Hindawi
Mobile Information Systems
Volume 2019, Article ID 9083956, 14 pages
https://doi.org/10.1155/2019/9083956

Research Article

Location-Based Test Case Prioritization for Software Embedded in Mobile Devices Using the Law of Gravitation

Xiaolin Wang,[1,2] **Hongwei Zeng**,[1] **Honghao Gao**,[3,4] **Huaikou Miao**,[1] **and Weiwei Lin**[1,2]

[1] *School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China*
[2] *Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 200444, China*
[3] *Computing Center, Shanghai University, Shanghai 200444, China*
[4] *Shanghai Key Laboratory of Intelligent Manufacturing and Robotics, Shanghai 200072, China*

Correspondence should be addressed to Honghao Gao; gaohonghao@shu.edu.cn

Received 19 July 2018; Revised 26 October 2018; Accepted 25 November 2018; Published 2 January 2019

Guest Editor: Jaegeol Yim

Considering that some intelligent software in mobile devices is related to location of sensors and devices, regression testing for it faces a major challenge. Test case prioritization (TCP), as a kind of regression test optimization technique, is beneficial to improve test efficiency. However, traditional TCP techniques may have limitations on testing intelligent software embedded in mobile devices because they do not take into account characteristics of mobile devices. This paper uses a smart mall as a scenario to design a novel location-based TCP technique for software embedded in mobile devices using the law of gravitation. First, *test gravitation* is proposed by applying the idea of universal gravitation. Second, a specific calculation model of *test gravitation* is designed for a smart mall scenario. Third, how to create a faulted test case set is designed by the pseudocode. Fourth, a location-based TCP using the law of gravitation algorithm is proposed, which utilizes test case information, fault information, and location information to prioritize test cases. Finally, an empirical evaluation is presented by using one industrial project. The observation, underlying the experimental results, is that our proposed TCP approach performs better than traditional TCP techniques. In addition, besides location information, the level of devices is also an important factor which affects the prioritization efficiency.

1. Introduction

Nowadays, the Internet of Things (IoT) develops more and more widely [1]. It is based on wireless sensor networks (WSNs) which combine intelligent software and sensor devices and makes smart home and smart city possible [2, 3]. With the development of hardware (chips) and software (intelligent systems), smart mobile devices (such as smart dust) are gradually emerging, which integrate sensors, processors, intelligent software, and communications. Smart mobile devices not only have an ability to transmit and monitor information but also perform sophisticated intelligent information processing and intelligent prediction by the intelligent software [4] and location-based service (LBS) [5]. The software in each device has its specific functions. For example, some devices are used to monitor and process temperature information and others are used to process population information. Devices are provided with location-dependent information and interact with other devices in a location-dependent way. It is location information and software complexity of devices that make software testing face a major challenge in IoT.

Regression testing, reusing test suites, is performed on a modified program to install confidence that the system behaves correctly and that modifications have not adversely affected unchanged portions of the program [6]. Test case prioritization (TCP), sorting test cases depending on some criteria, is a way to increase the efficiency of regression testing [7]. It aims at improving the rate of fault detection. Traditional TCP techniques mainly focus on the algorithm design for testing software to improve test prioritization efficiency. However, in IoT, traditional TCP techniques have limitations because they do not take into account the characteristics of hardware devices, such as location information.

The law of gravitation, according to Newton's Philosophiae Naturalis Principia Mathematica [8], indicates that there is a force of gravitational attraction existing between any two objects, which is given by the following equation:

$$F = G\frac{m_1 m_2}{r^2}, \tag{1}$$

where G is the universal gravitational constant, m_1 is the mass of one object, m_2 is the mass of the other object, r is the radius of separation between the center of masses of each object, and F is the force of attraction between two objects. The universal gravitation has been applied to the field of data analysis. For example, many research studies make data gravitation (simulating the universal gravitation) applicable to machine learning [9–11]. In IoT, if we can utilize the law of gravitation to prioritize test cases, will it improve the test efficiency?

In this paper, a new location-based TCP using the law of gravitation technique is developed to solve TCP problem of software embedded in mobile devices. This technique is designed for adapting to a smart mall scenario. It is not just a test case prioritization approach but additionally enables to make characteristics of devices utilized, thereby allowing the order of test cases to be beneficial for test efforts. *Test gravitation* is defined in this technique. Under this definition, a specific calculation model of *test gravitation* for a smart mall scenario is designed. First, it calculates the masses of each test case and each faulted test case. The creation of a faulted test case set is related with the occurred faults which detected by those preselected test cases that test different-location-area device representatives. Then, the distance between two specific test cases can be calculated according to location information of devices. For each test case, *test gravitation* is calculated from this test case to each faulted test case. Finally, test cases are prioritized based on *test gravitation*.

The contributions of this work include the following:

(i) *Test gravitation* is proposed based on the law of gravitation. A specific calculation model of *test gravitation* adapted to a smart mall scenario is given. Specially, the creation of the faulted test case set used in the calculation of *test gravitation* is designed in detail.

(ii) A location-based TCP using the law of gravitation technique is proposed, and its algorithm is designed by the pseudocode. Its feasibility is illustrated with a small example.

(iii) An empirical evaluation is presented by using one industrial project. In addition, it discusses whether different evaluation metrics (with or without considering severities of faults) will influence the experimental conclusions. It is also discussed what factors affect the prioritization efficiency.

The rest of this paper is organized as follows: Section 2 describes test case prioritization problem, traditional TCP techniques, special TCP techniques, and TCP problem in a smart mall scenario. Section 3 presents a location-based TCP using the law of gravitation method and simulates its feasibility with an example. Section 4 describes an empirical evaluation and analyzes the results. Section 5 discusses some related work on test case prioritization and mobile application testing. Finally, the conclusions and future work are given in Section 6.

2. Background

2.1. Test Case Prioritization Methodology. Regression testing, attempting to validate modified version P' of the original program P, checks the results for conformance with requirements [12]. Many techniques have been proposed to improve the cost-effectiveness of regression testing. Test case prioritization is one of these approaches, which rearranges test cases to increase the rate of fault detection during the whole regression testing.

Test case prioritization problem is a research hotspot in the field of software testing. It sorts test cases by using some criteria to detect more faults as fast as possible. A complete definition of TCP problem was first proposed by Rothermel et al. [13]:

Given a test suite already selected (T), the set of all possible prioritizations (orderings) of T (PT), and an objective function from PT to the real numbers (f), which yields an award value for that ordering.

Problem. Find $T' \in PT$ such that $(\forall T'')(T'' \in PT)(T'' \neq T')[f(T') \geq f(T'')]$.

Many test case prioritization techniques have been proposed during the past two decades. Elbaum and Rothermel et.al [13–16] discussed test case prioritization techniques of the fine-grained entity, such as coverage prioritization (statement or branch coverage, etc.) and fault-exposing-potential (FEP) prioritization. Meanwhile, total strategy and additional strategy are proposed [15]. Both are built on a Greedy algorithm which selects a local optimal solution within the search space at each round. Srikanth et al. [17–19] proposed a value-driven approach called PORT which does not require structural coverage information. The PORT algorithm was based on four factors: customer priority, requirements volatility, implementation complexity, and fault proneness of the requirements. Arafeen and Do [20] proposed a test case prioritization technique using requirement-based clustering. It incorporated traditional code analysis information which could improve the effectiveness of test case prioritization techniques.

2.2. Traditional TCP Techniques. Existing TCP methods—prioritizing test cases based on coverage [16] or requirement [19], or even time-aware [21, 22]—are all based on the optimization of software system itself. That is to say, traditional TCP methods focus on improvement of methods themselves. The factors they consider are based on characteristics of software and do not involve characteristics of hardware. Most of the software they test is also cross-platform web application.

There are several classical test case prioritization techniques, introduced as follows:

Random prioritization [16]. Random prioritization orders test cases randomly. It is simple and convenient, but unstable.

Total coverage prioritization [16]. It orders test cases based on the descendent number of units covered by these test cases. When multiple test cases cover the same number of units, the order is determined randomly.

Additional coverage prioritization [16]. It orders test cases to achieve maximized coverage as early as possible. It first picks the test case with the greatest coverage and then successively adds those test cases that cover the most yet uncovered parts.

Prioritization of Requirements for Test (PORT) [17–19]. It orders test cases based on the descending order of weighted priority (WP) values so that the test case with a higher WP value will be ordered in the front.

Optimal prioritization [16]. It prioritizes test cases using the faults, and it can obtain the ordering of test cases that maximizes a test suite's rate of fault detection. It provides an upper bound on the effectiveness of the other heuristics.

2.3. TCP Techniques Utilizing the Execution Information. When a test case has been executed, it generates execution information, such as its fault detection. As regression testing becomes more complex, scholars have considered the impact of execution information of test history to the current test prioritization.

2.3.1. History-Based TCP. A history-based TCP technique [23] sorts test cases according to the selection probabilities calculated from test history. It defines the selection probability of each test case as follows:

$$\begin{cases} P_0 = h_1, \\ P_k = \alpha h_k + (1-\alpha) P_{k-1}, \end{cases} \tag{2}$$

where P is selection probability, h is test history, and α is a smoothing constant used to weight individual histories.

Three test histories (based upon each test case's execution history, its fault detection, and the program entities it covers) have been investigated on the effect of test prioritization. Their experimental results show that historical information may be useful in reducing costs and increasing the effectiveness of long-running regression testing processes.

2.3.2. Adaptive TCP. As a main method of dynamic programming [24], the adaptive idea is also used in test case prioritization. Two types of adaptive TCP techniques are introduced here. They all take advantage of the impact of occurred faults to prioritize test cases in current test round.

(1) Adaptive TCP guided by output inspection. An adaptive test case prioritization guided by output inspection [25], which combines the test-case scheduling process and the test-case execution process, prioritizes test cases as the following process:

First, it calculates the initial fault-detection capability of all test cases based on the execution information of the previous output and then selects a test case t with the largest fault-detection capability. Second, t is executed on the modified program, and it records the output of t. Third, it modifies the fault-detection capability of remaining unselected test cases based on t's output and selects the test case with the largest modified fault-detection capability. Fourth, it repeats the preceding two steps until all the test cases have been prioritized and run.

(2) TCP based on adaptive sampling strategy. TCP techniques using cluster filtering [26, 27] select and prioritize test cases as the following process: first, it partitions the test suite based on cluster analysis; then, it selects test cases according to sampling strategy; finally, it prioritizes the selected test cases. In the sampling strategies, the *adaptive sampling* strategy is that it first initially selects one execution at random from each cluster and then all others of its cluster are selected if the first one selected from the cluster is a failure.

2.4. TCP Problem in a Smart Mall Scenario. Figure 1 is a simple distribution diagram of mobile devices for a smart mall scenario. In the figure, a wireless transmitter icon represents a smart mobile device mentioned and studied in this paper. The cloud icon represents central processing. A person with a mobile phone represents a handheld mobile device. Among them, white devices are distributed around specific locations (stores) to monitor and process specific-location information. Black devices are distributed in the middle of the mall to monitor certain types of information and perform distributed information processing. Each mobile device in the mall has its own unique function; that is, its internal intelligent software achieves specific requirements. Integrated testing of the software in devices throughout the mall becomes extremely complicated. For example, in the case like Figure 2, each restaurant has a mobile device that manages information about this restaurant. It can, via Internet, monitor the number of incoming customers/remaining seats, the number of dishes, the temperature, etc., and pushes location preferences, food preferences, etc. to guests (other mobile devices) entering the restaurant. In the hall of the mall, there are restaurant-proxy mobile devices that collect real-time restaurant and people data, via wireless network. It also intelligently pushes the best restaurants (vacant, near, etc.) to the mall customers (other mobile devices) at the current moment via Internet. This can schedule mall customers in real time, which may avoid occurring the case that all customers crowded in front of one restaurant. All data need to be transmitted to the control center for large-scale data processing via wireless network.

When discussing TCP problem in a smart mall scenario, traditional TCP methods can be improved by adding

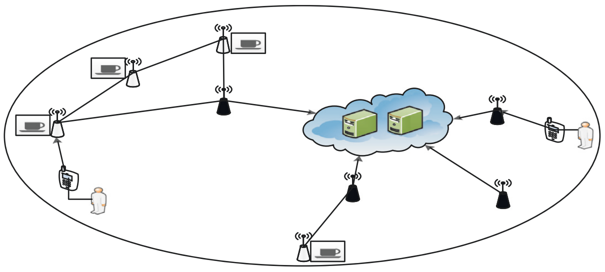

Figure 1: A simple mobile device distribution for a smart mall.

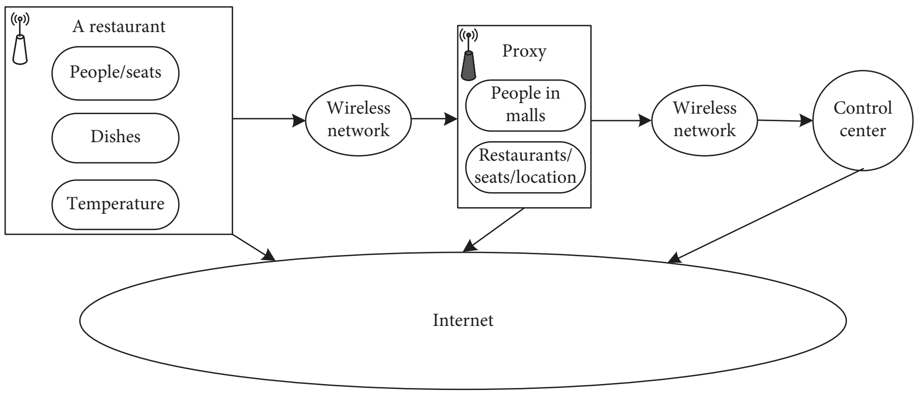

Figure 2: Simple software functions and communication structures for restaurant applications.

location information in sorting test cases to adapt the test order for the new scenario. Mobile devices are located in different locations, making them communicate more frequently (functionally interact more closely) with other close-range devices. According to distances between devices, the correlation between functions of intelligent software attached to devices is also strong or weak. As shown in Figure 1, software functions of the black device on the left side should have a greatest relationship with software functions of the other three white devices which communicate with this black device. Test cases test software functions of mobile devices. We set the granularity of a test case as testing all of the functional requirements of a mobile device. In this way, the node graph between traditional test cases becomes a node graph between actual mobile devices (Figure 3). In Figure 3, the left ellipse is a test-case node graph where a circle icon indicates a test case, and the right ellipse is a device node graph where a square icon indicates a mobile device. r represents the distance between test cases or devices. The virtual distance between test cases is mapped to the actual distance between devices.

3. Location-Based TCP Using the Law of Gravitation

This section combines test case information, fault information, and location information to propose a new location-based TCP technique using the law of gravitation.

3.1. Test Gravitation.

Test gravitation (TG) is introduced to simulate the universal gravitation in our method. *Test gravitation* F between two test cases t_a and t_b can be defined as follows:

$$F = G\frac{m(t_a)m(t_b)}{r^2}, \tag{3}$$

where G is the test gravitational constant, $m(t_a)$ and $m(t_b)$ are the masses of t_a and t_b, respectively, and r is the distance between t_a and t_b.

G is related to the environment of regression testing. If the criterion of m and r is certain, G should be unique. In this paper, we will not research its influence on the proposed method. So, G is set as 1.

Different attributes of a test case can represent different substances that make up this test case. If this test case detected faults, the attributes of a fault, which is as another type of substances, are also included to make up this test case. The weight of two types of attributes (substances) together makes up the total mass of this test case.

Definition 1. Test case mass m. The mass $m(t)$ of a test case t is defined as follows:

$$m(t) = \mu\sum_{i=1}^{n} w_i + (1-\mu)\sum_{j=1}^{e}\sum_{i=1}^{k}\overline{w_i}, \tag{4}$$

where n is the total number of attributes of t, w_i is the weight of ith attribute of t, e is the total number of faults which t detects, k is the total number of attributes of a fault, $\overline{w_i}$ is the weight of ith attribute of this fault, and μ is a smoothing constant, which is $0<\mu\leq 1$.

For instance, in the implementation process, w_1 can be represented as the coverage of a test case and w_2 can be

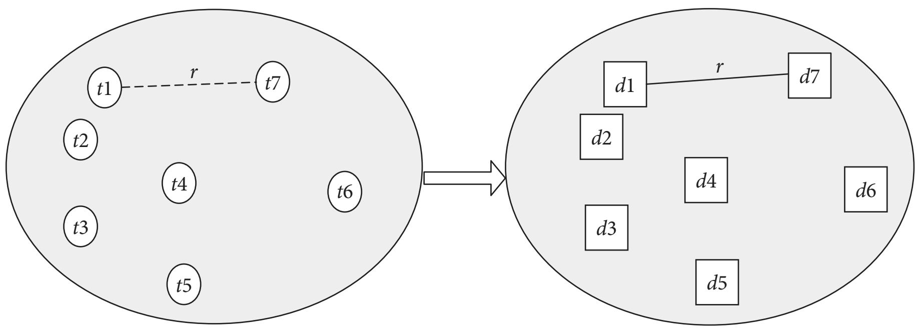

Figure 3: A test-case node graph maps to a device node graph.

represented as the importance level of a test case; $\overline{w_1}$ can be represented as the location level of a fault, $\overline{w_2}$ can be represented as the severity of a fault, and so on.

Because faults cannot be known in advance during the actual testing process, there are two ways to obtain faults and their attributes. One way is presetting faults, which can be given based on expert decision or deep learning; the other way is utilizing occurred faults.

Definition 2. Distance r. It indicates the distance between two test cases, denoted as $r(t_a, t_b)$.

For instance, r can be calculated according to the business level (tree relationship) between test cases or according to the spatial distance between devices they are located.

3.2. TG Calculation Model. In a smart mall scenario, according to the above definitions, we design a specific calculation model of *TG* to make preparations for prioritizing test cases. This model calculates a force F from a test case to a faulted test case.

(1) Importance level (*TI*) of a test case t is selected as the only attribute of t. *TI* is determined by the functional level of the mobile device (*DL*) which t tests. The mass of t is

$$m(t) = DL(d), \tag{5}$$

where d is the device tested by t. *DL* is divided into 5 levels. It can use a linear assignment, such as $n(n \in \{1, 2, 3, 4, 5\})$, or a nonlinear assignment, such as $a^n(a \in Z, n \in \{1, 2, 3, 4, 5\})$.

(2) Fault severity (*FS*) is selected as the only attribute of a fault f. The values of *FS* and *TI*(*DL*) compose $m(t_{(f)})$; that is

$$m(t_{(f)}) = \mu DL(d_{(f)}) + (1-\mu)\sum_{j=1}^{e} FS(f_j), \tag{6}$$

where $d_{(f)}$ is the device tested by the faulted test case $t_{(f)}$ and f_j is jth fault detected by $t_{(f)}$. *FS* is divided into 5 levels, like *DL*.

Occurred faults, detected by preselected test cases in current test round, are used to create a set of faulted test cases (*FTS*). The formation of a *FTS* will be described in detail in Section 3.3.

(3) Spatial location distance of devices is selected to calculate r. r is the 3-dimensional Euclidean distance between a device which a test case tests and a device which a faulted test case tests, as shown in Figure 4. It is defined as follows:

$$r(t_i, t_{(f)}) = ED(d_i, d_{(f)}), \tag{7}$$

where d_i is the device whose software is tested by t_i, and $d_{(f)}$ is the device tested by $t_{(f)}$.

(4) From the above, a specific calculation model of *TG* between a test case t_i and a faulted test case $t_{(f)}$ is as follows:

$$\begin{cases} F(t_i, t_{(f)}) = G\dfrac{m(t_i)m(t_{(f)})}{r(t_i, t_{(f)})^2}, \\ G = 1, \\ m(t_i) = DL(d_i), \\ m(t_{(f)}) = \mu DL(d_{(f)}) + (1-\mu)\displaystyle\sum_{j=1}^{e} FS(f_j), \\ 0 < \mu < 1, \\ r(t_i, t_{(f)}) = ED(d_i, d_{(f)}), \end{cases} \tag{8}$$

where d_i is the device whose software is tested by t_i, $d_{(f)}$ is the device tested by $t_{(f)}$ which detected e faults, and f_j is jth fault detected by $t_{(f)}$.

3.3. Faulted Test Case Set. Occurred faults which detected by some preselected test cases in the current test round are used as the faults mentioned in Section 3.2, so how to collect these occurred faults to create a *FTS* is an important step. The fault attaches to the device. We use clusters of devices to obtain a *FTS*. Algorithm 1 describes a clustering process of devices. Euclidean distance is used as the dissimilarity metric.

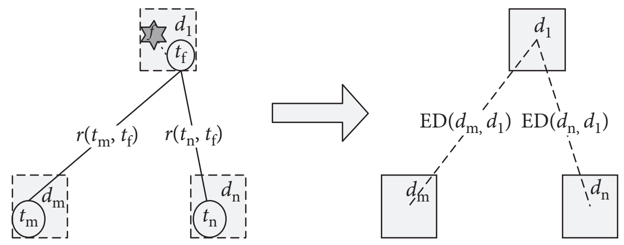

FIGURE 4: The distance between a test case and a faulted test case maps to the distance between devices.

After devices clustered, one device is selected randomly from each cluster as the representative of this cluster. Test cases that test these representatives are put into a test subset *ST*. *ST* is executed. If faults occurred, the test cases which detected these faults are combined into a *FTS*. Algorithm 2 shows the pseudocode of this process.

3.4. Location-Based TCP Using the Law of Gravitation Algorithm

Definition 3. Test case priority *P*. It indicates the priority of a test case in the execution order. The priority *P* is defined as follows:

$$P = \sum_{i=1}^{h} F_i, \tag{9}$$

where h is the number of faulted test cases and F_i is the force F of this test case to the ith faulted test case. The larger the P value is, the earlier this test case will execute.

Algorithm 3 shows the location-based TCP using the law of gravitation approach. Its input is a test suite T. Its output is the prioritized test order T'. First, m of each test case t is calculated according to the level of a mobile device which t tests. Second, a faulted test case set *FTS* is created according to algorithms 1 and 2. m of each faulted test case $t_{(f)}$ is calculated based on both *FS* and *DL*. Third, the distance r between each t and each $t_{(f)}$ can be calculated according to location information of devices. Fourth, for each t, the force F is computed from this t to each $t_{(f)}$. Fifth, the priority P of each t is calculated according to F. Finally, test cases are sorted in descending order of P to obtain a prioritized test execution order T'.

3.5. Example for Simulating Smart Mall. We simulate a smart mall scenario with Figure 5 to explain how to prioritize test cases. There are five mobile devices (d_1–d_5) in the figure, shown by squares. Each device is tested by a test case for its internal intelligent software functions. So, there are five test cases (t_1-t_5), shown by circles. Assume that the devices are clustered into 2 clusters: $c_1(d_1, d_2, d_3)$ and $c_2(d_4, d_5)$. d_2 and d_4 are extracted randomly to be device representatives, and a subset ST $\{t_2, t_4\}$ is formed. After ST run, two faults (f_1 and f_2) are found by t_2, which are shown by stars in the figure. A dashed line in the figure shows the 3-dimensional Euclidean distance between two devices.

In the above example, let us consider a test case prioritization problem defined over a set of five test cases $T(t_1, t_2, t_3, t_4, t_5)$ with a set of one faulted test case $FTS(t_{(f)1})$ from Table 1. From Figure 5, according to the location of devices, the distances between test cases are obtained, as in Table 1. We suppose that all faults (including their Severity levels) detected in current testing round are shown in Table 2.

We take t_1 as a sample and calculate the force F of t_1 to $t_{(f)1}$, as $F_1 = 0.135$. According to Equation (9), we get the priority value of t_1 which is $P_1 = 0.135$. Similarly, the priority P of t_2, t_3, t_4, and t_5 are $P_2 = 22.5$, $P_3 = 0.28125$, $P_4 = 0.0002$, and $P_5 = 0.000225$. The prioritization order is t_2-t_3-t_1-t_5-t_4, and the *APFDc* [28] value of this order is 78.57%. According to Table 2, the optimal prioritization sorts test cases as the order t_2-t_3-t_1-t_5-t_4 (or t_2-t_3-t_1-t_4-t_5), whose *APFDc* value is 78.57%. The random prioritization sorts test cases as one order t_5-t_1-t_3-t_2- t_4, whose *APFDc* value is 41.43%. It can be seen that the effect of our location-based TCP using the law of gravitation has a good effect, which is even consistent with the optimal prioritization.

4. Empirical Evaluation

To investigate the effectiveness of the method, called location-based TCP, using the law of gravitation (L-TCP from now on), an empirical evaluation is performed in terms of the following research questions:

(i) RQ1: Is L-TCP approach more effective in the rate of fault detection than other traditional prioritization techniques?

This research question aims at understanding whether the L-TCP method can detect faults earlier than other traditional test case prioritization techniques. To answer this question, this paper applies four traditional TCP techniques for comparison.

(ii) RQ2: When evaluating the efficiency of techniques, is there any difference in the experimental conclusions for whether or not considering faults severities?

Whether or not to consider severity of a fault will undoubtedly make a difference in the judgment of the prioritization effect. This research question mainly discusses the influence of two evaluation metrics on the experimental conclusions.

(iii) RQ3: In addition to location information, what other factors the prioritization efficiency is also related to?

In the smart mall scenario, test prioritization efficiency of software may be related to the information of mobile devices. This research question combines analyses of the above two questions to discuss factors that influence the efficiency of prioritization.

4.1. Object. The object used in this experimental study is a real industrial project which is for chip testing and has

```
      Input: $D = \{d_1, d_2, \cdots d_n\}$, $k$ //a device set, and the number of clusters
      Output: $C$ //a set of $k$ clusters
(1)  $C = \varnothing$;
(2)  put each $d \in D$ as a cluster $c$;
(3)  add all clusters into $C$; //Initialization: get a single-cluster set $C$
(4)  Do //Iteration: make clusters merge.
(5)    For each $c_i,\ c_j \in C$
(6)      If ($c_i$ and $c_j$ have the minimum 3-dimensional Euclidean distance)
(7)        merge $c_i$ and $c_j$ into a new cluster $c_{new}$;
(8)        delete $c_i$ and $c_j$ from C;
(9)        $C = C \cup \{c_{new}\}$;
(10)     End if
(11)   End for
(12) Until The number of clusters in C is $k$ //Break condition
```

Algorithm 1: Clustering.

```
      Input:
        $C = \{c_1, c_2, \cdots c_m\}$ //a set of device clusters
        $T = \{t_1, t_2, \cdots t_m\}$ //a set of test cases
      Output:
        $FTS$ //a set of faulted test cases
(1)  $FTS = \varnothing$;
(2)  While ($FTS == \varnothing$) do
(3)    $ST = \varnothing$;
(4)    For each $c \in C$
(5)      Randomly select one $d$ (i.e., $d_s$) from $c$;
(6)      Select the test case $t \in T$ (i.e., $t_s$) which tests the software of $d_s$;
(7)      $ST = ST \cup \{t_s\}$;
(8)    End for
(9)    For each $t \in ST$
(10)     Execute $t$;
(11)     if ($t$ detects faults) then
(12)       put $t$ into $FTS$;
(13)     End if
(14)   End for
(15) End while
(16) Return $FTS$
```

Algorithm 2: Creating a *FaS*.

approximately 140,000 lines of codes (LOC), totally. It has many versions, and each version has a few of requirements. The test suite of each version is relatively small. The granularity of test cases is coarse-grained. That is to say, each test case may contain dozens or even hundreds of test scripts, but it tests only one chip function (requirement). These features can be used to simulate test data for a smart mall scenario. First of all, functions of the hardware chip are similar to those of smart devices, so characteristics of the faults may be similar, too. Second, each test case covers only one specific requirement, which can simulate to test one mobile device. Third, there are many rounds (versions) of regression testing, and there are new test cases introduced in each round, which can simulate a step-by-step integration testing environment for the smart mall scenario. The project data include the number of test cases, the functional requirements covered by test cases, the faults detected by test cases, the fault severities, and so on. We use this project data as a basis and then simulate the distance data between devices. Finally, they are formed into a complete data required for this experiment. There are six versions chosen for this experiment. The basic information is shown in Table 3.

4.2. Variables and Measures

4.2.1. Independent Variables. To address our research questions, one independent variable is manipulated: test case prioritization technique. Besides our proposed L-TCP approach, the following traditional test case prioritization approaches are also implemented for comparison.

(i) Random (R): this technique uses random prioritization technique to order test cases without using location information of devices in prioritization.

```
     Input: Test suite $T$
     Output: Prioritized test suite $T'$
     General process:
     Begin
(1)  Calculate $m$ of each $t$ in $T$
(2)  Cluster devices according to Algorithm 1
(3)  Create a $FTS$ according to Algorithm 2
(4)  Calculate $m$ of each $t_{(f)j}$ in $FTS$
(5)  for each $t_i$ in $T$
(6)    for each $t_{(f)j}$ in $FTS$
(7)      Calculate $r$ between $t_i$ and $t_{(f)j}$
(8)      Calculate $F_j$ of $t_i$ to $t_{(f)j}$
(9)    end for
(10)   Calculate $P$ of $t_i$
(11) end for
(12) Sort all $t$ in $T$ based on the descending order of $P$ and obtain the new test execution order, being $T'$
(13) return $T'$
     End
```

Algorithm 3: Location-based TCP using the law of gravitation.

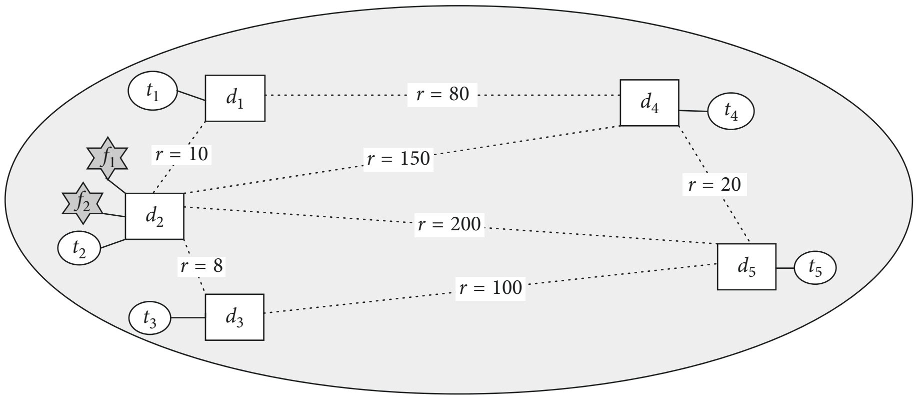

Figure 5: A testing example simulating a smart mall.

Table 1: Information of T and FTS.

T	m	FTS	m	$r(t, t_{(f)})$	$t_{(f)1}$
t_1	3	$t_{(f)1}$	4.5	t_1	10
t_2	5			t_2	1
t_3	4			t_3	8
t_4	1			t_4	150
t_5	2			t_5	200

Table 2: All faults detected by test cases.

	f_1	f_2	f_3	f_4		FS
t_1			√		f_1	3
t_2	√	√			f_2	1
t_3				√	f_3	1
t_4					f_4	2
t_5						

(ii) Total coverage (TC): this technique uses total coverage prioritization technique to order test cases without using location information of devices in prioritization.

Table 3: Basic information of ChipTest.

Version	v-9	v-10	v-11	v-12	v-13	v-14
New requirements	5	9	5	8	7	16
New test cases	9	9	5	8	7	16
Total test cases	9	18	23	31	38	54
Faults	7	8	5	8	7	8

(iii) Additional coverage (AC): this technique uses additional coverage prioritization technique to order test cases without using location information of devices in prioritization.

(iv) Requirement prioritization (PORT): this technique uses prioritization of requirements for test technique to order test cases without using location information of devices in prioritization.

4.2.2. Dependent Variable and Metric. Details on the measures for the dependent variables of these experiments are given here.

APFD. To measure how rapidly a prioritized test suite detects faults, average percentage of fault detection (*APFD*) is used as the dependent variable.

APFD [28], the weighted average of the percentage of faults detected, focuses on the rate of fault detection during the testing life of a test suite. It assumes that the faults severities are equivalent. The equation of *APFD* is as follows:

$$APFD = 1 - \frac{TF_1 + TF_2 + \cdots + TF_m}{nm} + \frac{1}{2n}, \tag{10}$$

where n is the number of test cases, m is the number of faults, and TF_i is the index of the first test case that reveals the ith fault in the execution order T. The value of *APFD* varies from 0 to 100%. Since n and m are fixed for any orders, a higher *APFD* value indicates that the faults are detected earlier during the testing process.

APFDc. When considering faults severities, we use *APFDc* [28], the (cost-cognizant) weighted average percentage of faults detected, to reward test case orders proportionally to their rate of units of fault severity detected. We assume that test case costs are identical. The equation of *APFDc* is simplified as follows:

$$APFDc = 1 - \frac{\sum_{i=1}^{m} (fs_i \times TF_i)}{n \times \sum_{i=1}^{m} fs_i} + \frac{1}{2n}, \tag{11}$$

where fs_i is the severity of the ith fault and other symbols have the same definition as in the equation of *APFD*.

4.3. Case Study Design.

Suppose that there are many smart mobile devices in a mall and each device is responsible for its own unique functions. We now need to test the functionality of their internal intelligence software. We assume that one test case is in charge of testing one device, and it tests all of the software functionality of this device.

First, it collects data. To conduct the comparative experiments, five types of data information are required, including test case, levels of test cases (devices), fault, severities of faults, distance of devices, and coverage information.

The preparation of test case, fault, severities of faults, and coverage information is trivial because it is already available in the original data of the object system. For the preparation of the levels of test cases (devices), we grade test cases according to their name description. The preparation of the distance between mobile devices requires us to give them values by simulating the smart mall scenario.

Second, it performs test case prioritization techniques. This experimental study implements five approaches (TC, AC, R, PORT, and L-TCP) for comparison. Because of the indeterminacy of some prioritization techniques, each technique runs 20 times for each experiment and the average values are presented as results. The smoothing constant is set $\mu = 50\%$.

Third, it calculates *APFD* and *APFDc* for each prioritized test order from each technique. All measure values are compared across different techniques. The results emerging from this comparison are presented in the Section 4.4.

All the experiments are conducted on the same computer which is configured as 64-bit windows 8 operating system, Intel(R) Core(TM) i3-2130 CPU and 4 GB memory.

4.4. Results and Analysis.

In this section, we present the results of the experiment(s) and analyze their relevance to our research questions above.

4.4.1. RQ1: Comparison with Traditional TCP Techniques.

Figure 6 shows the box plots of five techniques across all the system versions of ChipTest. The horizontal axis shows versions, and each box in a version presents one TCP technique. The vertical axis presents *APFD* values. Each boxplot shows the median, upper/lower quartile, and max/min *APFD* values achieved by a technique.

From the boxplot, L-TCP, as indicated by *APFD* scores, significantly outperforms the others because its median point reaches up to the highest. Besides L-TCP, PORT performs better with a higher median point. TC, AC, and R have a similar effect, and their median points of *APFD* locate approximately between 40% and 65%.

For instance, let us choose the data of v-9 for analysis. We use *M*(*Median*, *Q1*, *Q3*) to denote the median, first, and third quartiles *APFD* values for each technique and *M1-M5* to denote the five techniques: TC, AC, R, PORT, and L-TCP, respectively. So, results of the five techniques are *M1(38.89, 29.37, 48.02)*, *M2(34.13, 29.37, 43.25)*, *M3(38.1, 31.35, 46.43)*, *M4(45.24, 40.88, 46.83)*, and *M5(78.57, 78.57, 78.57)*, respectively, which clearly indicates that L-TCP overall performs better than the others.

For evaluating the confidence level of the observed results, we test their statistical significance. First, a single sample K-S test is used to check the normal distribution of the data of each technique from 120 executions (20 running × 6 versions). The significance level is $\alpha = 0.05$. Their results are as follows: in Table 4, the first row is the names of TCP approaches mentioned above. The second row shows the judge of normal distribution for TC, AC, R, PORT, and L-TCP. The third row shows their significance probability values under the null hypothesis.

From Table 4, their results accept the null hypothesis (p values are all greater than 0.05). So, *APFD* values of the five prioritization techniques satisfy a normal distribution.

Next, the paired-samples t-test is employed to obtain sufficient statistical evidence. f_1 and f_2 are defined as the values of *APFD*, which are prioritized by two prioritization approaches, respectively.

The following two hypotheses are considered:

H_0: $f_1 = f_2$, if two techniques have the same effectiveness in the rate of fault detection.

H_1: $f_1 > f_2$, if f_1 is significantly better than f_2.

If the p value is less than the significance level ($\alpha = 0.05$), we can reject the null hypothesis and accept the alternative hypothesis.

Table 5 reports the results of statistical testing by using the data from 120 executions. Their results show that L-TCP

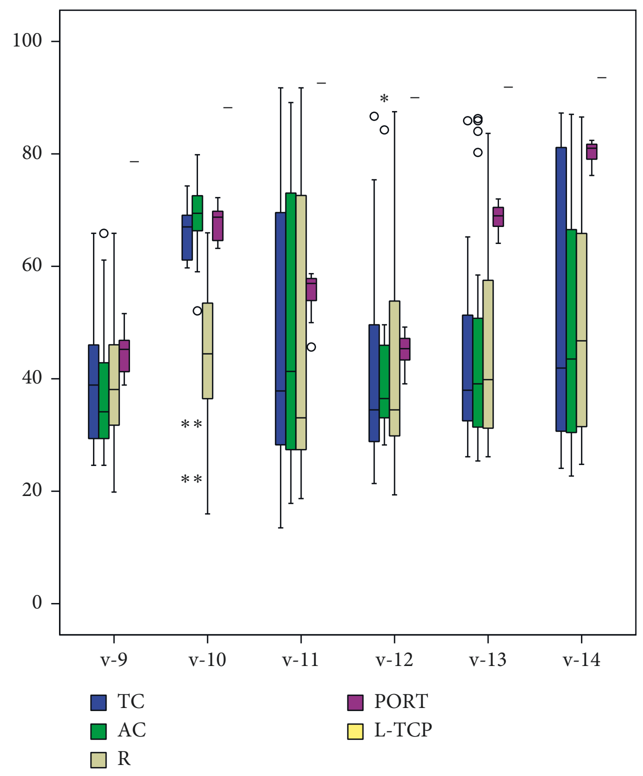

Figure 6: *APFD* distributions of different techniques in different versions. *Discrete value; "open circle" indicates an extreme value; –median value.

Table 4: Sample K-S test results of *APFD* values of the five TCP techniques.

	TC	AC	R	PORT	L-TCP
Normal distribution?	Y	Y	Y	Y	Y
Sig.	0.976	0.709	1	0.977	0.773

Table 5: Statistical test results from comparing *APFD* values of L-TCP to four opponent techniques.

	L-TCP	
	p value	t
TC	0.000	22.947
AC	0.000	21.728
R	0.000	25.486
PORT	0.000	28.295

is statistically significantly better than other TCP techniques because its t values are greater than 0 and p values are less than 0.05.

For instance, compared with TC, the p value of L-TCP equals 0.000 and its t value equals 22.947, so we can reject the null hypothesis that L-TCP and TC have the same effectiveness in the rate of fault detection and accept the alternative hypothesis that L-TCP is significantly better than TC.

4.4.2. RQ2: Effects of Different Evaluation Metrics. In the evaluation metrics of fault-detection rate, *APFD* is one that does not consider faults severities and *APFDc* is one that considers faults severities.

Figure 7 shows *APFD* and *APFDc* distributions of different techniques in different versions. As can be seen from Figure 7, L-TCP has highest values in both *APFD* and *APFDc* evaluations. That is to say, the prioritization effect of L-TCP is the best among other techniques, regardless of whether or not faults severities are considered in evaluation. In addition, the trend of other techniques is similar in both evaluations (except in version 9 and version 11); that is, PORT is a second best technique besides L-TCP. In version 9, *APFD* evaluation shows that effects of TC, AC, R, and PORT are similar, as shown in Figure 7(a). However, in *APFDc* evaluation, PORT is significantly better than the other three techniques in version 9, as shown in Figure 7(b). In version 11, *APFD* evaluation shows that PORT is significantly better than the others, but in the evaluation of *APFDc*, AC is slightly better than PORT.

Therefore, from the results, whether or not considering faults severities will not affect the conclusion of RQ 1, that is, L-TCP is superior to other techniques. It is just that the degree of excellence varies across different metrics.

4.4.3. RQ3: Factors Affecting Prioritization Efficiency. From the analysis of the results of RQ1 and RQ2, it can be seen that L-TCP is the best technique to improve the rate of faults detection. In addition to L-TCP, PORT is the second best performing technique.

In-depth analysis shows that, first of all, according to the characteristics of test data, L-TCP mainly affects the prioritization efficiency by location information of devices. That is, in the smart mall scenario, location information is the main factor affecting the test order efficiency of intelligent software embedded in mobile devices. Second, in the smart mall scenario, PORT sorts test cases according to the priority of software functions of mobile devices in this experiment. The functional priority of a device determines the level of a device, and the device level determines the mass of a test case which tests this device. In retrospect, in the smart mall scenario, the *test gravitation* calculation model considers both *device location information* and *device level*, which are the main factors that influence the test prioritization efficiency. So, this may be the reason why L-TCP can achieve better sorting results in this smart mall scenario.

4.5. Threats to Validity. In terms of the internal validity, the choice of the smoothing constant μ can affect the results. In this paper, the selection of this parameter has been based on equalization, that is, $\mu = 50\%$. Further investigations can study the effect of the smoothing constant.

The threats to external validity are from the object, its test data, and its faults used by this experimental study. To reduce this threat in the object, the experimental object we select is the system that tests chips, which is an object that is relatively close to the simulated scenario. Moreover, we select multiple successive versions (6 versions) for

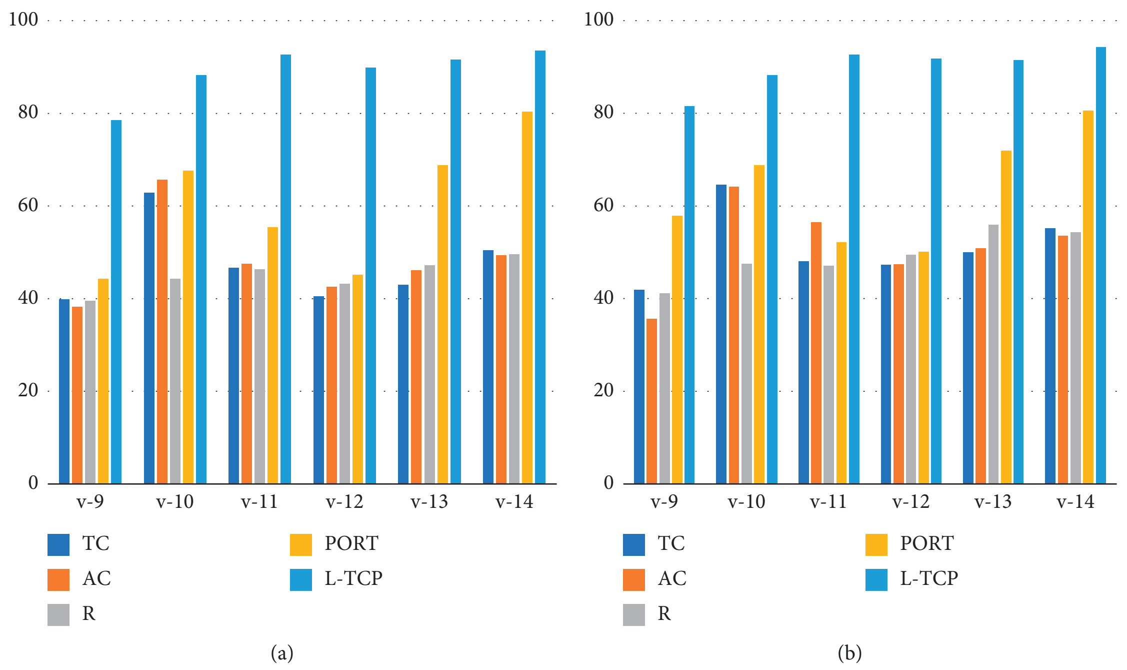

Figure 7: *APFD* and *APFDc* distributions of different techniques in different versions: the histograms of (a) *APFD* and (b) *APFDc*.

experiments to simulate step-by-step integration testing of a smart mall scenario. The second external threat lies in the test data in this object. Although the data are relatively real, it is not complete enough for the research in this paper. For incomplete data (such as the lack of distances between devices), we try to simulate the data supplement according to the scenario. The third external threat is the faults. For faults, we use actual real faults in order to be closer to the real scenario.

The threat to construct validity lies in whether the experimental results are measured in a correct way. To reduce this threat, firstly, *APFD* is used to measure the effectiveness of a prioritized test case order since *APFD* can measure the rate of fault detection and has been widely used in the evaluation of the test case prioritization problem. Second, *APFDc* is also used to measure accurately the rate of units of fault severity detected since it considers faults severities.

5. Related Work

Test case prioritization has been an interesting research field for nearly two decades. Rothermel et al. [13] firstly proposed the complete definition of TCP problem which is finding a permutation of T in order to maximize some objective functions. They focus on code-coverage TCP methods at code-level [13–16]. In 2001, Elbaum conducted specific research for TCP metrics, including APFD and APFDc [28]. APFD metric proclaims that all faults have the same severity and all test cases have equal costs. APFDc, units of fault severity detected per unit test cost, considers unifying test case costs and fault severities. Their study was primarily focused on white-box testing but not on black-box testing. Zhang et al. [29] considered requirement priorities to TCP and proposed an algorithm called TCP_RP_TC. The prioritization technique must predict requirement priorities and test costs before test suite execution, but the prediction was difficult in practice. Chu-Ti et al. [30] presented a history-based TCP method with software version awareness. Yuchi et al. [31] designed and analyzed TCP using weight-based methods for GUI applications. Garg and Datta [32, 33] used test case prioritization in web applications based on modified functionalities or database changes. Saha et al. [34] proposed a fully automated and lightweight test prioritization approach (REPiR) to address the problem of regression test prioritization by reducing it to a standard information retrieval problem so that the differences between two program versions formed the query and the tests constituted the document collection. Some researchers [35] focused on test case prioritization based on mutation analysis. It is an effective method, but the cost is expensive. Another novel refactoring-based approach (RBA) was proposed by Alves et al. [36] which reordered an existing test sequence utilizing a set of refactoring fault models. It promoted early detection of refactoring faults. Wang and Ali et al. [37] proposed a resource-aware multiobjective optimization solution with a fitness function defined based on four cost-effectiveness measures. Prioritizing test cases for the testing of location-aware services was proposed by Zhai et al. [38, 39], and it brings in service selection into a test case prioritization technique for testing the location-based web services.

Mobile application testing is a research direction for testing on mobile devices. However, most of mobile application testing focuses on performance testing or stand-alone testing which sees the software of mobile devices as a stand-alone software. Gao et al. [40] provided a general tutorial on mobile application testing that first examined testing requirements and then looked at current approaches for both native and Web apps for mobile devices. Muccini, Di Francesco, and Esposito [41] investigated new research directions on mobile applications testing automation, by answering three research questions. Given the first research question (RQ1) *are mobile applications (so) different from traditional ones, so to require different and specialized new testing techniques?*, the natural answer seems to be *yes, they are*. About (RQ2) *what are the new challenges and research directions on testing mobile applications?*, the challenges seem to be many, related to the contextual and mobility nature of mobile applications. As far as concern (RQ3) *which is the role automation may play in testing mobile applications?*, some potentials for automation have been outlined, being aware that a much deeper and mature study shall be conducted. Dantas et al. [42] proposed a set of testing requirements, elicited using the results of an extensive research on how the testing process for mobile applications is done in the literature and in practice. Morla and Davies [43] created a test environment that supports the evaluation of key aspects of location-based applications without the extensive resource investment necessary for a full application implementation and deployment. Zhang and Adipat [44] proposed a generic framework for conducting usability tests for mobile applications through discussing research questions, methodologies, and usability attributes. Vilkomir [45] evaluated the effectiveness of coverage approaches for selecting mobile devices (i.e., smartphones and tablets) to test mobile software applications. Amalfitano et al [46] addressed the problem of testing a mobile app as an event-driven system by taking into accounts both context events and GUI events. Kim, Choi, and Wong [47] proposed a method to support performance testing utilizing a database established through benchmark testing in emulator-based test environment at the unit test level.

6. Conclusion

This paper proposes a location-based TCP using the law of gravitation approach. It introduces *test gravitation*, which combines three factors (test case information, fault information, and location information), to prioritize test cases. Test case information involves the level of mobile device. Fault information includes the severity of fault. In addition, we use occurred faults to create a faulted test case set. It is obtained in three steps: devices clustering, test subset extraction, and running preselected test cases. Location information involves the actual location of devices. It is used to calculated the 3-dimensional Euclidean distance between two devices. Finally, it experimentally verifies the effectiveness of L-TCP technique in comparison with several traditional test case prioritization techniques.

The experimental results show that the median APFD value of L-TCP is 78.57%, which is higher than the values of the baseline methods. When employing the paired-samples t-test, L-TCP's t values are greater than 0 and p values are less than 0.05. Specially, (1) comparing with TC, the p value of L-TCP equals 0.000 and its t value equals 22.947; (2) comparing with AC, the p value of L-TCP equals 0.000 and its t value equals 21.728; (3) comparing with R, the p value of L-TCP equals 0.000 and its t value equals 25.486; and (4) comparing with PORT, the p value of L-TCP equals 0.000 and its t value equals 28.295. These results indicate that L-TCP is statistically significantly better than other TCP techniques and it can detect more faults than others at the same time consumption.

When considering the factor of faults severities during the evaluation, the conclusion that L-TCP is superior to other techniques will not be affected. It is just that its degree of excellence varies across different metrics. In the smart mall scenario, location information of devices is the main factor which influences the prioritization performance. Furthermore, the level of devices is also important.

The next step is to expand the scope of empirical evaluation and try to make the conclusion more accurate. Moreover, how to give an appropriate parameter is also a research direction.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant nos. 61572306 and 61502294), the IIOT Innovation and Development Special Foundation of Shanghai (grant no. 2017-GYHLW-01037), and the CERNET Innovation Project (grant nos. NGII2017051 and NGII20170206).

Supplementary Materials

The ChipTest Data.xlsx file is the data of the project used in the experimental study of my paper. The results.xlsx file is the detailed results of my experimental study. The result analysis.xlsx file includes the original graphs of the results analysis, which are also shown in my paper. (*Supplementary Materials*)

References

[1] C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of IoT and cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 964–975, 2018.

[2] G. Han, L. Zhou, H. Wang, W. Zhang, and S. Chan, "A source location protection protocol based on dynamic routing in WSNs for the Social Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 689–697, 2018.

[3] L. Atzori, A. Iera, and G. Morabito, "The internet of things: a survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[4] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu, "A new user similarity model to improve the accuracy of collaborative

filtering," *Knowledge-Based Systems*, vol. 56, pp. 156–166, 2014.

[5] P. M. Adams, G. W. B. Ashwell, and R. Baxter, "Location-based services—an overview of the standards," *BT Technology Journal*, vol. 21, no. 1, pp. 34–43, 2003.

[6] G. Rothermel and M. J. Harrold, "A safe, efficient regression test selection technique," *ACM Transactions on Software Engineering and Methodology*, vol. 6, no. 2, pp. 173–210, 1997.

[7] X. Chen, J. H. Chen, X. L. Ju, and Q. Gu, "Survey of test case prioritization techniques for regression testing," *Ruan Jian Xue Bao/Journal of Software*, vol. 24, no. 8, pp. 1695–1712, 2013.

[8] I. Newton, *Philosophiae Naturalis Principia Mathematica*, Royal Society, London, UK, 1st edition, 1687.

[9] L. Peng, B. Yang, Y. Chen, and A. Abraham, "Data gravitation based classification," *Information Sciences*, vol. 179, no. 6, pp. 809–819, 2009.

[10] C. Wang and Y. Q. Chen, "Improving nearest neighbor classification with simulated gravitational collapse," in *Proceedings of International Conference on Natural Computation*, pp. 845–854, Springer, Changsha, China, August 2005.

[11] M. Indulska and M. E. Orlowska, "Gravity based spatial clustering," in *Proceedings of 10th ACM international symposium on Advances in geographic information systems*, pp. 125–130, ACM, New York, NY, USA, November 2002.

[12] G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," *IEEE Transactions on Software Engineering*, vol. 22, no. 8, pp. 529–551, 1996.

[13] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Prioritizing test cases for regression testing," in *Proceedings of International Symposium on Software Testing and Analysis*, pp. 102–112, ACM, Portland, OR, USA, August 2000.

[14] G. Rothermel, R. H. Untch, C. Chengyun Chu, and M. J. Harrold, "Prioritizing test cases for regression testing," *IEEE Transactions on Software Engineering*, vol. 27, no. 10, pp. 929–948, 2001.

[15] G. Rothermel, R. H. Untch, C. Chu et al., "Test case prioritization: an empirical study," in *Proceedings of IEEE International Conference on Software Maintenance (ICSM'99)*, pp. 179–188, IEEE, Oxford, England, UK, August-September 1999.

[16] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: a family of empirical studies," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 159–182, 2002.

[17] H. Srikanth and L. Williams, "Requirements-based test case prioritization," *IEEE Transactions on Software Engineering*, vol. 28, 2002.

[18] H. Srikanth, L. Williams, and J. Osborne, "System test case prioritization of new and regression test cases," in *Proceedings of 2005 International Symposium on Empirical Software Engineering*, p. 10, Noosa Heads, Queensland, Australia, November 2005.

[19] H. Srikanth and S. Banerjee, "Improving test efficiency through system test prioritization," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1176–1187, 2012.

[20] M. J. Arafeen and H. Do, "Test case prioritization using requirements-based clustering," in *Proceedings of 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation (ICST)*, pp. 312–321, IEEE, Luxembourg, March 2013.

[21] K. R. Walcott, M. L. Soffa, G. M. Kapfhammer et al., "Timeaware test suite prioritization," in *Proceedings of 2006 International Symposium on Software Testing and Analysis*, pp. 1–12, ACM, Shanghai, China, May 2006.

[22] S. Alspaugh, K. R. Walcott, M. Belanich et al., "Efficient time-aware prioritization with knapsack solvers," in *Proceedings of the 1st ACM International Workshop on Empirical Assessment of software Engineering Languages and Technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*, pp. 13–18, ACM, New York, NY, USA, 2007.

[23] J. M. Kim and A. Porter, "A history-based test prioritization technique for regression testing in resource constrained environments," in *Proceedings of 24rd International Conference on Software Engineering ICSE 2002*, pp. 119–129, IEEE, Orlando, FL, USA, May 2002.

[24] X. Luo, Y. Lv, R. Li, and Y. Chen, "Web service QoS prediction based on adaptive dynamic programming using fuzzy neural networks for cloud services," *IEEE Access*, vol. 3, pp. 2260–2269, 2015.

[25] D. Hao, X. Zhao, and L. Zhang, "Adaptive test-case prioritization guided by output inspection," in *Proceedings of 2013 IEEE 37th Annual Computer Software and Applications Conference (COMPSAC)*, pp. 169–179, IEEE, Kyoto, Japan, July 2013.

[26] W. Dickinson, D. Leon, and A. Podgurski, "Finding failures by cluster analysis of execution profiles," in *Proceedings of 23rd International Conference on Software Engineering*, pp. 339–348, IEEE Computer Society, Toronto, ON, Canada, May 2001.

[27] D. Leon and A. Podgurski, "A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases," in *Proceedings of International Symposium on Software Reliability Engineering*, pp. 442–453, Denver, CO, USA, November 2003.

[28] S. Elbaum, A. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization," in *Proceedings of 23rd International Conference on Software Engineering ICSE 2001*, pp. 329–338, IEEE, Toronto, Canada, May 2001.

[29] X. Zhang, C. Nie, B. Xu et al., "Test case prioritization based on varying testing requirement priorities and test case costs," in *Proceedings of Seventh International Conference on Quality Software QSIC'07*, pp. 15–24, IEEE, Portland, Oregon, USA, October 2007.

[30] C. T. Lin, C. D. Chen, C. S. Tsai et al., "History-based test case prioritization with software version awareness," in *Proceedings of 2013 18th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pp. 171-172, IEEE, Singapore, July 2013.

[31] C.-Y. Huang, J.-R. Chang, and Y.-H. Chang, "Design and analysis of GUI test-case prioritization using weight-based methods," *Journal of Systems and Software*, vol. 83, no. 4, pp. 646–659, 2010.

[32] D. Garg, A. Datta, and T. French, "A two-level prioritization approach for regression testing of web applications," in *Proceedings of 19th Asia-Pacific Software Engineering Conference (APSEC)*, vol. 2, pp. 150–153, IEEE, Hong Kong, China, December 2012.

[33] D. Garg and A. Datta, "Test case prioritization due to database changes in web applications, Software Testing," in *Proceedings of 2012 IEEE Fifth International Conference on Verification and Validation (ICST)*, pp. 726–730, IEEE, Montreal, QC, Canada, April 2012.

[34] R. K. Saha, L. Zhang, S. Khurshid et al., "An information retrieval approach for regression test prioritization based on

program changes," in *Proceedings of 37rd International Conference on Software Engineering ICSE 2015*, pp. 268–279, IEEE, Firenze, Italy, May 2015.

[35] R. Just and F. Schweiggert, "Higher accuracy and lower run time: efficient mutation analysis using non-redundant mutation operators," *Software Testing, Verification and Reliability*, vol. 25, no. 5–7, pp. 490–507, 2014.

[36] E. L. G. Alves, P. D. L. Machado, T. Massoni, and M. Kim, "Prioritizing test cases for early detection of refactoring faults," *Software Testing, Verification and Reliability*, vol. 26, no. 5, pp. 402–426, 2016.

[37] S. Wang, S. Ali, T. Yue et al., "Enhancing test case prioritization in an industrial setting with resource awareness and multi-objective search," in *Proceedings of 38th International Conference on Software Engineering Companion*, pp. 182–191, ACM, Austin, Texas, USA, May 2016.

[38] K. Zhai, B. Jiang, and W. K. Chan, "Prioritizing test cases for regression testing of location-based services: metrics, techniques, and case study," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 54–67, 2014.

[39] K. Zhai, B. Jiang, W. K. Chan et al., "Taking advantage of service selection: a study on the testing of location-based web services through test case prioritization," in *Proceedings of 2010 IEEE International Conference on Web Services (ICWS)*, pp. 211–218, IEEE, Miami, FL, USA, July 2010.

[40] J. Gao, X. Bai, W.-T. Tsai, and T. Uehara, "Mobile application testing: a tutorial," *Computer*, vol. 47, no. 2, pp. 46–55, 2014.

[41] H. Muccini, A. Di Francesco, and P. Esposito, "Software testing of mobile applications: challenges and future research directions," in *Proceedings of 7th International Workshop on Automation of Software Test*, pp. 29–35, IEEE Press, Zurich, Switzerland, June 2012.

[42] V. L. L. Dantas, F. G. Marinho, A. L. da Costa et al., "Testing requirements for mobile applications," in *Proceedings of 24th International Symposium on Computer and Information Sciences ISCIS 2009*, pp. 555–560, IEEE, Guzelyurt, Northern Cyprus, Turkey, September 2009.

[43] R. Morla and N. Davies, "Evaluating a location-based application: a hybrid test and simulation environment," *IEEE Pervasive computing*, vol. 3, no. 3, pp. 48–56, 2004.

[44] D. Zhang and B. Adipat, "Challenges, methodologies, and issues in the usability testing of mobile applications," *International Journal of Human-Computer Interaction*, vol. 18, no. 3, pp. 293–308, 2005.

[45] S. Vilkomir, "Multi-device coverage testing of mobile applications," *Software Quality Journal*, vol. 26, no. 2, pp. 197–215, 2017.

[46] D. Amalfitano, A. R. Fasolino, P. Tramontana et al., "Considering context events in event-based testing of mobile applications," in *Proceedings of 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 126–133, IEEE, Luxembourg, March 2013.

[47] H. Kim, B. Choi, and W. E. Wong, "Performance testing of mobile applications at the unit test level," in *Proceedings of Third IEEE International Conference on Secure Software Integration and Reliability Improvement SSIRI 2009*, pp. 171–180, IEEE, Shanghai, China, July 2009.