

Extracting Overlay Invariants of Distributed Systems for Autonomic System Management

Hanhuai Shan
University of Minnesota, Twin Cities
Minneapolis, MN
shan@cs.umn.edu

Guofei Jiang and Kenji Yoshihira
NEC Laboratories America
Princeton, NJ
{gfi, kenji}@nec-labs.com

Abstract—Many large-scale distributed systems have been built with great complexity to run Internet services. Due to the heterogeneity and dynamics of complex systems, it is very difficult to characterize their behavior precisely for system management. While we collect large amount of monitoring data from distributed systems as system observables, it is hard for us to interpret the data without constructing reasonable system models. Our previous work proposed algorithms to extract invariants from monitoring data to profile complex systems. However, such invariants are extracted between pairwise system measurements but not among multiple measurements. Based on minimal redundancy maximal relevance subset selection and least angle regression, this paper proposes an efficient algorithm to automatically extract *overlay invariants* from the layer of pairwise invariant networks. The overlay invariants link separated pairwise invariant subnets and enable us to support many system management tasks such as fault detection and capacity planning. Experimental results from synthetic data and real commercial systems are also included to demonstrate the effectiveness and efficiency of our algorithm.

Keywords—System management, Invariant networks, Algorithms, Least angle regression

I. INTRODUCTION

In recent ten years, we have witnessed the unprecedented success of Internet services such as Google.com and Amazon.com. Many large-scale distributed systems have been built with great complexity to run such Internet services. These large-scale systems usually consist of thousands of components including operating systems, application software, servers, networking and storage devices, and each component may also has its specific configuration. Such heterogeneity becomes an important factor of system complexity and increases the difficulty to generalize management practices across systems with different functionality and architecture. In addition, the dynamic nature of the system also increases its complexity. For example, the user workloads are always changing, the interactions among different components vary from time to time, and the software and hardware components are also frequently replaced and upgraded. Therefore, it is very difficult to characterize the behavior of a complex system precisely.

In practice, large amount of monitoring data are collected from distributed systems as observables to track system states. CPU utilization and network traffic statistics are common examples of such data. However, without reasonable system models, it is hard to interpret and reason such monitoring data for system management. Meanwhile, thousands of heterogenous components in large-scale systems may be connected by thousands of different models. It is formidable to manually build such a large number of models for system management. Therefore we need to develop a general approach that can automatically extract models from system measurements such as the CPU utilization, memory and disk I/O. In this paper, system measurements refer to the intensities of system monitoring metrics (e.g. the percentage of CPU utilization), which are usually driven to go up and down by the volume of external workloads. Since these metrics are monitored with sampling intervals, essentially we collect the time series of these metrics (i.e. system measurements) along time.

Our previous work [1] proposed an exhaustive search algorithm to extract *pairwise* invariant relationships, referred to as (pairwise) *invariants* between system measurements. Since many system measurements respond to the volume of workload accordingly, their time series often hold linear relationships with each other. Such pairwise relationships are in a form of a linear regression model as $\mathbf{x}_1 = g(\mathbf{x}_2) = w\mathbf{x}_2 + w_0$ ¹, where \mathbf{x}_1 and \mathbf{x}_2 are system measurements. If the modeled relationships hold all the time, they are regarded as the invariants of the underlying system. System measurements are connected by these invariants to form a pairwise invariant network as shown in the bottom part of Figure 1, where each node denotes one measurement, and a solid line between two nodes denotes an invariant between these two. Although it is hard to characterize complex systems in a holistic way, each invariant is able to capture some local properties of its related components. Therefore if we can discover a large number of invariants involving different measurements, the whole system can be well characterized by the invariant network. While the

¹We omit the error term for brevity.

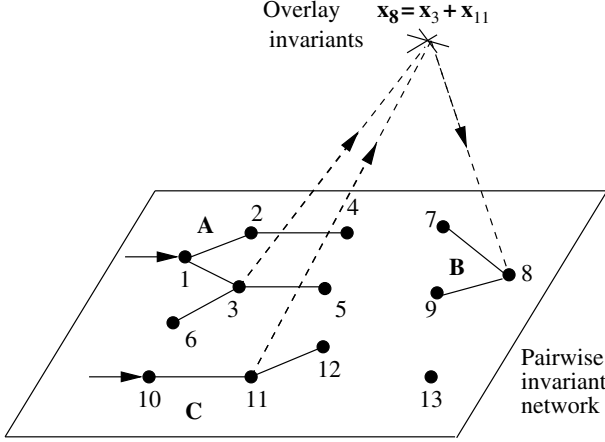


Figure 1. Pairwise and overlay invariants.

pairwise invariant network in [1] works reasonably well for such purpose, they have the following two limitations:

First, a pairwise invariant only captures the relationship between two measurements, while some relationships might involve multiple measurements. In large-scale distributed systems, we often deploy multiple replicas to scale up applications. For example, as shown in Figure 2, user requests are processed by three parallel web servers, which forward their results to a single application server for further processing. The resource usage of the application server is proportional to the weighted summation of three web servers' workloads. Therefore, it is common to have the summing up/splitting up relationships among multiple components. Such relationships motivate us to search the invariants among multiple measurements.

Second, pairwise invariant networks may not have enough connectivity to build the relationships among all components, while such connectivity of invariant networks could be important for system management tasks such as capacity planning [2]. As shown in Figure 1, the pairwise invariant networks contain several disconnected subnets (denoted as A, B and C). Our previous work [2] can only propagate the relationship within a subnet. For example, within the subnet A in Figure 1, assuming that $\mathbf{x}_2 = g(\mathbf{x}_1)$ and $\mathbf{x}_4 = h(\mathbf{x}_2)$, we can derive $\mathbf{x}_4 = h(g(\mathbf{x}_1))$. However, there is no relationship between the nodes/measurements² in A and the nodes in B or C, since there is no connection between A and B or A and C.

Introducing invariants with multiple measurements overcomes the above two drawbacks. First, they are more appropriate to model the summing up/splitting up relationship among multiple components. Second, they might be able to connect originally separated subnets. For example, in Figure 1, the invariant $\mathbf{x}_8 = \mathbf{x}_3 + \mathbf{x}_{11}$ connects A, B and C, as shown by the dotted arrows. Through the pairwise invariants

²We will use "measurements" and "nodes" interchangeably.

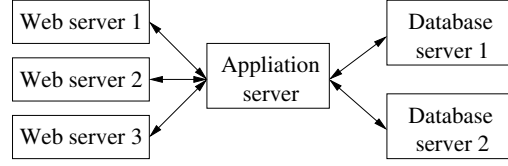


Figure 2. Summing up/splitting up relationships.

within each subnet, we may also build the relationship among other nodes in A, B and C through propagation. For example, if we know $\mathbf{x}_3 = 2\mathbf{x}_6$, $\mathbf{x}_{11} = \mathbf{x}_{12} + 5$, and $\mathbf{x}_8 = 3\mathbf{x}_7$, we can have another relationship as $3\mathbf{x}_7 = 2\mathbf{x}_6 + \mathbf{x}_{12} + 5$. The invariants like $\mathbf{x}_8 = \mathbf{x}_3 + \mathbf{x}_{11}$ are on the upper layer of pairwise invariant networks, acting like bridges to connect originally separated subnets. We therefore refer to these invariants as *overlay invariants*. In this paper, we propose an algorithm to automatically extract overlay invariants, which can characterize the relationships among multiple measurements and significantly improve the connectivity of invariant network.

The rest of the paper is organized as follows: In Section II, we give a brief overview of the preliminaries and related work. In Section III, we propose the algorithm for extracting overlay invariants. Section IV discusses the use of such invariants for autonomic system management. We present the experimental results in Section V and conclude in Section VI.

II. BACKGROUND AND RELATED WORK

A. Background

We give a brief introduction on the background of our algorithm: pairwise invariant networks and least angle regression algorithm.

1) *Pairwise invariant networks*: Our previous work [1][2] proposed an algorithm to automatically extract pairwise linear relationships between system measurements. For simplicity, we use the same notation \mathbf{x}_i for both the name and the value of resource consumption measurements, such as CPU usage, memory usage, network traffic volume, etc.. For every pair of measurements $(\mathbf{x}_i, \mathbf{x}_j)$, a linear model is constructed in the form of $\mathbf{x}_i = w_1\mathbf{x}_j + w_0$ and evaluated with a *fitness score*. For any measurement $\mathbf{x} = x(1) \dots x(T)$ over time $t = 1 \dots T$, given the estimation $\hat{\mathbf{x}} = \hat{x}(1) \dots \hat{x}(T)$ generated from the model, the fitness score is defined as

$$f = \left[1 - \sqrt{\frac{\sum_{t=1}^T |x(t) - \hat{x}(t)|^2}{\sum_{t=1}^T |x(t) - \bar{x}|^2}} \right], \quad (1)$$

where \bar{x} is the mean of real $x(t)$ over $t = 1 \dots T$. The range of fitness score is $(-\infty, 1]$. A higher fitness score indicates a better pairwise model $\mathbf{x}_i = w_1\mathbf{x}_j + w_0$. We further evaluate the models with many windows of new monitoring data. Only those models with fitness scores higher than a threshold in all time windows are considered as *pairwise*

invariants. Since most resource consumption measurements are driven to go up and down by the same external factor - the volume of user workload, we verified that such invariant relationships widely exist in large computing systems [1]. As shown in Figure 1, the combination of measurements (nodes) and invariants (edges) form a pairwise invariant network.

2) *Least angle regression*: In this paper we use least angle regression (LAR) [3] for variable selection. LAR selects the relevant inputs for predicting the response y from a set of the candidate inputs, and builds a linear relationship between y and the inputs. Given a response, LAR selects one input at a time. The selection is based on the correlation between the candidate inputs and the response, as well as the correlation between the candidate inputs and the previous selections. Therefore, the first selected input is the most important one in term of determining the value of response, and it affects all the following selections. The result of LAR is a regression model as

$$\mathbf{y} = w_0 + w_1\mathbf{x}_{j_1} + \dots + w_s\mathbf{x}_{j_s} ,$$

where $\mathbf{x}_{j_1} \dots \mathbf{x}_{j_s}$ are the selected inputs, and $w_0 \dots w_{j_s}$ are the coefficients. For the details of the algorithm, please refer to [3].

LAR is closely related to least absolute shrinkage and selection operator (Lasso) [4], which minimizes the residual sum of squares, subject to a constraint on the l_1 norm of regression coefficients, i.e.,

$$\begin{aligned} \min \|\mathbf{y} - (w_0 + \sum_j w_j \mathbf{x}_j)\|_2^2 \\ \text{subject to } \sum_j |w_j| \leq \tau . \end{aligned}$$

The l_1 penalized regression forces most of the coefficients w_j to become zero. Only the inputs with non-zero coefficients are selected for regression. These inputs are considered as factors that really affect the response.

B. Related work

Queueing models have been widely used for system performance analysis. Uргаonkar et al. [5] used a closed queueing network to model multi-tier Internet applications and used mean value analysis to calculate the response time of multi-tier distributed systems. Stewart et al. [6] characterized workloads with a transaction mix vector and exploited nonstationarity for performance prediction in distributed systems. However, it is not clear how to profile large-scale complex systems with queueing models. In this paper, we automatically extract overlay invariant networks as well as pairwise invariant networks from system monitoring data to support system management.

Machine learning algorithms have also been used in system modeling. Cohen et.al. [7] used a tree-augmented naive (TAN) Bayesian network to learn the probabilistic relationship between SLA violations and resource usages.

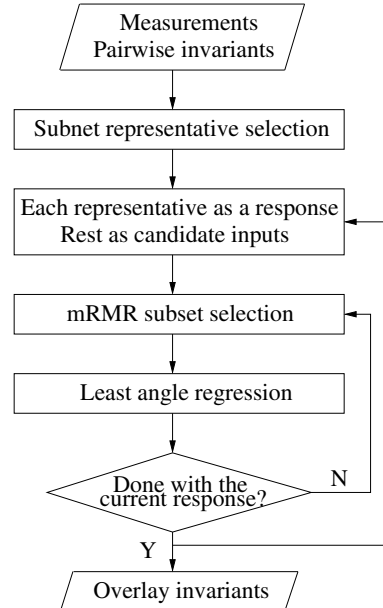


Figure 3. Workflow of extracting overlay invariants.

They further used this learned Bayesian network to identify performance bottlenecks during SLA violations. In the Elba project, Parekh et.al. [8] compared the TAN Bayesian network with other classifiers like decision trees in performance bottle detection. In addition, regression models have also become popular in system performance analysis. Our previous work [2] proposes an exhaustive search algorithm to extract pairwise relationships between system measurements for capacity planning. Given a workload, we follow the invariant network to estimate the capacity needs of various components. Jiang et al. [9] proposes a greedy algorithm to identify three-variable relationships based on ordinary least square regression guided by non-constant error variance. Zhang et al. [10] proposed a regression-based analytical model for performance analysis of multi-tier applications. In this paper, we take a systematic and automated approach to search the multiple-variable relationships across the subnets in pairwise invariant networks for system management purpose.

III. EXTRACTING OVERLAY INVARIANTS

As discussed in Section II-A1, we extracted pairwise invariants from system measurements (system monitoring data). The combination of these pairwise invariants forms a pairwise invariant network as shown in the bottom part of Figure 1, where each node represents a measurement and each solid line represents a pairwise invariant. Since the pairwise invariant does not exist in every pair of measurements, as we can see from Figure 1, the pairwise invariant network usually contains several separated subnets and a number of isolated nodes. In this section, we propose an algorithm to extract *overlay invariants*, which connect subnets as well as

Algorithm 1 Minimal redundancy maximal relevance

Input: Response measurement \mathbf{x}_i

 Candidate inputs $\{\mathbf{x}_1 \dots \mathbf{x}_{i-1} \mathbf{x}_{i+1} \dots \mathbf{x}_n\}$
 $f(\cdot, \cdot)$ between each pair of measurements
 Number of inputs to be selected k
Output: The selected mRMR Set S

- 1: Add \mathbf{x}_{j_1} with the largest $f(\mathbf{x}_i, \mathbf{x}_{j_1})$ to $S = \emptyset$.
 - 2: **for** $c = 2$ to k **do**
 - 3: $x_{j_c} = \operatorname{argmax}_{x_j \in X-S} \left(f(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{|S|} \sum_{\mathbf{x}_l \in S} f(\mathbf{x}_l, \mathbf{x}_j) \right)$
 - 4: Add \mathbf{x}_{j_c} to S .
 - 5: **end for**
-

isolated nodes. In particular, the overlay invariants are in a form of a regression model

$$\mathbf{x}_i = w_0 + w_1 \mathbf{x}_{j_1} + w_2 \mathbf{x}_{j_2} + \dots + w_s \mathbf{x}_{j_s}, \quad (2)$$

where $\{\mathbf{x}_{j_1} \dots \mathbf{x}_{j_s}\}$ are in different subnets of pairwise invariant network, so the overlay invariant connects originally separated subnets together. To discover such regression models, our goal includes two aspects:

- 1) Discover the set of measurements $\{\mathbf{x}_{j_1} \dots \mathbf{x}_{j_s}\}$ which together have a relationship with the response variable \mathbf{x}_i .
- 2) Estimate the coefficients $\{w_0, \dots, w_s\}$, such that we build a mathematic model between \mathbf{x}_i and $\{\mathbf{x}_{j_1} \dots \mathbf{x}_{j_s}\}$.

The main workflow of our algorithm is illustrated in Figure 3: Given the pairwise invariant network, we first select a representative node for each subnet. Each representative is then used as the response variable in turn, and the rest are used as the candidate inputs. We run “minimal redundancy maximal relevance” (mRMR) subset selection [11] and LAR [3] alternately to extract overlay invariants. The following subsections describe the details of each step respectively.

A. Selecting subnet representatives

All measurements within a pairwise subnet have strong linear relationships with each other, directly or indirectly. Therefore, as we have discussed, if one node in subnet A is connected by an overlay invariant with nodes in subnets B and C, we can build connection among all nodes in A, B and C through propagation. In that case, we can pick one measurement from each subnet as the representative and find overlay invariants among the representatives. Later the representative serves as a gateway to reach other nodes in its subnet. While there are thousands or even millions of measurements in large-scale systems, choosing representatives dramatically reduces the search space and speeds up the computation.

In principle, the representative could be any node in the subnet since we can reach all other nodes in the subnet

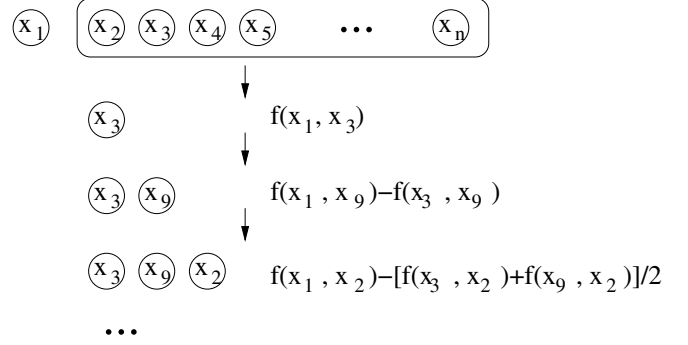


Figure 4. An example of mRMR subset selection.

through propagation. However, each invariant contains a certain amount of modeling error, and the error gets accumulated during propagation. Generally, more propagation hops needed, more estimation error will be introduced. Therefore, we have the following two criteria to choose the representatives:

- 1) In the pairwise invariant subnet, the representative should have as many immediate neighbors as possible, so the overlay invariant could be propagated to reach as many nodes as possible in one hop.
- 2) The pairwise invariants between the representative and other nodes should be accurate to reduce the propagation error. Since the accuracy is measured by the fitness score shown in Eq.1, the representative should have the highest fitness scores with all other measurements in the same subnet. To reduce the computational cost, the fitness scores with each node’s immediate neighbors are used instead.

Based on the above criteria, we take the following approach to select a representative for each subnet: For each node, we calculate the summation of fitness scores $f_{neighbor}$ with all its immediate neighbors, we then pick the node with the largest $f_{neighbor}$ as the representative for the subnet. All the selected subnet representatives and all the isolated nodes together constitute the Representative Set of the whole dataset. We only work on the Representative Set in the following steps.

B. mRMR subset selection

Given the Representative Set, to find an overlay invariant as in Eq.2, we pick one measurement each time as the response variable, and take the remaining measurements as the Candidate Set of inputs. We need to select the set of measurements which have a relationship with the response variable and determine the corresponding coefficients. In principle, as we have introduced in Section II-A2, least angle regression can give us such results directly. However, we will encounter the following two issues with a direct application of LAR:

Algorithm 2 Extracting overlay invariants

Input: Representative Set $X = \{x_1 \dots x_n\}$
 $f(\cdot, \cdot)$ between each pair of representatives
 An upper bound u on num. of inputs in invariants
 Number of times $m(=10)$ to run mRLR
 A threshold $\xi(=0.9)$ on $f(\cdot, \cdot)$ for verification

Output: Overlay invariants and corresponding $f(\cdot, \cdot)$

- 1: Take first 80% of X as the training set X^{train} , and whole X as the validation set.
- 2: **for** $i = 1$ to n **do**
- 3: Take x_i^{train} as the response variable, and $X^{train} - \{x_i^{train}\}$ as the Candidate Set C .
- 4: **for** $q = 1$ to m **do**
- 5: Use mRMR to select mRMR Set S_q of $2u$ measurements from C .
- 6: Center x_i^{train} and normalize the inputs in S_q .
- 7: Use LAR(x_i^{train}, S_q, u) to get the regression model λ_{iq} , and the first variable a on LAR's path.
- 8: $C = C - \{a\}$.
- 9: Apply λ_{iq} to X^{train} and X to get fitness score f_{iq}^{train} and f_{iq} . λ_{iq} is verified only if $f_{iq}^{train} > \xi$ and $f_{iq} > \xi$.
- 10: Mapping verified λ_{iq} to unnormalized space.
- 11: **end for**
- 12: Select invariants from verified λ_{iq} and keep the corresponding f_{iq} .
- 13: **end for**

- 1) In large-scale systems, even the Candidate Set may contain thousands of measurements. LAR tends to be very slow in such a high dimensional search space. Moreover, the size of the Candidate Set is even significantly larger if we consider any temporal dependency.
- 2) Multicollinearity will affect LAR's accuracy. Multicollinearity refers to a situation where a subset of inputs are highly correlated. It introduces redundancy and decreases prediction accuracy. In our case, for example, among 10 variables $x_1 \dots x_{10}$, if we have $x_2 = x_3 + x_4$, when we consider x_1 as the response variable and $x_2 \dots x_{10}$ as the Candidate Set, a subset of the candidate inputs, i.e. $\{x_2, x_3, x_4\}$, are highly correlated given their linear relationship. In such cases, it introduces perfect multicollinearity.

Therefore, before applying LAR, we take a subset selection to break multicollinearity and reduce the size of Candidate Set to speed up the algorithm .

Given a response variable, the criterion of subset selection is to find a set of measurements which *jointly* have the largest dependency on the response variable. While this strategy is computationally difficult to implement, we take an alternative strategy called “minimal redundancy maximal relevance” (mRMR) [11]. Its main idea is to search a

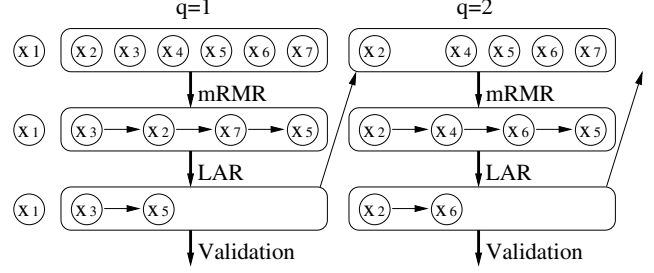


Figure 5. An example of extracting overlay invariants using mRLR.

subset which has high relevance with the response variable but low redundancy within itself. Instead of using mutual information as in [11], we use the fitness score as the metric to measure relevance and redundancy. Our algorithm based on mRMR is illustrated in Algorithm 1, where the fitness score of pairwise invariant $f(\cdot, \cdot)$ is already calculated during pairwise invariant search. At each step c , for each measurement x_j , its relevance (R_1) with the response variable x_i is the fitness score $f(x_i, x_j)$, and its redundancy (R_2) to the selected set is defined as the mean of the fitness scores with all previously selected inputs.

At each step, the measurement maximizing $R_1 - R_2$ is selected. For example, as shown in Figure 4, x_1 is the response variable and $x_2 \dots x_n$ are the Candidate Set. At step 1, x_3 is selected because it has the largest fitness score with x_1 . At step 2, x_9 is selected because it maximizes $R_1 - R_2$ where $R_1 = f(x_1, x_9)$ and $R_2 = f(x_3, x_9)$. At step 3, x_2 is selected because it maximizes $R_1 - R_2$ where $R_1 = f(x_1, x_2)$ and $R_2 = [f(x_3, x_2) + f(x_9, x_2)]/2$. We continue the process until k variables are selected.

mRMR gives us a much smaller set of candidates named as “mRMR Set”. In the next step, LAR algorithm will further remove the unrelated measurements from mRMR set. Though there is no guarantee that the mRMR subset selection will remove all multicollinearities in the Candidate set, it can greatly ease the problem.

C. Extracting invariants with mRLR

mRMR Set is fed to LAR to further reduce the number of inputs and determine the regression coefficients. However, LAR selects a set of inputs sequentially based on “least angle” rule. As we have discussed in Section II-A2, each selection is dependant on all previous selections. If we consider the order of selected inputs as “a path”, the first selected input actually affects all the rest of selection. If the first selection is wrong, the whole path would be misled. Therefore, a conservative strategy is to run LAR from different start points. The overlay invariant extraction algorithm mRLR which runs mRMR and LAR alternately is presented in Algorithm 2.

We only consider the Representative Set obtained from Section III-A as the algorithm input. The user needs to

specify the following parameters: an upper bound u on the number of inputs in each invariant, which is usually set using domain knowledge; a threshold ξ of fitness score ($\xi = 0.9$ by default) used to verify invariants; and the number of times m ($m = 10$ by default) to run mRLR for each response variable. A larger m leads to more extracted invariants but also more computational time. For each \mathbf{x}_i as a time series, we take the first 80% of data points to construct a training set X^{train} to discover the overlay invariants and use the whole dataset X as the validation set. Before we run LAR, the response variable is centered to have zero mean, and the inputs are normalized to have zero mean and unit length.

For each response \mathbf{x}_i^{train} and the corresponding Candidate Set C , we run mRMR and LAR successively. After each run, we remove the first variable on LAR's path from C and then run mRMR and LAR again, totally for m times. By removing the first input in the previous run, we actually force LAR to find a different input to start, leading to a totally different path and a regression model. Therefore, if the default LAR's path does not generate a good invariant, there are still chances to find another one in the following runs. We can track the error terms each time a new variable is selected, and remove from LAR's path the last few variables which do not significantly reduce the modeling error. The regression models with fitness scores larger than ξ on both X^{train} and X are considered as invariants. For each response variable, we can either keep all verified invariants or only pick the one with the highest fitness score. The invariants are mapped back to the unnormalized space.

In mRLR, we reduce the search space dramatically through mRMR subset selection, but meanwhile, we also explore more search space by running the algorithm multiple times under the guidance of previous search results. To illustrate the mRLR process, we give an example in Figure 5. Given a response \mathbf{x}_1 and the candidate set $\{\mathbf{x}_2 \dots \mathbf{x}_7\}$, if we set $u = 2$, the first two iterations $q = 1, 2$ are proceeded as follows: At $q = 1$, we first run mRMR and LAR to get one LAR path— $\mathbf{x}_3 \rightarrow \mathbf{x}_5$ as well as the corresponding coefficients. The extracted regression model will be validated with new data. Meantime, the first input on the previous LAR path, \mathbf{x}_3 , is removed to yield a reduced Candidate Set $\{\mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7\}$. The mRMR and LAR run again on the new set to yield a new path $\mathbf{x}_2 \rightarrow \mathbf{x}_6$. In the same way the algorithm runs for m iterations.

Due to system delay or unsynchronized data logging, the response variable \mathbf{x}_i at time t may not only depend on other variables at time t , but also their previous values at $t-1, t-2$ etc.. Algorithm 2 could be extended to incorporate such time dependency. At step 7, for each response $\mathbf{x}_i(t)$, we originally feed LAR with the measurements at the same time t from S_q , i.e., $\{\mathbf{x}_{j_1}(t) \dots \mathbf{x}_{j_{2u}}(t)\}$. When taking time dependency into consideration, we will include the inputs from time $t-d$ to t , where d is the maximum time lag for each input, i.e., we now feed LAR with $\{\mathbf{x}_1(t-d) \dots \mathbf{x}_1(t), \dots \mathbf{x}_{2u}(t-d)$

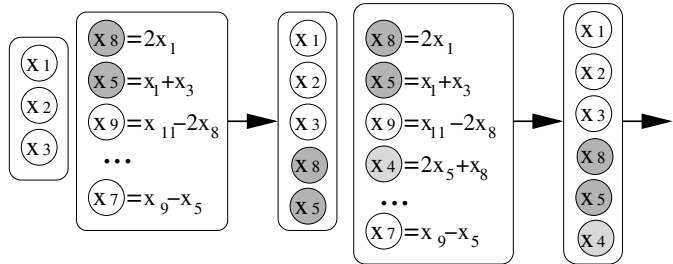


Figure 6. An example for capacity planning process

$d) \dots \mathbf{x}_{2u}(t)\}$. LAR will then choose the proper inputs at the proper time.

IV. APPLICATION OF INVARIANTS

Each invariant is able to capture some local properties of its related components, which are governed by the physical properties or logic of system components. The overlay invariants, together with the pairwise invariants, form an invariant network to effectively characterize large, dynamic and complex systems. Essentially we extract a "virtual network of invariants" from the physical systems, which keeps monitoring the activities of various system components. By tracking any changes of invariants, we can obtain a deep situation awareness on system states, which is critical for autonomous system management. Since the main purpose of this paper is to extract overlay invariants from distributed systems, we briefly discuss two applications of invariants for autonomous system management.

A. Fault detection and diagnosis

A lot of real time transaction systems require high system reliability and availability. Even minutes of service down time in transaction systems could lead to severe loss. Therefore, it is critical to perform fault detection and diagnosis effectively and efficiently. Invariants can be very useful for fault detection and diagnosis.

Given an invariant, e.g. $\mathbf{x}_1 = \mathbf{x}_2 + \mathbf{x}_3$, such invariant should always hold when system runs normally. In other words, $|x_1(t) - \hat{x}_1(t)| < \epsilon$, where x_1 is the real value at time t for node 1, $\hat{x}_1(t)$ is the predicted value for node 1 at time t according to the invariant, and ϵ is the threshold of modeling error. By keeping on checking whether the real output deviates the predicted value, i.e., whether the invariant is violated, we can support fault detection. In case the invariant is violated, by looking at the measurements involved in the invariant, and tracing the other invariants involving these measurements recursively, we may pinpoint the cause of the fault.

Since we extract a large number of invariants from monitoring data to profile complex distributed systems, we can track the change of these invariants in real-time to support fault detection and diagnosis [12]. Our invariant-based solutions were tested in many large-scale commercial

Table I
RESULTS OF LAR AND MRLR ON SYNTHETIC DATA.

	LAR			mRLR		
	precision	recall	F1-measure	precision	recall	F1-measure
$\sigma^2 = 0$	1	1	1	1	1	1
$\sigma^2 = 0.3$	1	0.97	0.98	1	1	1
$\sigma^2 = 0.6$	0.93	0.87	0.89	1	1	1
$\sigma^2 = 0.9$	0.80	0.80	0.80	0.90	0.90	0.90

systems and were proved to be very effective for problem determination. Therefore, we have commercialized our technology and developed a product - "Invariant Analyzer" [13].

B. Capacity planning

In the internet services, on one side, users expect high Quality of Service which requires service providers to deploy sufficient hardware resources. On the flip side, an oversized system with scale could significantly waste resources and increases service provider's total cost of ownership. Therefore the ideal case is to match resource assignment for each component with its capacity need in large scale distributed systems.

The overlay invariants and pairwise invariants build a relationship between the component measurements and user workloads, either directly or indirectly. In practice, we usually predict the long-term growth of user workloads with trend analysis. For example, estimating the growth of transaction volumes of an E-commerce website for next 3 months. However, it is not clear on how to map the growth of user workloads into the requirement of various system resources. Given the estimated volume of user workloads, we can estimate the capacity needs for system components using the invariants. By comparing the predicted capacity needs with the actual system resources, the service provider could upgrade or downgrade the components' capacity accordingly. For example, given the predicted volume of online customers for next 3 months, an Internet service operator can follow those invariants (i.e., the equations) to estimate the capacity needs of the backend database servers and further check their current resource configurations to decide whether to upgrade these servers.

Figure 6 shows an example to illustrate this process. Before the algorithm starts, we have three estimated volumes of incoming workloads: $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ (complex systems usually have multiple inputs), and all extracted invariants including pairwise and overlay invariants. For those invariants only taking these three available inputs, we calculate their response variables, which are \mathbf{x}_8 and \mathbf{x}_5 (shaded) in this case. \mathbf{x}_8 and \mathbf{x}_5 are then added to the set of available nodes, making another invariant with response \mathbf{x}_4 active. Similarly, \mathbf{x}_4 becomes a new available input for the next iteration. We keep on doing this process until all reachable nodes are calculated.

V. EXPERIMENTS

Since the ground truth on the invariants in real data is unknown, we run experiments on both synthetic and real data to demonstrate the effectiveness of our algorithm.

A. Results on synthetic data

For synthetic data, we generate 100 measurements and each is a time series with 125 data points. These measurements are generated from a multivariate Gaussian distribution $N(\boldsymbol{\mu}, I + \sigma^2(1 - I))$, where I is the identity matrix and $\sigma^2 \in [0, 1]$ is a scaler controlling the correlation between each pair of measurements. These measurements are used to simulate the system monitoring metrics from different components. From these measurements, we take 10 groups of 2 measurements, 10 groups of 3 measurements, and 10 groups of 4 measurements, and sum up the measurements in the same group to generate 30 invariants, whose response variables (denoted as the set Q) are also added to the measurement set. As a result, we have 130 measurements in total. In the experiments, we take different values of σ^2 as $\{0, 0.3, 0.6, 0.9\}$ to generate four synthetic datasets.

We compare invariants extraction using LAR and mRLR. For simplicity, we only extract invariants which have the measurements in Q as the response variables. For each response variable, we only keep the verified invariant with the highest fitness score. In this case, precision, recall and F1-measure are defined as follows:

$$\begin{aligned} \text{precision} &= \frac{|\text{correct invariants extracted}|}{|\text{all invariants extracted}|}, \\ \text{recall} &= \frac{|\text{correct invariants extracted}|}{|Q|}, \\ \text{F1-measure} &= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \end{aligned}$$

where $|\cdot|$ denotes the number of elements in the set, and a "correct" invariant refers to the extracted invariant with the same inputs as the real one. Table I compares the results of LAR and mRLR. When the correlation among input measurements is low, both algorithms achieve high precision and recall. mRLR gradually outperforms LAR when such a correlation increases. The F1-measure stays as high as 0.9 for mRLR even when $\sigma^2 = 0.9$. Such results indicates that when there are high correlation among the system components (which is the case in most systems since the system is a whole with multiple components functioning together), mRLR performs much better than LAR alone.

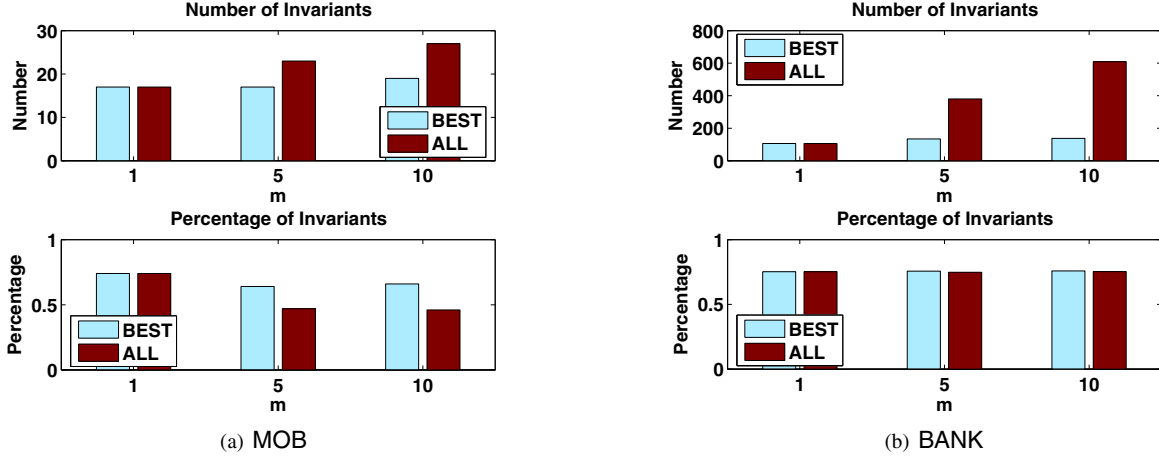


Figure 7. Number and percentage of extracted invariants with $f^{test} > 0.9$ using BEST and ALL.

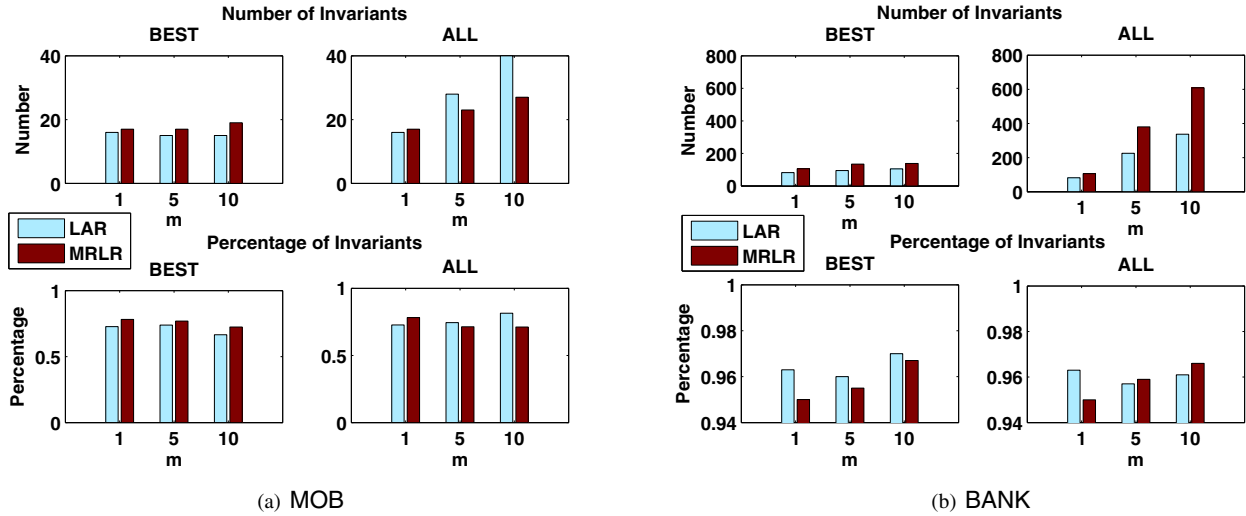


Figure 8. Number and percentage of extracted invariants with $f^{test} > 0.9$ for LAR and mRLR.

B. Results on real data

We also run experiments on two data sets collected from real commercial systems. MOB is a resource usage data set collected from a mobile infrastructure. It contains 254 measurements and each measurement is a time series with 126 data points. The data collected on two different days are used for training and testing respectively. Another resource usage data set BANK is collected from a typical three tier internet banking system. It contains 983 measurements and each is a time series with 533 data points. The measurements during time window $t = 1 : 200$ are used for training and the remaining data points during $t = 201 : 533$ are used for testing. For both datasets, pairwise invariants and their fitness scores have been extracted as in [2]. Moreover, since the sampling rate for MOB and BANK monitoring data is very low (one sample in several minutes), we do not consider

any time dependency in our experiments.

1) *mRLR using ALL and BEST*: Given the pairwise invariants, we generate the Representative Set and run mRLR with different choices of m to extract overlay invariants, which are applied to a test set to get the predicted values of the measurements. The predicted values are then compared to the real values collected from real systems to derive the fitness score f^{test} . A higher f^{test} indicates a better prediction, hence a better overlay invariant.

Given all extracted invariants for each response variable, we can either keep the one with the highest fitness score or keep all invariants with fitness scores above a threshold. Here we refer to the first strategy as “BEST” and the second as “ALL”. Note that when $m = 1$, the two strategies are the same since we only extract one invariant for each response variable. If we consider the invariants with $f^{test} > 0.9$ as the “good” invariants, the number and percentage of good

Table II
 RUNNING TIME (SEC) USING LAR AND MRLR WITH DIFFERENT CHOICES OF m .

	LAR			mRLR		
	$m = 1$	$m = 5$	$m = 10$	$m = 1$	$m = 5$	$m = 10$
MOB	19.98	95.04	248.31	9.74	44.36	79.81
BANK	495.57	2683.62	5330.72	49.59	238.53	474.22

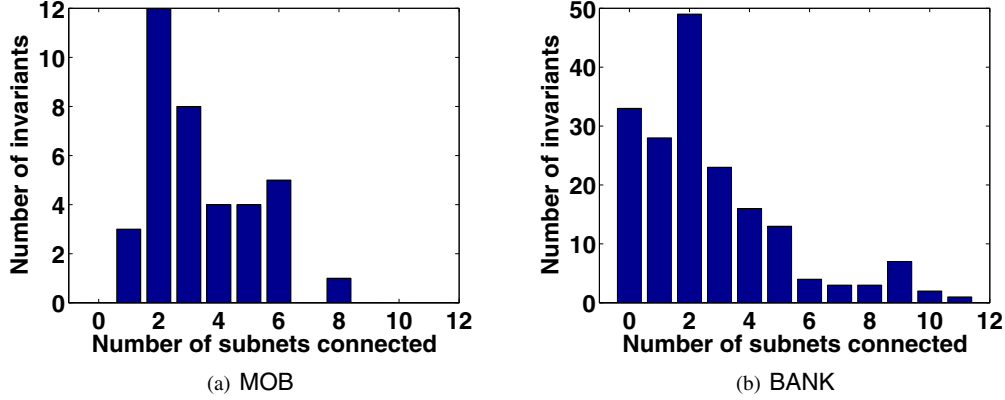


Figure 9. Histogram of connected subnets. Each bar n shows the number of invariants which connect n subnets from pairwise invariants

invariants extracted by BEST and ALL are presented in Figure 7, where “percentage” refers to the percentage of “good” invariants in the testing process among all extracted invariants from the training process. Since we do not have the ground truth for real invariants, we can not calculate the precision and recall explicitly. However, we can use the number and percentage of good invariants in Figure 7 to compare the precision and recall of different algorithms. A larger number of good invariants indicates a higher recall, and a higher percentage of good invariants indicates a higher precision. The observations are as follows:

- 1) ALL yields much more invariants than BEST, especially on BANK, i.e., ALL has a higher recall than BEST. The result also indicates that there could be multiple qualified regression models for each response variable, but with different inputs.
- 2) BEST has a higher precision than ALL on MOB and a similar precision with ALL on BANK.
- 3) Overall, BEST tends to yield a higher precision and ALL tends to yield a higher recall. Therefore, we may choose to use BEST or ALL depending on the application. If we want to connect more measurements, we may use ALL, and if we want to extract accurate invariants, we may use BEST.

2) *mRLR vs. LAR*: LAR can not be directly compared to our approach which includes extra steps in search space exploration. To have a fair comparison, we also run LAR m times for each response variable in Representative Set, following a similar way as we run mRLR, i.e., each time we force LAR to find a new path by removing the first input in the previous search. The results of “good” invariants from LAR and mRLR are presented in Figure 8. As discussed

earlier, though we can not calculate the recall and precision directly, we can compare them between two algorithms. In terms of precision, LAR and mRLR win a same number of times. In terms of recall, mRLR win most of the times, especially on BANK, where mRLR extracts a significantly larger number of good invariants using ALL. In practice, a high recall is preferable if good precision and recall can not be both achieved. This is because the incorrect invariants could be easily removed in sequential validation with more monitoring data, where we continue to collect the runtime monitoring data from 24*365 operational systems to validate invariants.

In terms of computational time, mRLR is much more efficient than LAR. The total invariant extraction time of LAR and mRLR on two datasets are given in Table II. With a same m , mRLR is 2-3 times faster on the small dataset MOB, and more than 10 times faster on the larger dataset BANK. LAR becomes significantly slower when the size of Candidate Set becomes very large. The efficiency of mRLR makes it applicable to the large amount of monitoring data from large-scale systems.

C. mRLR properties

The main goal of extracting overlay invariants is to connect separated pairwise subnets and increase connectivity. As shown in Figure 9, we analyze the number of subnets connected by each overlay invariants. Each bar with x-coordinate n denotes the number of invariants that connect n subnets. For example, the bar on $n = 0$ denotes the number of invariants only connecting the isolated nodes, the bar on $n = 1$ denotes the number of invariants connecting the isolated nodes and 1 invariant subnet, and the bar on $n = 2$

Table III
NUMBER OF MEASUREMENTS IN THE LARGEST SUBNET BY USING DIFFERENT ALGORITHMS.

	Pairwise	LAR			mRLR			Total
		$m = 1$	$m = 5$	$m = 10$	$m = 1$	$m = 5$	$m = 10$	
MOB	26	206	200	205	183	216	216	527
BANK	18	513	658	655	697	733	736	983

denotes the number of invariants connecting 2 invariant subnets. We can see that a majority of overlay invariants connect more than two subnets, which can significantly improve the reachability and coverage.

Table III shows the number of invariants in the largest subnet resulted from pairwise invariant, LAR, and mRLR, and it also shows the total number of measurements as a reference. The observations are as follows: First, using overlay invariants yields a much larger subnet than using pairwise invariants only, which indicates that overlay invariants indeed increase the connectivity of invariant networks. Second, LAR and mRLR on MOB yield similar size subnets, but mRLR on BANK yields a larger subnet, which includes more than 70% of total measurements. Therefore, mRLR is not only faster than LAR as in Table II, but it also connects more subnets. Third, even with a small number of runs (e.g. $m = 1$), mRLR can connect a large number of measurements on BANK, which makes it very efficient given the running time with different choices of m from Table II.

VI. CONCLUSION

In this paper, we proposed an algorithm to extract overlay invariants based on pairwise invariant networks. The algorithm is a combination and improvement of minimal redundancy maximal relevance subset selection and least angle regression. In particular, the algorithm first selects the representatives from each pairwise subnet, and then runs mRMR and LAR alternatively to explore the overlay invariants. The overlay invariants can connect the separated pairwise invariant subnets and significantly improve the reachability and measurement coverage of the invariant network. Experimental results from both synthetic data and real commercial systems demonstrated the effectiveness and efficiency of our algorithm. Extracted overlay invariants, together with pairwise invariants, can be used in a wide variety of applications such as fault detection and diagnosis, capacity planning, and so on. Since both the pairwise invariants and overlay invariants represents only the linear relationships among system measurements, future work includes extracting the invariants for non-linear relationships.

REFERENCES

[1] G. Jiang, H. Chen, and K. Yoshihira, "Discovering likely invariants of distributed transaction systems for autonomic system management," in *ACM International Conference on Autonomic Computing and Communications (ICAC)*, Dublin, Ireland, June 2006, pp. 199–208.

[2] G. Jiang and H. Chen and K. Yoshihira, "Profiling services for resource optimization and capacity planning in distributed systems," *Cluster Computing*, vol. 11, no. 4, pp. 313–329, 2008.

[3] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[4] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[5] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "Analytical modeling of multitier internet applications," *ACM Transactions on the Web*, vol. 1, no. 2, 2007.

[6] C. Stewart, T. Kelly, and A. Zhang, "Exploiting nonstationarity for performance prediction," *SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 31–44, 2007.

[7] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox, "Capturing, indexing, clustering, and retrieving system history," *SIGOPS Operating Systems Review*, vol. 39, no. 5, pp. 105–118, 2005.

[8] J. Parekh, G. Jung, G. Swint, C. Pu, and A. Sahai, "Comparison of performance analysis approaches for bottleneck detection in multi-tier enterprise applications," in *IEEE International Workshop on Quality of Service*, New Haven, CT, USA, 2006, pp. 302–306.

[9] M. Jiang, M. Muawar, T. Reidemeister, and P. Ward, "System monitoring with metric-correlation models: Problems and solutions," in *ACM International Conference on Autonomic Computing and Communications (ICAC)*, Barcelona, Spain, June 2009, pp. 13–22.

[10] Q. Zhang, L. Cherkasova, N. Mi, and E. Smirni, "A regression-based analytic model for capacity planning of multi-tier applications," *Cluster Computing*, vol. 11, no. 3, pp. 197–211, 2008.

[11] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance and min-redundancy," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[12] G. Jiang, H. Chen, and K. Yoshihira, "Modeling and tracking of transaction flow dynamics for fault detection in complex systems," *IEEE Trans. Dependable Sec. Comput.*, vol. 3, no. 4, 2006.

[13] NEC Invariant Analyzer, <http://www.nec.com/global/prod/invariantanalyzer/>.