http://www.jsebs.org
ISSN: 2288-3908

The Journal of Society for e-Business Studies
Vol.20, No.4, November 2015, pp.41-59
http://dx.doi.org/10.7838/jsebs.2015.20.4.041

로컬 API(Anomaly Process Instances) 탐지법을 이용한 컨테이너 터미널 이벤트 분석

The use of Local API(Anomaly Process Instances) Detection for Analyzing Container Terminal Event

전대욱(Daeuk Jeon)*, 배혜림(Hyerim Bae)**

초 록

시스템이 다양화 되면서 동시에 저장된 로그도 다양하게 분석할 필요가 생겼다. 이러한 로그 데이터 분석에 관한 필요성이 강해지는 환경이 시간 순으로 발생하는 이벤트 단위의 로그로부터 프로세스 모델을 도출하고, 시스템을 개선시키는 활동에 이바지하도록 요구하고 있다. 기존에는 개별 이벤트 단위의 로그를 분석하면서 속성들의 관계를 파악하는 연구가 활발했다. 본 논문에서는 로그 데이터를 활용한 예외적인 형태의 프로세스 인스턴스를 판별하는 방법으로 LAPID (Local Anomaly Process Instance Detection)를 제안한다. LAPID는 액티비티-릴레이션 매트릭스(Activity relation matrix)를 사용해서 계산된 거리 값을 활용하여, API(Anomaly Process Instance)를 탐색한다. 제시한 방법의 유용성을 검증하기 위하여 항만 물류에서 발생하는 컨테이너 이동에 대한 트레이스(Trace)를 포함하는 로그 데이터에서 예외적인 상황의 프로세스 실행이 가지는 특징을 도출하였다. 이를 위하여 본 논문에서는 국내의 실제 항만에서 발생한 이벤트 로그를 이용하여 사례연구를 수행하였다.

ABSTRACT

Information systems has been developed and used in various business area, therefore there are abundance of history data (log data) stored, and subsequently, it is required to analyze those log data. Previous studies have been focusing on the discovering of relationship between events and no identification of anomaly instances. Previously, anomaly instances are treated as noise and simply ignored. However, this kind of anomaly instances can occur repeatedly. Hence, a new methodology to detect the anomaly instances is needed. In this paper, we propose a methodology of LAPID (Local Anomaly Process Instance Detection) for discriminating an anomalous process instance from the log data. We specified a distance metric from the activity relation matrix of each instance, and use it to detect API (Anomaly Process Instance). For verifying the suggested methodology, we discovered characteristics of exceptional situations from log data. To demonstrate our proposed methodology, we performed our experiment on real data from a domestic port terminal.

키워드 : 이상치, 프로세스 마이닝, 프로세스 인스턴스, 로컬 이상치, 컨테이너 터미널
Anomaly, Process Mining, Process Instance, Local Anomaly, Container Terminal

---

This work was supported by the IT R&D program of MOTIE / KEIT (Ministry of Trade, Industry and Energy / Korea Evaluation Institute Of Industrial Technology No. 10054514).

* First Author, Pusan National University, Industrial Engineering(jdw@pusan.ac.kr)
** Corresponding Author, Pusan National University, Industrial Engineering(hrbae@pusan.ac.kr)

Received: 2015-08-07, Review completed: 2015-08-19, Accepted: 2015-09-09

1. 서 론

비즈니스 현장에 도입되는 시스템이 다양화되면서 수많은 데이터가 저장되고 있다. 저장된 데이터에 내재된 정보의 가치가 높게 평가되면서, 정보를 찾기 위한 방법이 연구되고 있으며, 여러 방법 중에 이벤트 로그(Event logs)를 활용한 프로세스 마이닝(Process mining)이 있다. 프로세스 마이닝은 이벤트를 인스턴스(Instance) 단위로 묶어서 해당 인스턴스의 트레이스(Trace) 정보를 분석함으로써 프로세스 모델(Process model)을 찾고 속성의 연관성을 찾는다[4, 11, 15]. 이러한 분석의 결과는 일반적인 프로세스 모델을 통해 비즈니스 프로세스(Business process)를 이해하는데 장점이 있다[14]. 그러나 데이터의 양이 급증하면서 데이터 전체의 특징을 이해하는 것과 함께 일부의 비정상적인 경로를 통해 발생되거나 혹은 특이한 값을 포함한 데이터를 분석할 필요성이 커지고 있다. 이와 같이 어떤 관측 개체가 다른 방법으로 발생되어, 나머지 관측 개체로부터 벗어나 있어서 비정상적이거나 특이하다고 판단되는 개체를 이상치(Anomaly)라고 한다. 이상치를 탐색하는 방법은 기계학습(Machine learning), 데이터 마이닝(Data mining), 패턴인식(Pattern recognition) 등의 분야에서 연구되고 있다. 그러한 결과로 개별 이벤트 정보의 속성값을 분석하여 이벤트의 관계를 규명하고, 그 의미를 파악하는 연구는 실제 많은 보안, 마케팅 방법에 활용되고 있다[16, 18, 27, 30].

프로세스 마이닝 분야에서도 이상치를 탐색하기 위하여 인스턴스를 단위로 하는 연구가 있었다[4]. 해당 연구에서는 이상치를 판별하기 위해 인스턴스가 발생하는 횟수에 기준하여 도출된 프로세스 모델을 설명하는 정도를 비교하였다. 그러나 기존 연구에서는 로그데이터에서 발생하는 유사 인스턴스 즉, 작업 목적이 같지만 이벤트의 반복적인 발생, 데이터 입력 오류 등을 포함한 인스턴스를 고려하지 못했다. 본 논문에서는 작업환경의 특성을 고려한 이벤트 로그에서 이상치를 판단하는 방법을 제시하고자 한다. 해당 이상치는 프로세스 마이닝을 위한 인스턴스로서 구체적으로 API (Anomaly process instance)를 찾는 방법을 제안한다. 그리고 고안된 방법론의 유효성을 컨테이너 터미널에서 발생한 실제 이벤트 로그를 사용하여 검증하고자 한다.

2. 관련 연구

2.1 프로세스 마이닝(Process Mining)

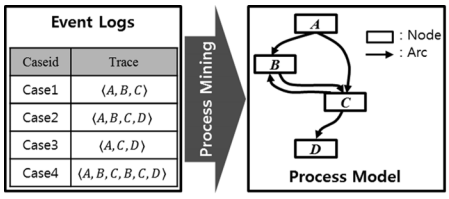
기존의 이상치를 탐색하는 방법론에서는 포인트(Point) 혹은 개체(Object) 단위로 이상치를 탐색하는데 반해서 프로세스 마이닝을 이용한 API 탐지는 시간 순으로 발생된 이벤트를 분석용 데이터로 사용하고 있다. 즉, 이상치 탐지의 대상이 프로세스 마이닝에서 분석을 위해 고려하고 있는 프로세스 인스턴스이다. 프로세스 마이닝은 정보시스템의 사용자가 남기는 이벤트 로그 정보를 취합하여 그로부터 지식을 도출하는 기술이다. 이것은 기존의 시스템에서 프로세스 모델을 찾아내는 도구로서 유용하며, <Table 1>과 같이 케이스식별자(Caseid), 액티비티(Activity), 타임스템프(Timestamp) 등의 속성을 활용한 데이터 분석을 수행한다. 근래에 많은 관심을 받고 있는 빅데이터 분야에서도

정보를 도출하기 위한 여러 방법 중에 하나로서 프로세스 마이닝을 고려하고 있다[29].

〈Table 1〉 Terms of Process Mining

Term	Description
Caseid	This is an identifier for distinguishing the sets of events. And, this is often referred to as 'process instance'.
Activity	This is an identifier for recognizing each work and a node of a process model.
Timestamp	This is time information that recorded with event.

프로세스 마이닝에서는 <Figure 1>과 같이 이벤트 로그 데이터에 포함된 많은 속성들 중에서 인스턴스를 구분하기 위해 케이스식별자를 키(Key)로 사용한다. 개별 인스턴스의 실제 작업인 트레이스는 이벤트의 집합으로서 이벤트를 구분하기 위한 속성으로 액티비티를 활용한다. 액티비티는 프로세스 모델(Process model) 상에서 노드(Node)에 표시된다. 본 논문에서는 프로세스 모델을 도출하기 위해 오픈 소스 툴인 'ProM'[28]을 사용하였다.



〈Figure 1〉 Overview for Detecting a Process Model from Event Logs

2.2 이상치 탐색(Anomaly Detection)

전통적인 통계분야에서 이상치는 확률밀도함수에서 상대적으로 낮은 발생확률을 갖는 관측치이다. 또한 이상치는 정상의 영역을 벗어난 관측 개체로서 일반적으로 정의된다. 정상과 비정상을 구분하기 위한 기준은 대부분 전문가의 영역에 속하는 것으로 시스템 혹은 서비스에서 해당 데이터가 의미하는 바를 정확히 이해하고 있어야 가능하다. 따라서 소수의 속성에 근거하여 데이터를 평가하는 것은 판단이 정확하지 않을 수 있으므로 이상치 탐지에도 다양한 분석 및 상이한 속성을 고려할 필요가 있다.

이상치 탐지는 일반적으로 데이터를 발생시키는 주체의 의도나 특성에 부합하지 못하는 데이터이다. 즉, 데이터가 생성되는 메커니즘이 다르다고 추측되는 데이터를 찾는 것이다. 이상치 탐지의 방법은 활용하는 분야에 따라서 다를 수 있으며, 분석을 위한 전처리 과정에서 비정상 데이터를 분류하기도 한다.

<Table 2>에서와 같이 각종 연구 분야 및 산업현장에서 이상치 탐지를 요구하는 경우가 많아지고, 개선을 위한 연구도 활발하다[2].

일반적으로 분석을 필요로 하는 복잡한 시스템에서는 분석을 수행하는 연구자가 전처리 과정을 효과적으로 이행하여야 한다. 모호한 데이터를 구분하고 이상치를 제외하기도 한다. 그러나, 경우에 따라 이상치를 제거함으로 인해 오히려 데이터의 특징을 도출하기 어렵게 한다. 분석 대상 데이터로부터 이상치를 정확한 기준으로 제외하고 있는지 검증되지 않기 때문이다[2]. 따라서, 예외적인 데이터가 가지는 의미가 필요하고, 대용량 데이터에서 명확하게 분석할 필요가 있다. 특정 시스템에서는 특징을 잘 나타내기 위해서 이상치 데이터를 활용하는 것이 더욱 용이할 때가 있고, 시스템이 이상적인 완벽함을 가지고 있지 못하기 때문이다. 예를 들어, 개발

〈Table 2〉 Needs of Anomaly Detection

Field	Description
Intrusion Detection Systems	In many host-based or networked computer systems, network traffic data may show unusual behavior because of malicious activity. The detection of such activity is referred to as intrusion detection [1].
Credit Card Fraud	In many cases, unauthorized use may show different patterns, such as a buying spree from geographically obscure locations. Such patterns can be used to detect outliers in credit card transaction data [22].
Interesting Sensor Events	The sudden changes in the underlying patterns may represent events of interest. Event detection is one of the primary motivating applications in the field of sensor networks [10].
Medical Diagnosis	In many medical applications the data is collected from a variety of devices such as MRI scans, PET scans or ECG time-series. Unusual patterns in such data typically reflect disease conditions [25].
Law Enforcement	Determining fraud in financial transactions, trading activity, or insurance claims typically requires the determination of unusual patterns in the data generated by the actions of the criminal entity [21].
Earth Science	A significant data about weather patterns, climate changes, or land cover patterns is collected through a variety of mechanisms such as satellites or remote sensing. Anomalies in such data provide significant insights about hidden human or environmental trends, which may have caused such anomalies [24].

단계에서 고려된 서비스와 개발 후 시스템이 제공하는 서비스의 범주가 다를 수 있다. 개발에 투입된 시스템 설계자의 정보력이 비즈니스 모델의 모든 경우를 포함할 수는 없기 때문이다. 최선의 결과물에 다다르기 위해서는 추가적인 보완 조치가 필요한 것이다. 서비스 또한 시점에 따라 다르게 적용되는 경우가 있다. 상품의 다양성, 소비자의 요구조건의 변화, 시장 상황의 변동 등이 끊임없이 비즈니스 모델을 변화시키고 있기 때문이다. '모든 변동성을 시스템에 적용할 수 있는가'라는 질문에 대해 '완벽한 시스템을 구축할 수 있는가'라는 해석 보다는 '얼마나 짧

〈Table 3〉 Techniques of Anomaly Detection [7]

Category		Technique
Classification Based		Neural Networks Based
		Bayesian Networks Based
		Support Vector Machines Based
		Rule Based
Nearest Neighbor Based		Distance to k Nearest Neighbor
		Relative Density
Statistical	Parametric Techniques	Gaussian Model Based
		Regression Model Based
		Mixture of Parametric Distributions Based
	Non-parametric Techniques	Histogram Based
		Kernel Function Based

은 시간에 시스템에 적용할 수 있는가'로 해석함이 옳다.

제품 혹은 서비스의 상품성 주기가 짧아지고 있기에 시스템 사용자의 탄력적인 운영이 필요하다. 시스템이 이미 탄력적인 상황을 고려하여 개발되었어도 미래를 예측하는 능력까지 기대할 수는 없다. 이러한 이유로 변형된 상황이 시스템에 끼치는 영향을 분석하기 위해서는 이상치 데이터의 의미가 더욱 크다고 할 수 있다.

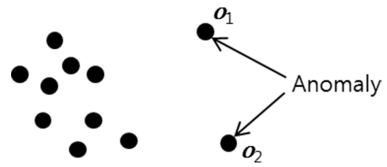
<Table 3>는 현재 많이 활용되는 이상치 탐지법이다. 본 논문에서는 '최근접 이웃 기반(Nearest neighbor based)' 분야의 '밀도기반(Relative density)'의 분석법을 활용한 LAPID(Local API detection)를 제안한다.

3. 밀도 기반 이상치 탐색

데이터의 형태가 다양해짐에 따라 이상치 분석법 또한 다양한 접근이 있었다. 최근에는 밀도 기반의 알고리즘(Algorithm)이 다양한 연구에 활용되고 있다[8, 17, 19]. 밀도 기반 탐지법은 LOF(Local outlier factor)를 계산하여 다차원 공간에서의 밀도를 비교하여 이상치를 판별한다[6]. 글로벌(Global) 관점에서는 두 개의 파라미터를 사용하여 이상치 여부를 판단하였으나[5], LOF 탐지법에서는 하나의 한계치인 k만을 파라미터로 사용하는 장점이 있다.

이상치는 기존의 연구된 방법으로 탐색되더라도 그 검증방법에 있어서는 한계를 가지고 있다. 데이터가 발생한 시점에서 시스템에 의해 일정한 작용으로 발생한 데이터를 이상치라고 정의하는 것은 외부적인 판단요인이 수 없이 영향을 줄 수 있기 때문이다. 예를 들어 사용자의 로그인 정보를 정상적으로 남긴 경우에도 서비스 범위를 벗어난 IP(Internet protocol)로 접속하는 경우이거나 혹은 통장 개설이 정상적으로 진행되었으나 승인자의 로그인 시간이 휴일인 경우 등과 같이 일반적이 않은 값이 전문가에 의해 판단될 수 있기 때문이다. 이렇듯 이상치 검증은 전문가의 영역이며, 탐색을 위한 방법론이 제시하는 결과는 전문가의 신속하고 정확한 판단을 돕는데 이바지 한다. 전문가를 보조할 수 있도록 시스템을 도입하고 있지만, 시스템이 모든 이상치를 미리 방지할 수 없는 것이 오히려 이상치를 탐지해야 되는 이유이기도 하다.

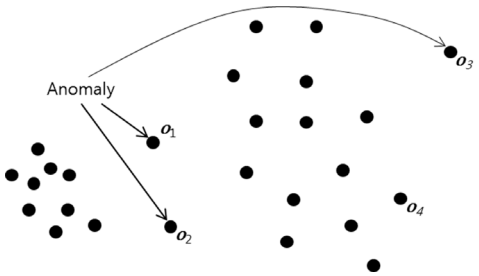
이상치를 탐지할 때 하나의 집단으로부터 상대적인 거리가 큰 개체를 찾는 방법으로 <Figure 2>에서 o_1, o_2의 경우에 다른 개체와의 거리가 크게 나타나는 글로벌 이상치 탐지법이 있다. 분석 데이터의 집합이 단일 군집을 이룰 때 이상치 판별법으로 용이하다.



〈Figure 2〉 Distance Based Anomaly

빅데이터 이슈와 더불어 다양한 로그 데이터의 패턴을 볼 때 분석을 필요로 하는 데이터는 <Figure 3>에서와 같이 거리만으로 판단하기 어려운 경우가 많다. 특히, o_1, o_2의 경우에 <Figure 2>에서 이상치로 판별 되었으나, 추가적인 데이터로 인해 정상의 범주에 포함될 수 있다. o_3는 직관적으로 분석 데이터에서 이상치임을 구분할 수 있다. 그러나 o_4의 경우는

판별하는 기준에 따라 달라진다. 이와 같이 보다 명확한 판별을 위해 로컬(Local) 이상치는 밀도를 고려하여, 일정한 거리에 존재하는 개체의 수가 작을 경우 즉, 주위의 밀도보다 작은 밀도를 가지는 개체를 이상치로 판단한다[12].



〈Figure 3〉 Density Based Anomaly

밀도를 기반한 이상치 판별법으로 LOF(Local outlier factor)[6]를 활용하였다. LOF는 전체 데이터에서 개별 개체로부터 일정 거리 내에 존재하는 타 개체들의 수를 (1)의 식과 같이 수치화하여 글로벌 이상치와 구분되는 로컬 이상치를 탐지할 수 있다.

다차원 공간의 밀도를 고려한 LOF 탐지법의 장점으로는 첫째, k값으로 조정할 수 있는 국소적인 이상치 탐지가 가능하다. 둘째, 개체들의 통계적 분포와는 무관하게 이상치를 검출할 수 있다. 셋째, 분포를 가정하는 파라미터가 필요하지 않고, 군집내의 개체 수를 나타내는 k만을 사용한다. 넷째, 모든 개체에 대해 이상치 탐색을 위한 단일 지표를 제공한다[17].

〈Table 4〉 Notations of Local Anomaly

Notation	Description
D	Set of objects
o	Object
$dist_k(o)$	The distance between and its k-nearest neighbor
$dist(o, p)$	The distance between and another object, $p \in D$
$N_k(o)$	The k-distance neighborhood of $N_k(o) = \{o' \mid o' \in D, dist(o, o') \leq dist_k(o)\}$

<Table 4>에서와 같이 D 데이터로부터 o 개체와 타 개체 사이의 거리를 $dist$로 정의하고, 해당 $N_k(o)$ 집합의 원소 수를 사용하여 식 (1)과 같이 LOF를 구한다.

$$LOF_k(o) = [\Sigma_{o' \in N_k(o)} \{lrd_k(o') / lrd_k(o)] / \| N_k(o) \| \quad (1)$$

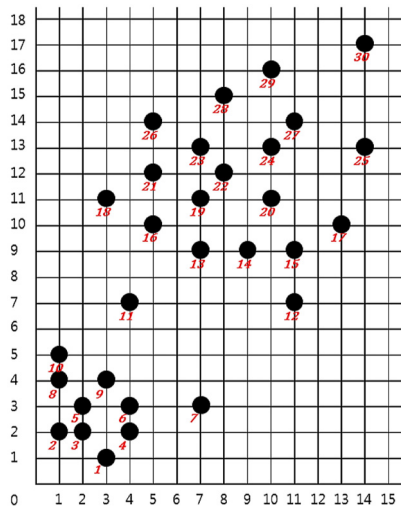
lrd(Local reachability density)는 식 (2)와 같이 구해지며, $N_k(o)$ 집합의 원소 수만큼 구하여야 한다. lrd를 구하기 위해 $reachdis_k(o, o')$는 식 (3)과 같이 $dist_k(o)$와 $dist(o, o')$에서 큰 값을 취한다.

$$lrd_k(o) = \| N_k(o) \| / \Sigma_{o' \in N_k(o)} reachdist_k(o, o') \quad (2)$$

$$reachdist_k(o, o') = max\{dist_k(o'), dist(o, o')\} \quad (3)$$

LOF는 하나의 개체를 기준으로 일정한 거리 내에서 밀도를 측정한다. 해당 밀도가 주변 개체를 기준으로 하는 일정한 거리 내에서의 밀도와 유사할수록 LOF가 1에 근사하게 되고, 밀도 차이가 클수록 1보다 큰 값이 나온다.

예를 들어 <Figure 4>와 같이 30개 포인트(Point)의 경우를 살펴보자. 포인트는 속성으로는 x, y 축에 위치하는 값을 가지며, 예제를 통해 2차원 평면에서 로컬 이상치를 찾아보았으며, 그 특징을 살펴보았다.



〈Figure 4〉 Example of 30-Points

〈Table 5〉 LOF of 30-Points with Each k-Value

Point ID	k = 3	k = 4	k = 5	k = 6
1	1.0822257	1.0511652	1.0357374	0.9692842
2	0.8547968	0.8948863	0.9751731	1.0174924
3	1.0388845	0.9689600	0.9845148	0.9978607
4	1.0565979	1.0224371	0.9699516	0.9166020
5	1.0329950	1.0180766	1.0143590	1.0268286
6	1.0497825	1.0224371	0.9452485	1.0021605
7	**1.7544229**	**1.6858739**	**1.6944900**	**1.4060583**
8	1.0416084	1.0411006	1.0514620	0.9435326
9	0.9398839	1.0026766	1.0014568	1.0108878
10	1.1624373	1.1262097	1.0923955	1.0401692
11	**1.5811168**	**1.3583346**	**1.3250220**	**1.1453431**
12	1.0811468	1.0763801	1.1129198	**1.2018961**
13	0.9221059	0.9933410	1.0267665	0.9998928
14	0.9440769	1.0592580	1.0771824	1.0580113
15	1.0956516	1.0763801	1.0374829	1.0764339
16	1.0656440	1.0731354	1.0131215	1.0661687
17	1.1253748	1.0680058	1.0913788	1.1174220
18	1.0337807	1.0752913	1.0869839	1.1400305
19	0.9868924	0.9057704	0.9430488	0.9577852
20	0.9253861	0.9185954	0.9934443	0.9568999
21	1.0784000	1.0202421	0.9949888	1.0168381
22	0.9190297	0.9295798	0.9065626	0.9071918
23	1.0182331	0.9736774	0.9485722	0.9322670
24	1.0415668	0.9517542	1.0153129	0.9838088
25	1.1618178	1.2147331	1.1900892	1.1887204
26	0.9706077	1.1604087	1.0496412	1.0434872
27	1.1330870	1.1144480	1.0189018	1.0262633
28	1.0105960	1.0976804	0.9961701	0.9983782
29	1.0851604	1.1259768	1.1514676	1.1983857
30	**1.2772324**	**1.3280538**	**1.3699802**	**1.3876095**

<Table 5>와 같이 '7', '11', '30' 포인트에서 LOF값이 크게 나타났다. 여기서 3개의 포인트는 순차적으로 선택한 것이므로 1개를 선택하거나, 5개를 선택하여 이상치로 판단하는 것은 연구자의 주관적인 판단에 근거한다.

실험에서 k값이 커지면 평면상에서 외곽의 인스턴스를 이상치로 판단하게 된다. 예제에서 '11' 보다 '12'의 LOF값이 커지는 경우로서 '11'은 상대적으로 중앙에 위치해 있고, '12'는 가장자리에 위치하고 있다. 따라서 LOF값은 적정한 k를 선정하는 것이 결과에 중요하게 작용한다.

4. API(Anomaly Process Instance) 탐지

4.1 액티비티-릴레이션 매트릭스 (Activity Relation Matrix)

이벤트 로그의 인스턴스는 앞선 예제의 포인트와는 다른 특징이 있다. 인스턴스에는 시간 순차적인 여러 이벤트가 포함되어 있다. 특히, 동일 이벤트를 포함한 인스턴스가 있을 수 있다. 즉, 같은 액티비티-릴레이션 매트릭스를

〈Table 6〉 Notations of Activity Relation Matrix

Notation	Description
e	Event
a_i	Activity
c^l	Caseid
h^l_{ij}	Number of c^l Frequency from a_i to a_j
M^l	Activity relation matrix of c_l
I^l	Set of process instances for c^l $I^l = \{e \mid \exists e \in \epsilon\}$, $I \subseteq \epsilon$,

가지는 인스턴스가 다수 존재할 수 있다. 하나의 인스턴스로 하나의 액티비티-릴레이션 매트릭스를 생성하고, 액티비티-릴레이션 매트릭스의 셀은 선행 이벤트와 후행 이벤트 정보를 가지는 액티비티-릴레이션(Activity relation)을 나타낸다.

액티비티-릴레이션 매트릭스는 순차적으로 발생한 이벤트를 나타내는 것으로서 방향성을 가지고 있다. 따라서 기준 액티비티(Base activity)와 대상 액티비티(Target activity)를 각각 고려하여 매트릭스를 구성하였다. <Table 7>에서 순차적으로 발생한 이벤트 데이터를 임의로 생성하였다. 임의 생성 데이터는 20개 케이스식별자의 130개 이벤트를 사용하였으며, 하나의 케이스식별자에 해당하는 이벤트의 트레이스는 < > 괄호로 표기하고, 하나의 이벤트는 알파벳 소문자로 표기하였다. 예제 데이터를 보면 케이스식별자가 'c^1'인 경우 포함한 이벤트의 액티비티명은 'a', 'b', 'c', 'd'로서 'a', 'b', 'c'는 1번씩 발생하였으며, 'd'는 2번 발생하였다. 각각의 이벤트는 시간순서를 가지고 있다[3, 23, 26].

〈Table 7〉 Example of Event Logs

Caseid	Event trace
c^1	<a, b, c, d, d>
c^2	<a, b, c, b, c, d>
c^3	<a, c, b, c, d, d, d>
c^4	<a, b, c, b, c, b, c, d, d>
c^5	<a, b, c, d, d>
c^6	<a, b, c, b, c, d>
c^7	<c, b, c, d, d, d>
c^8	<a, b, c, b, b, c, d, d>
c^9	<a, b, c, d, d>
c^{10}	<a, b, c, b, c, d>
c^{11}	<a, c, b, c, d, d, d>
c^{12}	<a, b, c, b, c, b, c, d, d>
c^{13}	<a, b, c, d, d>
c^{14}	<a, b, c, b, c, d>
c^{15}	<c, b, c, d, d, d>
c^{16}	<a, b, c, b, c, b, c, d, d>
c^{17}	<a, b, c, d, d>
c^{18}	<a, b, b, c, d>
c^{19}	<a, c, b, c, d, d, d>
c^{20}	<a, b, c, b, c, b, c, d>

케이스식별자가 'c^1'인 경우의 액티비티-릴레이션 매트릭스는 <Table 8>와 같이 행렬 형태로 변환된다. 'a' 액티비티에서 'b' 액티비티의 순서로 발생하는 경우를 "액티비티-릴레이션"으로 정의하고, 'ab'로 표기하였고, 'ab', 'bc', 'cd', 'dd' 액티비티-릴레이션이 각각 1번씩 발생하였다.

〈Table 8〉 Activity Relation Matrix

M^1		Target Activity			
		a	b	c	d
Base Activity	a		1		
	b			1	
	c				1
	d				1

액티비티-릴레이션의 존재를 비교하는 것이 기존 연구에서 문자열의 비교방법과 유사점이 있으나[20], 본 논문에서는 액티비티-릴레이션의 발생 횟수를 고려하고 있으며, 이벤트의 발생 순서를 고려하고 있다. 이벤트의 선후 관계를 내포한 액티비티-릴레이션을 활용하고, 액티비티-릴레이션의 발생 순서를 고려하지 않은 이유는 동일한 이벤트가 반복적으로 발생하기 때문이다.

인스턴스로부터 변환된 액티비티-릴레이션 매트릭스를 활용하여 <Table 4>의 '$dist$'를 구하기 위해서는 2개 인스턴스 사이의 거리를 구하여야 한다. 따라서, 식 (4)의 유클리디안 거

리(Euclidean distance)[9]를 사용한다.

$$\text{Distance: } d^{lm} = \|M^l - M^m\| = \{(h_{11}^l - h_{11}^m)^2 + (h_{12}^l - h_{12}^m)^2 + \cdots + (h_{ij}^l - h_{ij}^m)^2\}^{1/2} \quad (4)$$

액티비티-릴레이션의 발생여부로 측정하는 d^{lm}가 크다는 것은 비교대상(M^l, M^m)의 구조가 다르고, 발생 횟수의 차이가 크다는 의미이다.

4.2 로컬 API 탐지(Local Anomaly Process Instance Detection, LAPID)

로컬 이상치에 해당하는 인스턴스는 타 인스턴스와 상대적으로 거리가 떨어져있는 경우이고, 이는 밀도가 다르게 나타나는 이유이다. 거리 기반과 밀도 기반의 탐색 접근이 다른 것은 거리 측정 시에 활용한 액티비티-릴레이션 매트릭스의 특성에서 이해할 수 있다. 액티비티-릴레이션의 존재 여부와 그 발생 횟수로 인해 액티비티-릴레이션의 수만큼 존재하는 차원에서 멀어지거나 가까워지는 거리 값이 발생한다. 거리를 기준으로 하여 탐색하는 것보다 밀도를 기준으로 하여야 함을 <Figure 4>에서 7, 11포인트를 찾는 과정에서 보여주고 있다. 차원이 커질수록 거리 값만 고려한 탐색이 왜곡된 결과를 보일 수 있다, 따라서, 거리에 따라 API를 탐색하는 것에 비해 밀도를 고려하는 것이 보다 액티비티-릴레이션의 구조를 고려한 API임을 보이기에 용이하다. 기존의 프로세스 마이닝 분야에서 인스턴스의 발생빈도를 기준으로 판별하였으나[4], 액티비티-릴레이션 매트릭스를 활용한 LAPID는 인스턴스의 구조를 이벤트의 선후연관성에 주목하여 액티비티-릴레이션 구조에 따른 API를 탐색한다. LAPID 알고리즘은 <Figure 5>에 의사코드(Pseudo code)로 작성하였다.

Algorithm: Local API Detection (LAPID)
- Input data:
  $M = M^1, \cdots, M^l$: a set of activity relation matrix
  k: the number of nearest neighbors
- Output: Local API
- Method:

l: length of M
$bC = \{\ \}$: a set of k-th nearest activity relation matrix of base activity relation matrix
$tC = \{\ \}$: a set of k-th nearest activity relation matrix of target activity relation matrix
$n, m = 0$: number of elements
d_k^i: distance of k-th nearest activity relation matrix
$rd = 0$: reachdistance
$blrd = 0$: local reachability density of base activity relation matrix
$tlrd = 0$: local reachability density of target activity relation matrix
$sum\_lrd = 0$: temporary variable for calculating a sum
$lof = 0$: local outlier factor
$array\_lof = [10]$: top 10 of local outlier factor

```
1: for int i=0; i< l; i++
2:     for int j=0; j< l; j++
3:         if d^ij < d_k^i (i ≠ j)
4:             C ← M^j;
5:     n = |C|;
6:     for int j=0; j< n; j++
7:         if d_k^j > d^ij (i ≠ j)
8:             rd += d_k^j;
9:         else if d_k^j < d^ij (i ≠ j)
10:            rd += d^ij;
11:    blrd = n/rd;
12:    C = { };
13:    for int j=0; j< n; j++
14:        rd = 0;
15:        for int p = 0; p< l; p++
16:            if d^jp < d_k^j (j ≠ p)
17:                C ← M^j;
18:        m = |C|;
19:        for int p=0; p<m; p++
20:            if d_k^p > d^jp (j ≠ p)
21:                rd += d^jp;
22:            else if d_k^p < d^jp (j ≠ p)
23:                rd += d^jp;
24:        tlrd = m/rd;
25:        sum_lrd += tlrd/blrd;
26:    lof = sum_lrd/n;
27:    for int j=0; j<10; j++
28:        if lof > array_lof[j]
29:            array_lof[j] ← lof;
```

〈Figure 5〉 LAPID Algorithm

인스턴스로부터 LAPID를 적용하기 위해 <Table 7>의 예제 이벤트 로그를 분석하였으며, 변환된 액티비티-릴레이션 매트릭스는 <Table

〈Table 9〉 Activity Relation Frequency of Example-Instances

	aa	ab	ac	ad	ba	bb	bc	bd	ca	cb	cc	cd	da	db	dc	dd
1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1
2	0	1	0	0	0	0	2	0	0	1	0	1	0	0	0	0
3	0	0	1	0	0	0	1	0	0	1	0	1	0	0	0	2
4	0	1	0	0	0	0	3	0	0	2	0	1	0	0	0	1
5	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1
6	0	1	0	0	0	0	2	0	0	1	0	1	0	0	0	0
7	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	2
8	0	1	0	0	0	1	2	0	0	1	0	1	0	0	0	1
9	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1
10	0	1	0	0	0	0	2	0	0	1	0	1	0	0	0	0
11	0	0	1	0	0	0	1	0	0	1	0	1	0	0	0	2
12	0	1	0	0	0	0	3	0	0	2	0	1	0	0	0	1
13	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1
14	0	1	0	0	0	0	2	0	0	1	0	1	0	0	0	0
15	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	2
16	0	1	0	0	0	0	3	0	0	2	0	1	0	0	0	1
17	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1
18	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0
19	0	0	1	0	0	0	1	0	0	1	0	1	0	0	0	2
20	0	1	0	0	0	0	3	0	0	2	0	1	0	0	0	0

9>와 같이 구성하였다.

<Table 9>는 20개의 인스턴스로부터 16개 액티비티-릴레이션의 발생빈도를 나타내고 있다. ‘aa’ 액티비티-릴레이션에서는 모든 인스턴스에서 발생빈도가 0이지만 해당 액티비티-릴레이션을 제외를 하지 않아도 LOF 값에는 영향을 주지 않는다. 또한, 모든 인스턴스에서 발생빈도가 0이어서 제외하는 경우에 추가적인 데이터를 고려하여 분석하면, 제외된 액티비티-릴레이션이 발생할 수 있으므로 제외하지 않는 것을 기본으로 한다.

예제 인스턴스에서 LOF를 <Table 10>과 같이 구하였다. ‘3’, ‘11’, ‘19’ 인스턴스가 LOF값이 높게 나왔으며, 이상치로 판별되었다. LAPID에서는 k값이 커지면 액티비티-릴레이션의 개수만큼의 차원공간에서 가장자리에 위치한 인스턴스가 이상치로 판별된다.

〈Table 10〉 LOF of Example-Instances

	k = 5	k = 6	k = 7
1	1.0000	0.9963	0.9963
2	0.9715	0.9947	0.9947
3	**1.2213**	**1.0814**	**1.0814**
4	1.0885	0.9893	0.9893
5	1.0000	0.9963	0.9963
6	0.9715	0.9947	0.9947
7	1.1395	1.0082	1.0082
8	1.0965	1.0095	1.0095
9	1.0000	0.9963	0.9963
10	0.9715	0.9947	0.9947
11	**1.2213**	**1.0814**	**1.0814**
12	1.0885	0.9893	0.9893
13	1.0000	0.9963	0.9963
14	0.9715	0.9947	0.9947
15	1.1395	1.0082	1.0082
16	1.0885	0.9893	0.9893
17	1.0000	0.9963	0.9963
18	1.0000	1.0053	1.0053
19	**1.2213**	**1.0814**	**1.0814**
20	0.9922	1.0177	1.0177

4.3 거리 기반 API 탐지(Distance-Based Anomaly Process Instance Detection, DAPID)

LAPID로 탐색하는 API (Anomaly process instance)와 거리 기반의 DAPID의 결과를 비교하고자 한다. 비교를 통해 이상치의 결과를 상호 검증하며, 실제 다른 결과값이 도출되는 이유와 유사한 결과 부분을 해석하여 각각의 방법이 가지는 특징을 구분한다.

DAPID는 식 (5)와 같이 r과 π를 사용자가 선택하여 전체 인스턴스에서 인스턴스 사이의 거리가 가장 먼 것으로 판단되는 인스턴스를 판별한다[13].

〈Table 11〉 Notations of DAPID

Notation	Description
c^l	Case
C	Case universe $C=\{c_1, \cdots, c_T\}$
T	Total number of case
r	Distance threshold $(r>0)$
π	Fraction threshold $(0<\pi\leq 1)$
d^{lm}	Distance between M^l and M^m

$$\frac{\|\{c^l \mid d^{lm} \leq r\}\|}{T} < \pi \quad (5)$$

5. 사례 연구

컨테이너 이벤트 로그에 본 논문의 LAPID를

〈Table 12〉 Activity Relation Frequency of Containers

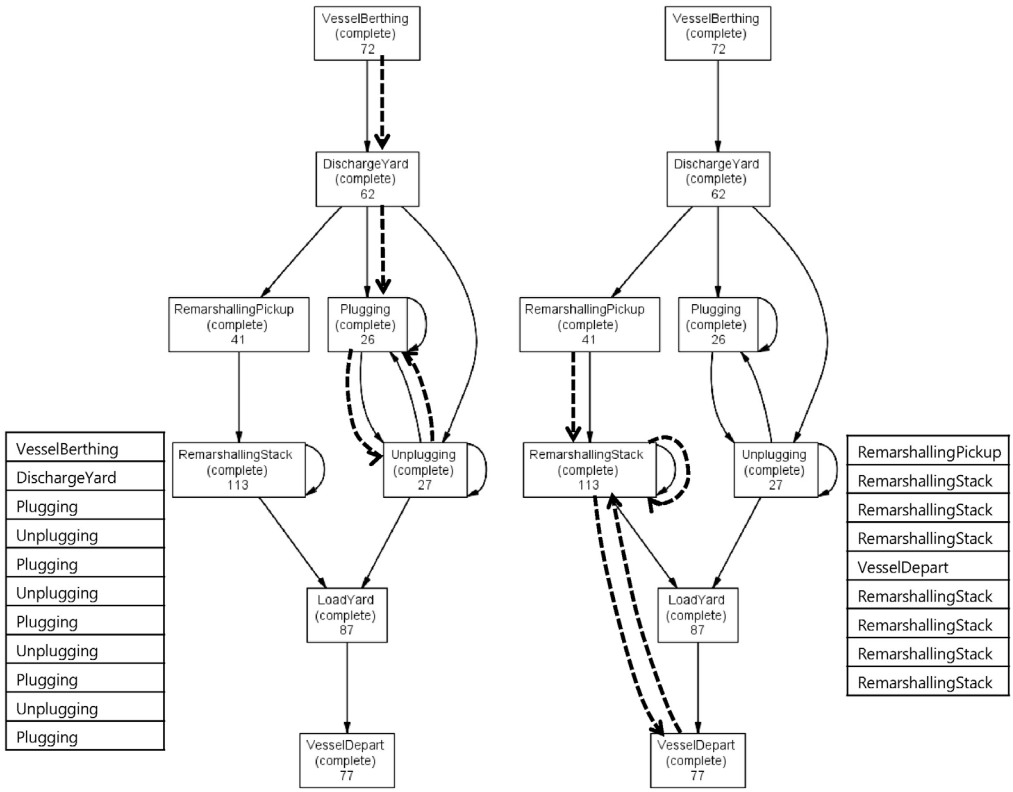
Activity relation / Caseid	DischargeYard//DischargeYard	DischargeYard//LoadYard	DischargeYard//Plugging	DischargeYard//RemarshallingPickup	DischargeYard//RemarshallingStack	DischargeYard//Unplugging	DischargeYard//VesselBerthing	DischargeYard//VesselDepart	LoadYard//DischargeYard	LoadYard//LoadYard	LoadYard//Plugging	LoadYard//RemarshallingPickup	LoadYard//RemarshallingStack	LoadYard//Unplugging	LoadYard//VesselBerthing	LoadYard//VesselDepart	Plugging//DischargeYard	Plugging//LoadYard	⋮
Container001	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
Container002	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
Container003	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
Container004	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
Container005	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
Container006	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
Container007	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	
Container008	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
Container009	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	
Container010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
⋮																			

적용하여 API를 탐색하였다. ㈜부산신항만 터미널에서 발생한 환적물 관련 컨테이너 처리 데이터에서 505개 이벤트를 추출하여 API를 판별하였다. 총 80개의 컨테이너를 포함하고 있으며, 8개의 액티비티를 고려하여, <Table 12>에서와 같이 액티비티-릴레이션의 발생빈도를 구하였다.

액티비티-릴레이션 매트릭스로부터 <Table 13>과 같이 이송정보에 대한 컨테이너별 LOF를 구하였다. 80개의 컨테이너 중에서 'Container058', 'Container041'의 LOF 값이 가장 크게 나타났고, 두 개의 컨테이너가 API로 탐지되었다. 이

〈Table 13〉 LOF of Containers

Caseid	k = 30	k = 31	k = 32	k = 33
~	~	~	~	~
Container039	1.0000	1.0000	0.9958	0.9966
Container040	2.4889	2.4889	1.6060	1.5931
Container041	**3.6188**	**3.6188**	**2.3940**	**2.3760**
Container042	1.0000	1.0000	0.9958	0.9966
~	~	~	~	~
Container057	1.8766	1.8766	1.2613	1.2531
Container058	**4.1484**	**4.1484**	**2.8461**	**2.8286**
Container059	3.0281	3.0281	2.0248	2.0102
Container060	1.0000	1.0000	0.9958	0.9966
~	~	~	~	~



(a) Container 058 (b) Container 041

〈Figure 6〉 API Path over a Container Process Model

러한 탐지는 분석대상 컨테이너 중에서 상대적인 순위이지, 절대적인 API로 판별하는 것과는 다르다. 또한 LAPID에서 k값의 범위는 이벤트 로그의 특징으로 인해 인스턴스로부터 영향을 받는다. 주된 이유로는 인스턴스에 동일한 이벤트가 반복적으로 발생하는 특징 때문이다. 이로 인해 동일한 프로세스를 가지는 인스턴스 개수가 많아질수록 LOF 값을 구하기 위한 k값 또한 커져야 한다. 이것은 LOF를 활용할 때의 단점으로서 k값을 일괄적으로 적용할 수 없다는 것이다. 그럼에도 불구하고, 단지 하나의 한계치만으로도 이상치를 탐지할 수 있다는 점에서 유용하다고 할 수 있다.

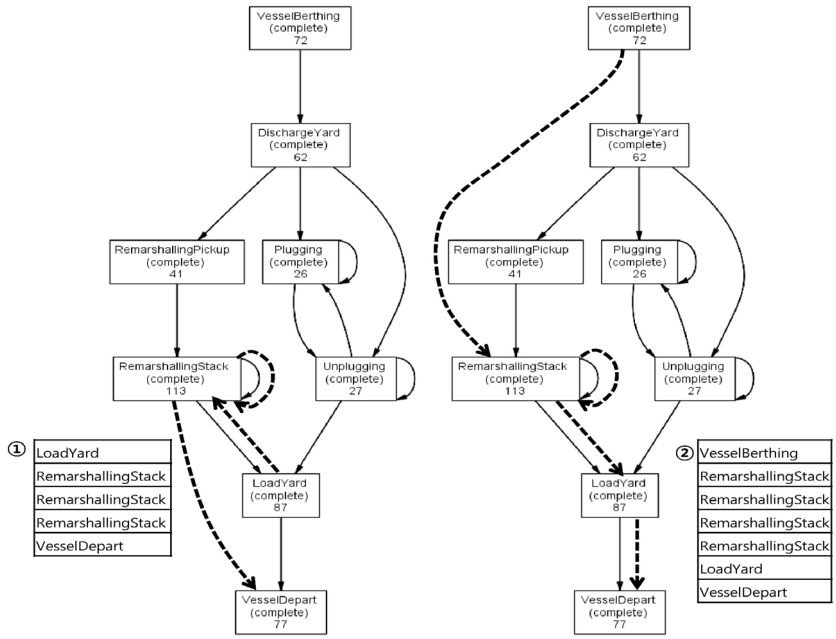
탐색된 인스턴스를 전체 프로세스 모델과 비교하였다. <Figure 6>에서 (a)는 냉동(Reefer) 컨테이너로서 하역 후에 야드에서 전원 연결 및 단전을 수 차례 반복하였다. 환적물을 포함하고 있으므로 선박에 선적을 하여야 하지만 해당 데이터에서는 처리되지 않았다. (b)는 일반 컨테이너이며, 최초에 야드에 위치하고 있는 컨테이너의 위치를 바꾸는 작업을 반복적으로 수행한 것으로 보이지만, 'RemarshallingPickup'이 없이 'RemarshallingStack'을 반복적으로 수행하였다. 픽업(Pickup)과 스택(Stack) 작업이 쌍으로 이루어져야 함에도 불구하고 누락된 작업이 있으므로, 데이터의 불완전성을 보여주고 있다. 또한 이미 선적하여 출항한 'VesselDepart' 이후에 터미널에서 발생한 이벤트를 발견하였다.

<Figure 7>에서 LAPID의 결과를 다른 API 탐지법인 DAPID의 결과와 비교하여 보았다. DAPID에서는 두 가지 조건(r, π)에서 0과 1로 탐색된 값의 합계를 높은 순위로 정렬하였다[13]. LAPID는 k가 30인 경우에 LOF가 높은 순위로 정렬하였다. 비교해보니 양쪽에서 모두 높은 순위로 API임이 확인된 인스턴스를 제외하고, 네 가지의 특징을 발견할 수 있었다.

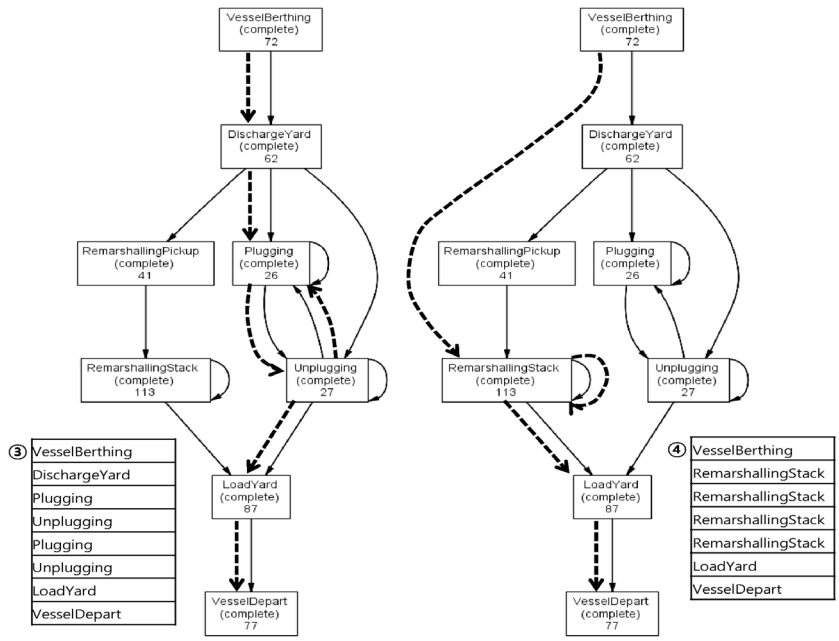
	DAPID	Results
⑤	Container058	80
	Container041	80
	Container062	77
①	Container035	76
	Container037	76
	Container054	76
③	Container055	72
	Container056	72
	Container059	72
	Container063	72
	Container064	72
	Container022	71
	Container033	70
	Container077	70
	Container072	69
④	Container011	66
	Container012	66
	Container015	66
	Container023	66
	Container065	66

LAPID	k=30	
Container058	4.1484	⑤
Container041	3.6188	
Container059	3.0281	
Container062	3.0109	
Container054	2.9301	
Container011	2.5824	④
Container012	2.5824	
Container015	2.5824	
Container023	2.5824	
Container065	2.5824	
Container067	2.5824	②
Container070	2.5824	
Container071	2.5824	
Container078	2.5824	
Container040	2.4889	
Container033	2.3192	
Container077	2.3192	
Container055	2.2922	③
Container056	2.2922	
Container063	2.2922	

〈Figure 7〉 Comparison of LAPID & DAPID in the Top 20



(a) Task Path of Container 037 (b) Task Path of Container 067



(c) Task Path of Container 055 (d) Task Path of Container 011

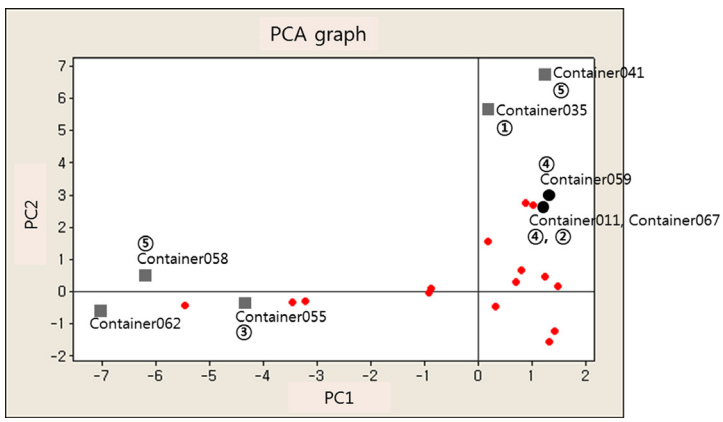
〈Figure 8〉 Difference between LAPID and DAPID

DAPID에서 20위 순위권에 있었으나 LAPID에서 제외된 경우, 반대로 LAPID에서 20위 순위권에 있었으나 DAPID에서 제외된 경우, 순위가 변동한 경우 등이었다.

<Figure 7>으로부터 구해진 네 가지 특징을 <Figure 8>에서 프로세스 모델을 통해 분석해 보았다. <Figure 8>의 (a)의 경우는 야드에 적치하는 'LoadYard' 작업 후에 재배치를 위한 픽업 작업인 'RemarshallingPickup'없이 쌓는 작업인 'RemarshallingStack'이 반복적으로 발생하였다. 이후에 선박에 적하하는 'LoadYard' 작업 없이 선박이 출항하는 'VesselDepart'가 발생하였다. 해당 프로세스 인스턴스는 전체 프로세스 모델과 확연히 다른 흐름을 보이고 있다. (b)는 선박이 입항한 후에 양하를 위한 액티비티 없이 'RemarshallingStack'을 수행하였다. (c)는 'Plugging', 'Unplugging'을 포함한 냉동 컨테이너이다. (d)는 (b)와 같은 프로세스를 나타내고 있으며, 전체 프로세스 모델과 비교하여 시작과 끝을 동일하게 수행하고 있으나 세부적인 액티비티-릴레이션에서 차이를 보이고 있다. 따라서, LAPID는 기존의 DAPID에서 찾지 못한 (b)와 (d)를 찾아냄으로써 기존 연구와의 차별되는 점이 있다.

LAPID와 DAPID의 차이점을 보이기 위해 주성분 분석(Principal component analysis, PCA)을 <Figure 9>와 같이 수행하였다. 탐색된 DAPID의 결과는 <Figure 9>에서 ①, ⑤의 경우와 같이 거리 값이 큰 인스턴스로서 가장 외곽에 위치하고 있다. 이에 반해 LAPID의 결과인 ②, ④의 경우에는 인접한 인스턴스가 존재하고 있지만 상대적인 밀도 차이가 존재하기 때문에 API로 탐색되었다. ③의 경우는 LAPID에서의 순위가 DAPID에서보다 낮게 나오는 인스턴스로서 외곽에 가깝게 위치하고 있다.

분석된 정보는 다음과 같이 세 가지 원인을 가질 수 있다. 첫째, 탐색된 API가 발생한 원인이 단순한 입력자의 입력 오류일 수 있다. 둘째, 시스템의 심각한 버그로 인해 반복적으로 발생하고 있을 수 있다. 셋째, 분석을 위해 고려된 데이터가 전처리 과정에서의 문제점을 포함할 수 있다. 이러한 원인들이 분석을 필요로 하게 만들었고, 더욱 세밀한 분석을 위해 전



〈Figure 9〉 PCA Result for Representing Process Instances of Containers

문가의 역할이 커지고 있으며, 보다 신속한 분석을 위한 도구가 필요한 이유이다. 직관적으로 빅데이터를 분석하는 것에는 한계가 있으며, 보다 효율적인 분석을 돕기 위한 방법으로서 본 논문의 방법론이 기여할 수 있다.

6. 결 론

시스템을 통해 자동으로 쌓여가는 데이터를 분석하기 위한 시도 및 연구가 꾸준히 있어왔으며, 많은 성과가 있었다. 본 논문은 프로세스 마이닝 연구의 일환으로 실제 컨테이너 터미널에서 발생하는 컨테이너 처리 작업을 분석하였다.

컨테이너 터미널의 이벤트 로그로부터 인스턴스 단위의 이상치를 찾는 문제에 있어서 LAPID와 DAPID는 다른 결과를 내포하고 있었다. 이러한 결과는 각각의 탐색법이 활용될 수 있는 인스턴스의 범주가 다르다는 것을 보여주고 있다. DAPID는 도출된 일반적인 프로세스 모델과 비교하여 누락된 액티비티-릴레이션이 많고, 반복적인 작업이 많이 포함된 인스턴스를 탐색하였다. 그에 반해 LAPID는 전체 프로세스 모델과 유사한 흐름을 보이고는 있지만 액티비티-릴레이션의 존재유무와 발생 회수가 상대적인 밀도 차이를 보이는 인스턴스를 탐지하였다. 따라서, 사례연구에서와 같이 인스턴스가 유사한 프로세스를 가지고 있으면서 다양하게 변형된 액티비티-릴레이션을 포함하고 있다면, LAPID를 사용하여 API를 탐색하는 것이 용이하다.

해양 물동량이 급속히 증가하는 환경에서 컨테이너 터미널의 물류 개선점을 발견하는 것은 경쟁력을 높일 수 있는 중요한 활동이다. 본 논문에서 제안한 LAPID를 통해 비정상 프로세스를 보이는 물류문제를 개선한다면 빠른 의사결정에 도움을 줄 수 있다. 또한, 보다 복잡한 비즈니스 영역에서도 로그 데이터만으로 개별 인스턴스의 이상여부를 판별할 수 있으므로 인스턴스를 추적하기 위한 시스템에 유용하게 적용할 수 있다.

References

[1] Agarwal, B. and Mittal, N., “Hybrid Approach for Detection of Anomaly Network Traffic using Data Mining Techniques,” Procedia Technology, Vol. 6, pp. 996-1003, 2012.

[2] Aggarwal, C. C., “Outlier analysis: Springer Science & Business Media,” 2013.

[3] Agrawal, R. and Srikant, R., “Mining sequential patterns,” in Data Engineering, 1995. Proceedings of the Eleventh International Conference on, pp. 3-14, 1995.

[4] Bezerra, F. and Wainer, J., “Algorithms for anomaly detection of traces in logs of process aware information systems,” Information Systems, Vol. 38, No. 1, pp. 33-44, 2013.

[5] Bhaduri, K., Matthews, B. L., and Giannella, C. R., “Algorithms for speeding up distance-based outlier detection,” in Proceedings of the 17th ACM SIGKDD international conference on Knowledge Discovery and Data Mining, pp. 859-867, 2011.

[6] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J., "LOF: identifying density-based local outliers," in ACM Sigmod Record, pp. 93-104, 2000.

[7] Chandola, V., Banerjee, A., and Kumar, V., "Anomaly detection: A survey," ACM Computing Surveys(CSUR), Vol. 41, No. 3, p. 15, 2009.

[8] Chen, S., Wang, W., and van Zuylen, H., "A comparison of outlier detection algorithms for ITS data," Expert Systems with Applications, Vol. 37, No. 2, pp. 1169-1178, 2010.

[9] Deza, M. M. and Deza, E., "Encyclopedia of distances: Springer," 2009.

[10] Du, W., Fang, L., and Peng, N., "LAD: Localization anomaly detection for wireless sensor networks," Journal of Parallel and Distributed Computing, Vol. 66, No. 7, pp. 874-886, 2006.

[11] Han, B., Jiang, L., and Cai, H., "Abnormal Process Instances Identification Method in Healthcare Environment," in Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on, pp. 1387-1392, 2011.

[12] Han, J., Kamber, M., and Pei, J., "Data mining: concepts and techniques: Morgan Kaufmann," 2012.

[13] Jeon, D., Bae, H., and Pulshashi, I. R., "Detection of Anomaly Process Instances using the arc matrix," BSC lab, Pusan National Univ., 2015.

[14] Kang, B. Y., Kim, D. S., and Kang, S. H., "Extended KNN Imputation Based LOF Prediction Algorithm for Real-time Business Process Monitoring Method," The Jounal of Society for e-Business Studies, Vol. 15, No. 4, pp. 303-317, 2010.

[15] Kim, H. K. and Shin, K. S., "Analysis and Improvement of Stocking and Releasing Processes in Logistics Warehouse Using Process Mining Approach," Journal of the Korean Operations Research and Management Science Society, Vol. 39, No. 4, pp. 1-17, 2014.

[16] Kim, K. H., Oh, K. H., Lee, Y. K., and Jung, J. Y., "Discovery of Travel Patterns in Seoul Metropolitan Subway Using Big Data of Smart Card Transaction Systems," The Journal of Society for e-Business Studies, Vol. 18, No. 3, pp. 211-222, 2013.

[17] Kim, S., Cho, N. W., Kang, S. H., "Density-based Outlier Detection for Very Large Data," Journal of the Korean Operations Research and Management Science Society, Vol. 35, No. 2, pp. 71-88, 2010.

[18] Kovach, S. and Ruggiero, W. V., "Online banking fraud detection based on local and global behavior," in ICDS 2011, The Fifth International Conference on Digital Society, pp. 166-171, 2011.

[19] Lee, J. S., Kang, B. Y., and Kang, S. H., "The Use of Local Outlier Factor(LOF) for Improving Performance of Independent Component Analysis(ICA) based Statistical Process Control(SPC)," Journal of the

Korean Operations Research and Management Science Society, Vol. 36, No. 1, pp. 39-55, 2011.

[20] Levenshtein, V. I., "Binary codes capable of correcting deletions, insertions and reversals," in Soviet physics doklady, p. 707, 1966.

[21] Lin, S. and Brown, D. E., "An outlier-based data association method for linking criminal incidents," Decision Support Systems, Vol. 41, No. 3, pp. 604-615, 2006.

[22] Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., and Sun, X., "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," Decision Support Systems, Vol. 50, No. 3, pp. 559-569, 2011.

[23] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., and Dayal, U. et al., "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," in International Conference on Knowledge Discovery in Databases and Data Mining, pp. 215-224, 2001.

[24] Potter, C., TAN, P. N., Steinbach, M., Klooster, S., Kumar, V., and Myneni, R. et al., "Major disturbance events in terrestrial ecosystems detected using global satellite data sets," Global Change Biology, Vol. 9, No. 7, pp. 1005-1021, 2003.

[25] Purarjomandlangrudi, A., Ghapanchi, A. H., and Esmalifalak, M., "A data mining approach for fault diagnosis: An application of anomaly detection algorithm," Measurement, Vol. 55, pp. 343-352, 2014.

[26] Shyur, H.-J., Jou, C., and Chang, K., "A data mining approach to discovering reliable sequential patterns," Journal of Systems and Software, Vol. 86, No. 8, pp. 2196-2203, 2013.

[27] Van der Aalst, W. M. P. and de Medeiros, A. K. A., "Process Mining and Security: Detecting Anomalous Process Executions and Checking Process Conformance," Electronic Notes in Theoretical Computer Science, Vol. 121, pp. 3-21, 2005.

[28] Van der Aalst, W. M., Discovery, "Conformance and Enhancement of Business Processes: Springer," 2011.

[29] Van Der Aalst, W., Adriansyah, A., de Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T. et al., "Process mining manifesto," in Business process management workshops, pp. 169-194, 2012.

[30] Xiong, W., Hu, H., Xiong, N., Yang, L. T., Peng, W.-C., Wang, X. et al., "Anomaly secure detection methods by analyzing dynamic characteristics of the network traffic in cloud communications," Information Sciences, Vol. 258, pp. 403-415, 2014.

저 자 소 개

전대욱 (E-mail: jdw@pusan.ac.kr)
2014년 부산대학교 산업공학과 (석・박사 통합수료)
현재 부산대학교 산업공학과 수료 후 연구생
관심분야 이상치 탐색, 프로세스 마이닝, 데이터 마이닝 등

배혜림 (E-mail: hrbae@pusan.ac.kr)
1998년 서울대학교 산업공학과 (석사)
2002년 서울대학교 산업공학과 (박사)
현재 부산대학교 산업공학과 교수
관심분야 프로세스 마이닝, 빅 데이터, BPM(Business Process Management), Multi organizational Process Management in e-Business Environments, Enterprise Information System Integration e-SCM & e-CRM 등