An Improved Method for Transaction Footprints Stripping with Bigraph System

Changyun Li[1,2]
Email: lcy469@163.com
[1]School of Computer Science, National University of Defense Technology, Changsha 410073, China
[2]School 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.

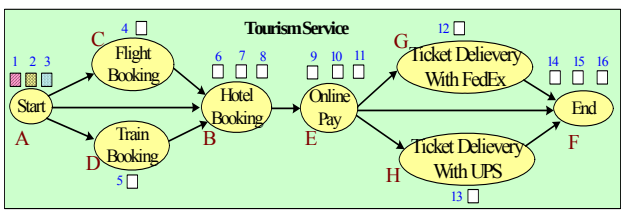


Figure 1. The transition model instance with incomplete tokens.

The model has Markovian when the time taken for a transaction to execute an application (represented as a state in the model) is only dependent on its outcome (represented as another state) and not on the past history of the transaction, the paper implements efficient and accurate stripping in special condition. The main idea is as follow: (a) construct corresponding relationships between external behavior and internal change of

doi:10.4304/jsw.7.9.2141-2148

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 $O(\sqrt{nm}\log nC)$ [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^r, 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 $O(n^r)$ takes time $\Omega(r)$. Hence, the running times are $O(rn(m+n\log n))$ and $O(r^2\sqrt{nm}\log n)$ 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 $O(r\sqrt{nm}\log n)$. 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 π 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 $G =< V_0 \cup V_1, E >$, the edges are represented by a 0-1 biadjacency matrix $A = [A(i, j)]$, here $A(i, j) = 1$ indicates an edge between $V_0(i)$ and $V_1(i)$. A matching $M \subset E$ 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, $i = 0,..., N_s$ denote the i^{th} state of the process and let $T_{i,j}$ 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 $T_{i,j}$ 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 $T_{i,j}$ drawn from a known PDF $f_{T_{i,j}}$ 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_0 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 π_k, for each state S_k in the model. Let Y_k 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_k is in ascending order with the most recent footprint being the last entry. Let $Y_k^{\pi_k}$ be the permutation of Y_k , according to the permutation vector π_k . By convention, we assign tokens to footprints Y_0 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 $N_s + 1$ number of states, the optimal MLR of transactions reduces to finding a set of N_s number of permutation vectors,

$$[\hat{\pi}_1^{ML}, \ ..., \ \hat{\pi}_{N_s}^{ML}] := \arg\max_{\pi_1,...,\pi_{N_s}} P(Y_1^{\pi_1}, \ ..., \ Y_{N_s}^{\pi_{N_s}} \mid Y_0) \quad (1)$$

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 showed in Fig.2, we will calculate permutation vector π given Y_0 and Y_1 . Here, the footprints Y_0 and Y_1 at states S_0 and S_1 are related through an unknown permutation vector π ,

$$Y_0(j) = Y_1(\pi(j)) - T(j), \quad 1 \le j \le |Y_1| \quad (2)$$

where $T(j)$ is the transition time of $Y_0(j)$, the j^{th} footprint at S_0 , $Y_1(\pi(j))$ is the j^{th} element of the permuted Y_1 , according to π . The footprints $Y_0(j)$ and $Y_1(\pi(j))$ are generated by the same transaction. We have $|Y_0| \ge |Y_1|$, since some transactions may still be resident at S_0 . The number of instances in state S_0 at the time of observation is

$$Cnt(S_0) = |Y_0| - |Y_1| \quad (3)$$

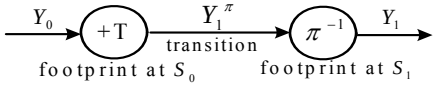


Figure 2. A two-state transactions model

Given the biadjacency matrix *A*, the number of unique valid matches in a batch, denoted by $N_B(|Y_0|,|Y_1|,A)$, provides an idea of the precision of tracking individual instances. For a complete batch with $|Y_0| = |Y_1| = n$, $N_B(n,n,A)$ is given by the permanent of the biadjacency matrix,

$$N_B(n,n,A) = perm(A) := \sum_{\pi} \prod_{i=1}^{n} A(i,\pi(i)) \quad (4)$$

where the sum is over all the permutation vectors π 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 $1 \le perm(A) \le n!$. 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_1 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_0 and this changes the structure of the bigraph. Given that $|Y_1| = k < |Y_0| = n$, there are *n-k* number of instances that have not yet made the transition and their departures occur after time $Y_1(k)$, the timestamp of the most recent footprint at S_1 . This information is incorporated by adding *n-k* number of identical copies of a dummy node, denoted by $V_1(\delta)$, to the bipartition V_1. Edges are added between $V_1(\delta)$ and any node $V_0(i)$ if $Y_1(k) - Y_0(k) < \Delta$, 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

$$N_B(n,k,A) = \frac{perm(A)}{(n-k)!} \quad (5)$$

since the permutations among the copies of the added node $V_1(\delta)$ are equivalent. When *n=k*, it reduces to (4). It is NP-hard to compute 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 match in (1) reduces to S_0

$$\begin{aligned} \hat{\pi}^{ML}(Y_0, Y_1; f) &:= \arg\max_{\pi} P(Y_1^{\pi} \mid Y_0) \\ &= \arg\max_{\pi} f_T[Y_1^{\pi} - Y_0] \end{aligned} \quad (6)$$

The MLR for a general joint PDF f_T of the transition times $T = [T(j)]$ requires search over all the permutation vectors π , which could be exponential in the batch size. We now make a simplifying assumption that all the instance transition times *T*(1), *T*(2), . . . are I.I.D. with PDF f_T . For a (Y_0, Y_1) batch, the MLR now reduces to

$$\begin{aligned} \hat{\pi}^{ML}(Y_0, Y_1; f) = \arg\max_{\pi} & \prod_{1 \le \pi(i) \le k} f_T[Y_1(\pi(i) - Y_0(i)] \\ & \prod_{k < \pi(i) \le n} \bar{F}_T[Y_1(\pi(k) - Y_0(i)] \end{aligned} \quad (7)$$

where $\bar{F}(t) = P[T > t]$ is the CCDF. For the bigraph $G = < V_0 \cup V_1, E >$, defined in the previous section with the added node $V_1(\delta)$ (henceforth, known as the CCDF node), we now assign a weight $W(i, j)$, for each edge (i, j),

$$W(i,j) := \begin{cases} -\log f_T[Y_1(\pi(i)-Y_0(i)], & j \le k \quad (8a) \\ -\log \bar{F}_T[Y_1(\pi(k)-Y_0(i)], & j = \delta \quad (8b) \end{cases}$$

The CCDF node is added to the bigraph on the assumption that the footprints arrive in the correct order. When instead, the footprints do not arrive in order, the CCDF node is not added and the edge weights are solely given by (8a).

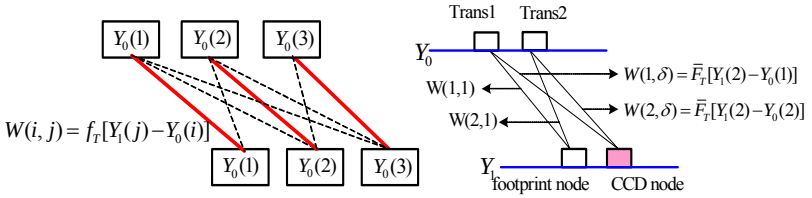
$$\pi^*(n,k,W) := \arg\max_{\pi} \sum_{i=1}^{n} W(i,\pi(i)) \quad (9)$$

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 $O(n(m+n\log n))$ 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, since n and m are substantially reduced. In the theorem below, we provide the MLR π^{ML} and the matching probability P^{ML}.

Theorem 1: I.I.D. transitions. In a two-state system, for I.I.D. transaction transition times according to a given PDF f_T, and the footprints arriving in the correct order, the MLR is given by the minimum-weight perfect matching in (10) and the probability that all footprints in an (n,k,W) batch are matched correctly under the MLR is

$$P^{ML}(n,k,W) = \frac{(n-k)!\exp(W^*)}{perm[\exp(W)]} \quad (10)$$

where $\exp(W) = \exp(W(i,j))$, W is given by (9), and W^* is matching value of maximum weight. For the case when the footprints do not arrive in the correct order, the MLR is a maximum-weight maximum cardinality, based solely on the edge weights in (8a).



(a)complete batch matching (b)partial batch matching including CCDF events

Figure 3. Simplifying MLR to weight matching

Above maximum-weight matching has exponential time complexity. When the values of n and m is very large, time taken by maximum-weight matching is very long, 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, this algorithm 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 $1.02*10^9$, however, according time is about $1.02*10^8$ 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 $V \mapsto \mathbb{R}_{\ge 0}$ 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 π as $\bar{c}(e) = \pi(v) + \pi(w) - c(e)$. Such an edge is tight if $\bar{c}(e) = 0$. We say that π is a feasible potential function for c if $\bar{c}(e) \ge 0$ or all edges $e \in E$. We say that a feasible potential function π is optimal if there is a matching M in G such that $\bar{c}(e) = 0$ 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_h and $G_h^{'}$ as follows: (a) G_h is formed by edges $(u,v) \in G$ such that $c(e) \in [C-h+1, C]$. An edge $e \in G_h$ has weight $c_h(e) = c(e) - (C-h)$. b) let π_h be an optimal potential function (OPF) for G_h. An edge $e=(v,w) \in G$ belongs to $G_h^{'}$ if $\pi_h(u) + \pi_h(v) - c(e) < 0$. In that case edge e has weight $c_h(e) = c(e) - \pi_h(u) - \pi_h(v)$.

Theorem 2. Consider G, G_h and $G_h^{'}$ as defined above and let $mwm(G)$ denote the weight of a maximum weight matching in G. Then $mwm(G) = mwm(G_h) + mwm(G_h^{'})$.

If π_h and $\pi_h^{'}$ are OPFs for G_h and $G_h^{'}$ respectively, then $\pi_h + \pi_h^{'}$ 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 $1, n, n^2, ..., n^{r-1}$. 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, graph G_h contains all edges of G with rank at most r-1 and these edges have weights $n-1, n^2-1, ..., n^{r-1}-1$. Assuming that π_h is an OPF for G_h, $G_h^{'}$ contains only the edges of G with negative reduced weight π_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 $G_h^{'}$. (b)

Edges $e=(v,w)\in G$ where $c(e)>1$ and $\pi_h(u)+\pi_h(w)-c(e)<0$. Due to the feasibility of π_h in G_h, $\pi_h(u)+\pi_h(w)-c_h(e)\geq 0$, where $c_h(e)=c(e)-1$. We conclude that $\pi_h(u)+\pi_h(w)-c(e)\geq -1$ and therefore all such edges also have cost 1 in G_h'. These edges are exactly the ones in G_h which are tight π_h.

The decomposition results in two subproblems (an example in Fig. 4). The subproblem in G_h' 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 $n-1, n^2-1, ..., n^{r-1}-1$. We will show that an optimally potential function for these weights is also optimal for the weights $1, n, n^2, ..., n^{r-2}$. Thus, the subproblem in G_h is a rank-maximal matching computation with r-1 ranks, which is recursively solved in time $T(r-1)$.

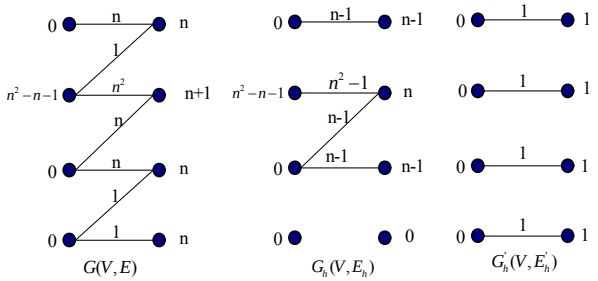


Figure 4. The edge decomposition example of Bigraph.

The cost of the algorithm should be considered, since the potential function π_h can take values up to $O(n^r)$. In this respect, we consider the following representation for an OPF π: (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 π, i.e., $\bar{c}(e)=\pi(u)+\pi(w)-c(e)=0$.

The above two sets can guide the construction of a matching M in $O(\sqrt{n}m)$ 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 $1, n, n^2, ..., n^{r-2}$ 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 $n-1, n^2-1, ..., n^{r-1}-1$. More precisely, there exists an OPF such that these two sets are its representation.

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

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

Let π' be the potential function obtained by multiplying π with n in theorem 3. From the feasibility of π, the definition of π' 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 $\bar{c}(e)=\pi'(u)+\pi'(w)-c(e)=0$ or $\bar{c}(e)\geq n$.

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

Definition 1: Equality Subgraph. For a graph $G(V,E)$ with edge costs c: $E\mapsto \mathbb{Z}_{>0}$ and a potential function $\pi: V\mapsto \mathbb{Z}_{\geq 0}$, let the equality subgraph $G_=(V,E_=)$ be the graph with edge set $E_= = \{e=(u,v)\in E:\ \bar{c}(e)=\pi(u)+\pi(v)-c(e)=0\}$.

D. The Problem of Combination

Now, We are left with the following two subproblems: (a) graph G_h' and an OPF π_h' which was obtained by a maximum cardinality matching computation, and (b) graph G_h and sets V^0, E^0 representing an OPF π_h.

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

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

(b) For the rest of the edges, we have to check whether $\pi_h(v)+\pi_h'(v)+\pi_h(u)+\pi_h'(u)=c(e)$. By the feasibility of π_h we know that $\pi_h(v)+\pi_h(u)\geq c(e)-1$. Moreover, for an edge, $\pi_h(v)+\pi_h(u)\neq c(e)-1$, we know that $\pi_h(v)-1+\pi_h(u)-1-(c(e)-1)\geq n-1$ (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 $\pi_h(v)+\pi_h(u)=c(e)-1$ and therefore belongs already to E^0, then it will remain tight if $\pi_h'(v)+\pi_h'(u)=1$. 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 computation. This is done in the following way. Let $G_{=}(V,E^0)$ be the final equality subgraph. Create a new graph $G^{\alpha\beta}$, containing two copies of $G_{=}$, $G^{\alpha}(V,E^{\alpha})$ and $G^{\beta}(V,E^{\beta})$. For a vertex $v \in V$, let v^{α} and v^{β} be the two copies in the new graph. Then, if $v \in V^0$ include the edge (v^{α},v^{β}). Finally find a maximum cardinality matching M in $G^{\alpha\beta}$. The matching $M \cap E^{\alpha}$ 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

Begin

If r = 1

Compute maximum matching M of G and optimal π;

based on π, 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 $E^+ = \{e = (v,u) \in E_r : (v,u) \in V^0\}$

Form unweighted $G_h^{'}(V, E^+ \cup E^0)$ and find OPF $\pi_h^{'}$ in $G_h^{'}$;

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

Set $V^0 = \{v \in V^0 : \pi_h^{'}(v) = 0\}$

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 delievery 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),

$$[\hat{\pi}_1^{ML},...,\hat{\pi}_{N_s}^{ML}] := \arg\max_{\pi_1,...,\pi_{N_s}} P(Y_1^{\pi_1},...,Y_{N_s}^{\pi_{N_s}} \mid Y_0) \quad (11)$$

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,

$$P(Y_1^{\pi_1},...,Y_{N_s}^{\pi_{N_s}} \mid Y_0) = \prod_{m=1}^{N_s} P(Y_m^{\pi_m} \mid \bigcup_{j \in P(m)} Y_j^{\pi_j}) \quad (12)$$

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 $(B_m)_{m \geq 0}$ such that states in any two sets in the partition do not share a common immediate predecessor,

$$P(S_k) \cap P(S_l) = \varnothing, \quad \forall S_k \in B_m,\ S_l \in B_j,\ m, j \neq 0 \quad (13)$$

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

$$P(Y_1^{\pi_1},...,Y_{N_s}^{\pi_{N_s}} \mid Y_0) = \prod_{m>0} P(\bigcup_{S_k \in B_m} Y_k^{\pi_k} \mid \bigcup_{S_l \in P(B_m)} Y_l^{\pi_l}) \quad (14)$$

where each term in the product corresponds to a bigraph system $(P(B_m), B_m)$, with the start state $P(B_m)$ 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 $P(B_m)$ 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 π_i^{ML} in (1). In the Theorem 5 below, some properties of the partition B_m are described.

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

$$N(P...(N(P(B_m)))) \subset B_m, \quad m > 0 \quad (15)$$

$$P...(N(P(B_m))) \subset P(B_m), \quad m > 0 \quad (16)$$

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

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

$$Cnt(P(B_m)) = \mid Y_{P(B_m)} \mid - \mid Y_{B_m} \mid \quad (17)$$

A batch of footprints is defined between the successive zero-crossings of $Cnt(P(B_m))$. Along the formula (8), given the transition probability matrix P of the SMP, for any states $S_k \in B_m$, $S_l \in P(B_m)$ and I.I.D. transition times drawn from PDF $f_{T_{k,l}}$, 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

$$W(i,j;S_k,S_l) := f_{T_{k,l}}(Y_l(j) - Y_k(i)),$$
$$\forall S_k \in P(S_l),\ 1 \le i, j \le |Y_k|,\ 1 \le j \le |Y_l| \quad (18)$$

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

$$P_{ccdf}(i,S_k) := \sum_{l \in N(k)} \bar{F}_{T_{k,l}}[Y_l(|Y_l|) - Y_k(i)] \quad (19)$$

where $Y_l(|Y_l|)$ is the most recent footprint at state S_l. In this case, the CCDF edge-weights corresponding to state S_k are

$$W(i,\delta_k;S_k) := P_{ccdf}(i,S_k),\quad \forall 1 \le i \le |Y_k| \quad (20)$$

$Cnt(S_k)$ number of identical copies of the CCDF node δ_k 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 $(P(B_m), B_m)$, where partition (B_m) 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 showed 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 $B_0 = \{A\}$, $B_1 = \{B, C, D\}$, $B_2 = \{E\}$ and $B_3 = \{F, G, H\}$ partition, and ({A, C, D}, {B, C, D}), ({B}, {E}) and ({E, G, H}, {F, G, H}) compose three bigraph system.

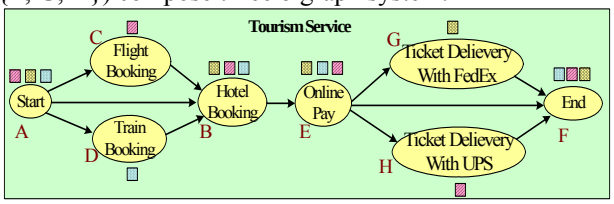


Figure 5. The MLR matching example of transaction footprints and instances that produce them.

VI THE SIMULATION EXPERIMENT

A. The Accuracy Test of Footprint Stripping

In order to validate the accuracy of footprint stripping method presented in this paper, practically monitored footprints with tokens are used as test data, we will compare computing results of footprints stripping with practical situation. Here, we only consider state transition time and ignore residence time in each states. Being convenient for illustration, we label every state(Fig. 1). The timestamp that footprints are in each state is showed in table 1.

TABLE 1.
THE FOOTPRINTS AND CORRESPONDENCE TIMESTAMPS IN FIG.5

No	1	2	3	4	5	6	7	8
timestamp	0.000	0.105	0.396	0.210	3.485	1.192	0.899	5.116
No	9	10	11	12	13	14	15	16
timestamp	2.095	5.898	1.608	4.096	2.066	6.586	3.215	6.118

All possible footprints and according weight sum about bigraph ({A, C, D}, {B, C, D}) are showed 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 showed 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 the maximum-weight sum in 4 kinds of footprint sequences, which is in correspondence with actual sequence in Fig 5.

At the aspect of pressure test of algorithm accuracy, we select 500 sets of actual data, and according testing results is illustrated in Fig. 6, the accuracy of stripping algorithm reaches 89% and misdiagnosis rate is about 11%. Compared to traditional methods, the improved algorithm has higher matching efficiency.

TABLE 2.
ALL POSSIBLE FOOTPRINTS AND WEIGHT SUM IN BIGRAPH 1.

No	footprint sequence	weight sum
1	1—4—6, 2—7, 3—5—8	10.8677
2	1—4—7, 2—6, 3—5—8	11.5921
3	2—4—6, 1—7, 3—5—8	10.7532
4	2—4—6, 3—7, 1—5—8	11.4001
5	……	——

TABLE 3.
ALL POSSIBLE FOOTPRINTS AND WEIGHT SUM IN BIGRAPH 2.

No	footprint sequence	weight sum
1	6—9, 7—11, 8—10	3.9788
2	6—11, 7—9, 8—10	3.8297
3	……	——

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

No	footprint sequence	weight sum
1	1—4—6, 2—7, 3—5—8	10.9853
2	1—4—7, 2—6, 3—5—8	11.2876
3	2—4—6, 1—7, 3—5—8	11.1052
4	2—4—6, 3—7, 1—5—8	10.8574
…	……	——

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

No	footprint sequence	weight sum
1	6—9, 7—11, 8—10	3.3656
…	……	——

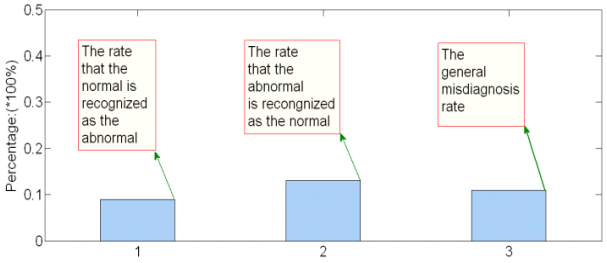


Figure 6. The misdiagnosis rate of transaction footprints stripping

B. The Comprasion 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.

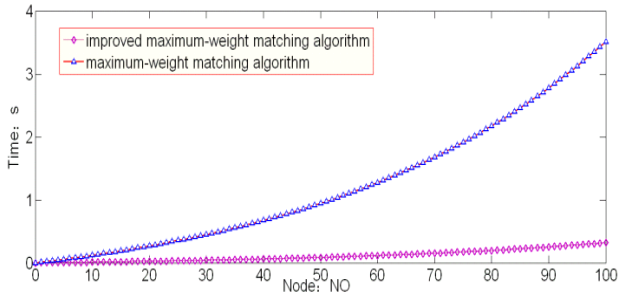


Figure 7. The contrast test of algorithm performance

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

[1] J. F. Man, Q. Q. Li, X. B. Wen. “Transaction Footprints Stripping with Incomplete Tokens in Open Network Environment”. PAAP 2010, Dalian, China. 2010, pp. 206-213.
[2] ARM. 2007-6-18, 2009-3-20. http://www.opengroup.org/tech/management/arm/.
[3] T. L. Ahuja, J. B. Magnanti. “Network Flows-Theory, Algorithms, and Applications”.Prentice Hall,1993, pp.1-35.
[4] H. N. Gabow, R. Tarjan. “Faster Scaling Algorithms for Network Problems. SIAM Journal of Computing”. Vol. 18, No. 5, 1989, pp. 1013–1036.
[5] R. W. Irving, T. Kavitha, K. Mehlhorn. “Rank-maximal Matchings”. The 15th Annual ACM–SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics. New Orleans, Louisiana, 2004, pp. 68–75.
[6] P. A. Ostrand. “Systems of Distinct Representatives. Journal of Mathematical Analysis and Applications”, Vol. 32, No. 1, 1970, pp. 1-4.
[7] L. Valiant. “The Complexity of Computing the Permanent”. Theoretical Computer Science, Vol. 8, No. 2, 1979, pp. 189-201.
[8] A. Anandkumar, C. Bisdikian, D. Agrawal. “Tracking in a Spaghetti Bowl: Monitoring Transactions Using Footprints”. The 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. Annapolis, MD, USA, 2008, pp. 133-144.
[9] J. Monnot, S. Toulouse. “The Path Partition Problem and Related Problems in Bipartite Graphs”. Operations Research Letters, Vol. 35, No. 5, 2007, pp. 677-684.
[10] D. Michai. “Reducing rank-maximal to maximum weight matching”. Theoretical Computer Science. Vol. 389, No. 1-2, 2007, pp. 125-132.

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.