( P \) of the SMP, for any states \( S_k \in B_{m} \), \( S_l \in P(B_k) \) and I.I.D. transition times drawn from PDF \( f_{t_{ij}} \), the edge weight between the \( i^{th} \) footprint at \( S_k \) and the \( j^{th} \) footprint at \( S_l \) of a batch is given by
\[ W(i, j; S_i, S_j) := f_{\delta_i, j}(Y_j - Y_i), \forall S_i \in P(S_i), \quad 1 \leq i, j \leq |Y|, \quad 1 \leq j \leq |Y| \]

For a partial batch, we have \( \text{Ct}(P(B_n)) > 0 \) and some instances are still residing at \( P(B_n) \). When the footprints arrive in the correct temporal order, we define the CCDF events for each state \( S_i \in P(B_n) \), whenever there are more footprints at \( S_i \) than the total at all its immediate successors, i.e., \( \text{Ct}(S_i) = Y_i - \sum_{k \neq i} Y_k > 0 \). The probability that transaction corresponding to the \( k \)th footprint at \( S_i \) is still resident at \( S_i \) is given by

\[ P_{\text{res}}(i, S_i) = \sum_{t \in \delta_i} P_{\text{res}}(t, S_i) \]

where \( Y_i(Y_t) \) is the most recent footprint at state \( S_i \). In this case, the CCDF edge-weights corresponding to state \( S_i \) are

\[ W(i, j; S_i, S_j) := P_{\text{res}}(i, S_i), \quad \forall 1 \leq i \leq |Y| \]

\[ \text{Ct}(S_i) \] number of identical copies of the CCDF node \( \delta_i \) are added to the bigraph.

**Theorem 6:** MLR in SMP. Given a SMP with an acyclic state transition digraph and I.I.D. transitions, the MLR is given by the decentralized minimum-weight matching in the bigraph systems \((P(B_n), B_n)\), where partition \((B_n)\) is given by the algorithm in literature [9] and the edge weights for each bipartite system are given by (18) and (20).

The instance of MLR matching between transaction footprints and according instances producing them is shown in Fig. 5. The probability that all the transactions are correctly tracked correctly is the product of the ML-probabilities of all the bigraph systems.

The algorithm in literature [9] is used to get \( B_6 = \{A\}, B_6 = \{B, C, D\}, B_6 = \{E\} \) and \( B_6 = \{F, G, H\} \) partition, and \((\{A, C, D\}, \{B, C, D\}, \{B\}, \{E\}) \) and \((\{E, G, H\}, \{F, G, H\}) \) compose three bigraph system.

![Figure 5. The MLR matching example of transaction footprints and instances that produce them.](image)

### Table 1. The Footprints and Correspondence Timestamps in Fig.5

<table>
<thead>
<tr>
<th>No</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp</td>
<td>0.000</td>
<td>0.105</td>
<td>0.396</td>
<td>0.210</td>
<td>3.485</td>
<td>1.192</td>
<td>0.899</td>
<td>5.116</td>
</tr>
<tr>
<td>No</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

All possible footprints and according weight sum about bigraph \((\{A, C, D\}, \{B, C, D\})\) are shown in Table 2. In table 2, there are 18 kinds of footprint sequences in this bigraph, only 4 of them have weight sum, the reason is that the difference of timestamp of other footprint sequence is negative, such case can’t occur in actual environment. The second footprint has the maximum-weight sum in 4 kinds of footprint sequences, which is in correspondence with actual sequence in Fig 5. All possible footprints and according weight sum about bigraph \((\{B\}, \{E\})\) are shown in table 3. In table 3, there are 6 kinds of footprint sequences in this bigraph, only 2 of them have weight sum, and the second footprint has the maximum-weight sum, which is in correspondence with actual sequence in Fig 5. Above test results shows that our method may effectively strip transaction footprints with incomplete tokens.

For the accuracy test of footprints stripping(including partial batch), the stripping results is also in correspondence with actual sequence in Fig.5. Table 4 shows all possible footprints and according weight sum about bigraph \((\{A, C, D\}, \{B, C, D\})\). Here, footprint 6, 9, 12 and 16 have not been produced still, their weights are calculated with formula (8b). The second footprint has

### Table 2. All Possible Footprints and Weight Sum in Bigraph 1

<table>
<thead>
<tr>
<th>No</th>
<th>footprint sequence</th>
<th>weight sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1−4−6, 2−7, 3−5−8</td>
<td>10.8677</td>
</tr>
<tr>
<td>2</td>
<td>1−4−7, 2−6, 3−5−8</td>
<td>11.5921</td>
</tr>
<tr>
<td>3</td>
<td>2−4−6, 1−7, 3−5−8</td>
<td>10.7532</td>
</tr>
<tr>
<td>4</td>
<td>2−4−6, 3−7, 1−5−8</td>
<td>11.4001</td>
</tr>
<tr>
<td>5</td>
<td>......</td>
<td>1.0</td>
</tr>
</tbody>
</table>

### Table 3. All Possible Footprints and Weight Sum in Bigraph 2

<table>
<thead>
<tr>
<th>No</th>
<th>footprint sequence</th>
<th>weight sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6−9, 7−11, 8−10</td>
<td>3.9788</td>
</tr>
<tr>
<td>2</td>
<td>6−11, 7−9, 8−10</td>
<td>3.8297</td>
</tr>
<tr>
<td>3</td>
<td>......</td>
<td>1.0</td>
</tr>
</tbody>
</table>

© 2012 ACADEMY PUBLISHER
TABLE 4.
ALL POSSIBLE FOOTPRINTS AND WEIGHT SUM IN BIGRAPH 1 (INCLUDING PARTIAL BATCH).

<table>
<thead>
<tr>
<th>No</th>
<th>footprint sequence</th>
<th>weight sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1→4→6, 2→7, 3→5→8</td>
<td>10.9853</td>
</tr>
<tr>
<td>2</td>
<td>1→4→7, 2→6, 3→5→8</td>
<td>11.2876</td>
</tr>
<tr>
<td>3</td>
<td>2→4→6, 1→7, 3→5→8</td>
<td>11.1052</td>
</tr>
<tr>
<td>4</td>
<td>2→4→6, 3→7, 1→5→8</td>
<td>10.8574</td>
</tr>
</tbody>
</table>

TABLE 5.
ALL POSSIBLE FOOTPRINTS AND WEIGHT SUM IN BIGRAPH 2 (INCLUDING PARTIAL BATCH).

<table>
<thead>
<tr>
<th>No</th>
<th>footprint sequence</th>
<th>weight sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6→9, 7→11, 8→10</td>
<td>3.3656</td>
</tr>
</tbody>
</table>

Figure 6. The misdiagnosis rate of transaction footprints stripping

B. The Compression Test of Algorithm Performance

Assuming that the emergence of each state of footprints obeys normal distribution, we will compare performance between traditional maximum-weight matching algorithm and improved one presented in the paper (Fig. 7). In the process of algorithm performance test, we increase bigraph vertex to 100 and make statistics time spending. We see that the temporal performance of improved algorithm is superior to traditional one more than tenfold in the same experiment environment, which is suitable for footprint stripping.

VI. CONCLUSION

The paper discusses transaction footprint stripping with incomplete tokens that researchers pay little attention to. The main idea is as follow: if transaction states are independent each other and their transition time composes a multimode SMP, all possible states are separated into multi-cutsets, each of them composes a bigraph system, the respective matching is executed in each bigraph system, and then the results of independent matching of many bigraph systems are spliced to gain the most possible footprint sequences, which is post-stripping and labeled. For satisfying the requirements of analysis efficiency in open environment, the paper improves traditional maximum-weight matching algorithm. Simulation experiment confirms that the method presented in this paper can effectively implement transaction footprint stripping with incomplete tokens.

ACKNOWLEDGMENT

The financial supports from the National Basic Research Program of China (973 Program) under the grant No. 2011CB302600, the Post-doctoral Science Foundation of China under the grant No. 20080440216, the Natural Science Foundation of Hunan Province under the grant No. 11JJ4050 and the Education Department Foundation of Hunan Province under the grant No. 11B039 are gratefully acknowledged.

REFERENCES


Changyun Li born in Leiyang, China, on Sep, 7, 1971. He received the PhD in Computer Software and Theory from Zhejing University in 2006. He is professor in School of Computer and Communication of Hunan University of Technology. His research interests include trust software, software architecture.