Rescue Wings: Mobile Computing and Active Services Support for Disaster Rescue

Yu-Jun Zheng, *Member, IEEE,* Qing-Zhang Chen, Hai-Feng Ling, and Jin-Yun Xue

Abstract—During disaster events, timely and targeted information provision and exchange could provide great help to the stricken population in difficult and complicated environments. This paper reports a service-oriented system, called Rescue Wings, for providing emergency support to sufferers and rescuers in disasters. The system utilizes mobile services to acquire real-time information about the users and environment, and constructs service agents (servants) to provide active services for mobile users. To perform their functions, the servants frequently invoke a set of intelligent services of Rescue Wings, which can further access a number of public services from government and other public organizations. We identify the most frequent request sequence patterns (FRSP) of Rescue Wings, and develop a new bio-inspired algorithm for efficiently scheduling the requests to minimize the response delay. The system has been tested in several disaster rescue drills, and has been successfully applied to the 2013 Ya'an Earthquake in Southeast China.

Index Terms—Service oriented architecture (SOA), disaster rescue, mobile services, active services, service request scheduling.

1 Introduction

We are now facing increasing threats from natural and man-made disasters. In order to effectively plan and implement disaster rescue operations, first responders have to obtain as much detailed information as possible about population dynamics, paying special attention to special groups such as children, the elderly, and the disabled. It is also expected that disaster and evacuation information should be transmitted to the affected population in a timely and accurate manner, which can have an enormous effect in saving lives and reducing damage. While the importance of customized and intelligent services for assisting the population in emergencies has always been recognized [1]–[4], challenges surrounding the design, deployment, management and integration of such a system have prevented its emergence and/or wide applications.

In the last three years, we have been working on the construction of a service-oriented system for providing emergency support to both responders and the stricken population in disaster rescue operations. The project, named Rescue Wings, is conducted mainly upon a combination of two emerging paradigms: mobile computing and cloud computing. On the one hand, the portability and ease of information storage and dissemination has enabled mobile devices to become one of the most viable means of communication with the population [5], and recent trends in mobile computing have motivated interest in accessing web services from mobile devices in order to extend their functionality and gain access to remote data [6]. On the other hand, cloud computing technologies enable configuring, scheduling, and coordination of shared resources via virtualization, and thus greatly facilitate communication between a broad range of public and private entities [7].

Rescue Wings deploys its client-side applications to mobile devices of end users, who can upload and maintain their profiles on the servers. When entering into the emergency mode, a service agent (servant), whose behaviors are defined by active service programs [8], [9], is instantiated for each Rescue Wings client (of a sufferer or a rescuer). The servants are responsible for actively collecting real-time information and monitoring the states of the clients on the spot, and providing required services to assist the users in self protection, escape, search and rescue (S&R), as well as mutual rescue. When detecting abnormal states, the servants can also remotely request ad hoc operations of the clients. Rescue Wings has access to a number of public services for obtaining rescue information, and provides a set of intelligent services for supporting the responders in different stages of the rescue operation. Here the purpose of the paper is threefold:

1) To present the architectural structure and working mechanism of Rescue Wings, which can provide guidance and assistance for the construction of similar service-based systems.
2) To propose an efficient heuristic algorithm for service request scheduling, which is crucial to the success of the system and can be useful for many other service scheduling problems.
3) To present simulation results and real-world applications of the system, the lessons learned from which can benefit both the system developers and

Manuscript received ...(date to be filled in by Editor).
Y.J. Zheng and Q.Z. Chen are with the College of Computer Science & Technology, Zhejiang University of Technology, Hangzhou, 310023 China (e-mail: yujun.zheng@computer.org).
H.F. Ling is with the College of Field Engineering, PLA University of Science & Technology, Nanjing, 210007 China.
J.Y. Xue is with the Jiangxi Provincial Lab of High-Performance Computing, Jiangxi Normal University, Nanchang, 330022 China.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

the disaster managers.

For illustration, we first present a simple case study of a servant's activities in assisting its client in a fire evacuation. By default, the option of location based services (LBS) is enabled on the client device. The servant continually gets the user location from LBS (every 15 seconds by default) and traces his/her movement path. Ideally, if the path moves towards an exit and the movement speed does not decrease too much, no action is needed; However, at a time the servant observes that the direction of the path changes towards a dangerous area. In response, the servant immediately sends an alert to the client, meanwhile loads the surrounding map of the client, marks the dangerous area together with the user location, and recommends an escape route to the user. The procedure continues until the user returns back to the right route, or the user manually stops receiving information. During the process the servant also invokes functions of other components including the *State analyzer* and the *Router* on the Rescue Wings server(s), which can further access external web services through the Internet. Finally the user successfully escapes, and the servant process terminates after observing this or receiving a confirmation messages. Fig. 1 presents the activity sequence diagram of the case study.

Since the middle of 2012, we have participated in several rescue drills and tested the effect of Rescue Wings in rescue support. The first real-world application of Rescue Wings was the 2013 Ya'an Earthquake in Sichuan province, China. During the rescue operation, there were more than four hundred mobile devices connected to the Rescue Wings servers, around 32,000 service requests had been processed, and at least 58,000 messages had been produced, transmitted, and utilized for assisting the stricken people and supporting the responders. According to on-the-spot investigation and post hoc analysis, the services provided by Rescue Wings contributed effectively to the performance and success of the operation. Until now, the client-side applications of Rescue Wings have been deployed to more than 18,000 end users.

The remainder of the paper is structured as follows: Section 2 discusses related work. Section 3 presents the framework of Rescue Wings and introduces the functionalities of Rescue Wings clients, servants, and rescue support services. Section 4 proposes an efficient bio-inspired algorithm for scheduling the requests to services of Rescue Wings. Section 5 introduces the experiments of Rescue Wings in two drills, Section 6 describes the application of the Ya'an Earthquake rescue operation, and Section 7 concludes.

2 Related Work

Early research on disaster management focused on individual problems in relief operations, such as emergency facility location, vehicle routing, evacuation planning, etc [10]–[13]. The development of information technology including Internet, database systems, and artificial intelligence, enables to integrate and coordinate a variety of operational tasks in decision-support systems (DSS). Wallace and Balogh [14] proposed a framework for categorizing decision making in disaster management into three levels including operational control, managerial control, and strategic planning and policy. They stressed that those tasks that are operational and structural should be the first candidates for computer assistance, permitting "what if" assessments. DSS may be used to support decisions in response to events that may never occur, but that does not reduce their values in training and preparedness.

DSS for disaster management must deal with uncertain and changing environments. Considering that organizations have to be prepared to improvise during response activities, Mendonca et al [15] developed a framework to support emergency managers in responding to real-time events in situations requiring either modification or creation of courses of action, which was assessed by a project at the Port of Rotterdam, Netherlands. Bryson et al. [16] used a similar idea in modeling disaster recovery planning, and proposed a mathematical programming approach to help the decision maker to select among competing plans. Rolland et al. [17] addressed multiple resource-constrained project scheduling in disaster response and recovery in a DSS, using hybrid meta-heuristics and a ground-up manner, considering that planning from the ground up by local response personnel is often more useful than a top-down plan developed by managers who are removed from the scene. Interested readers can refer to [18] for a categorization of DSS models in disasters.

In the last decade, geographical information system (GIS), global positioning system (GPS), and wireless communication technologies have been widely used to improve the speed of response in disaster management DSS. Chang et al. [19] illustrated the use of GIS for the management of chemical emergency events in an urban environment, putting emphasis on integrating relevant spatial data with useful multi-scale models under a user-friendly interface. Aim at facilitating quick emergency response to terrorist attacks like September 11th, 2011, Kwan and Lee [20] proposed an emergency management system to incorporate 3D GIS functionalities for representing the 3D structure of micro-spatial environments (e.g. internal structure of buildings) and conducting GIS-based analyses to provide real-time navigation guidance both for reaching the disaster site and for negotiating within a multi-level structure under emergency situations. They stressed that such a emergency response system should be built upon a highly flexible and distributed system architecture, where the 3D GIS database and decision support functionalities remain accessible to emergency personnel through multiple channels via wireless and mobile communications technologies.

Nicolai et al. [21] presented the design of a GIS-based DSS for North West Indiana region utilizing grid computing and visualization resources at Purdue University

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

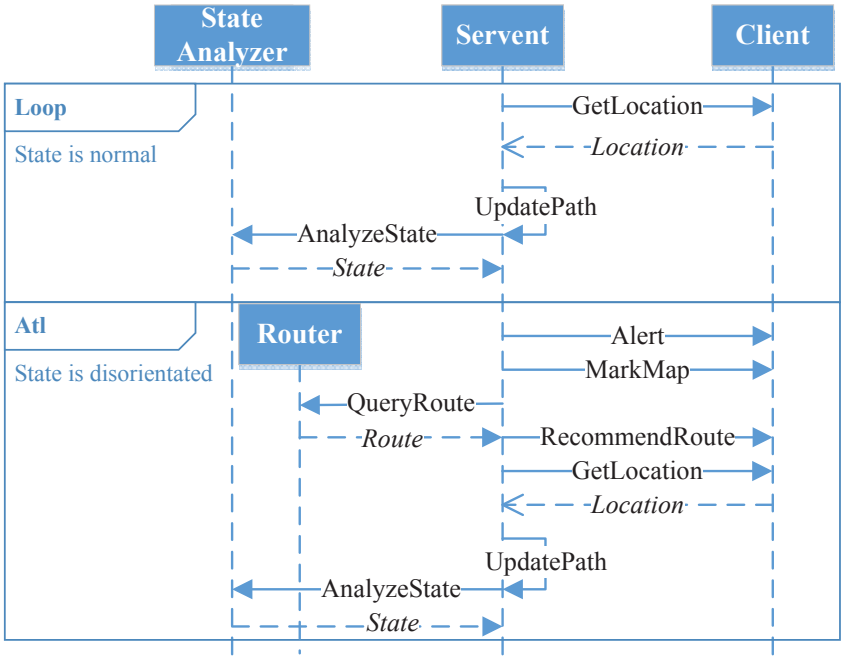


Fig. 1. The activity sequence diagram of the case study.

Calumet. The networking and hardware portion of the system consists of wireless handhold devices which can be brought to a disaster site and collect indigent demographic information, and thus support reliable emergency communication between affected population and local emergency management agency. The integration of GIS, GPS, and mobile technologies has also reported in emergency response for various disasters [22]–[24].

However, information exchange in traditional DSS is slow and often error prone, mainly because that different semantics of data sources present barriers to interoperability, which are especially unacceptable in emergency situations. Web services technology, which enables high-level interoperability among heterogeneous resources in the network, provides much more flexible mechanisms of interaction and coordination of business processes distributed across different organizations.

In [25] Hashmi et al. described a DSS that couples the efficient data collection of sensor networks with the flexibility and interoperability of a web services architecture, which is robust and scalable to meet the needs of emergency medical response. Tanasescu et al. [26] employed semantic web services to construct a DSS, which utilizes contexts including use cases and user roles and locations to enhance the emergency decision process. They also presented an application to a snowstorm emergency on the M11 motorway, 2003.

Jones and Thiebaut [27] presented a system for managing visitors in hazardous areas of process plants, using ubiquity and near real time operation of web services to connect external data, identify suspicious actions, and provide suggestions for decision-making. Weiser and Zipf [1] evaluated the suitability of web service orchestration technology in disaster management scenarios, and presented an example of an evacuation scenario after a bomb has been found, using the spatial data applications of a free GIS platform to support typical use cases for the fire department. The emergency DSS proposed by Smirnov et al. [28] uses web services to acquire information from heterogeneous sources and integrate distributed system components on facilities including mobile phones for receiving assignments. Li et al. [29] developed a Web-GIS for emergency management based on unified geo-spatial framework and web Services. The service-oriented architecture enables a multilevel distributed management of emergency events in a loosely coupled manner.

Our Rescue Wings system is an intelligent DSS for supporting disaster rescuing at the operational level according to the classification of [14]. Its front-end focuses on the accessibility to mobile devices, and its back-end integrates information from heterogeneous sources including GIS, SMS, and other public services using a web services architecture based on previous work [1], [26], [28]. Currently the GIS services of Rescue Wings are based on 2-D maps (unlike [20]) due to the bandwidth limit in rural areas in China (but most areas have been covered by local e-map services such as Baidu Map [30]).

3 The Framework of Rescue Wings

Rescue Wings consists of two generally independent systems. The first is a regular web application, which provides a number of web pages containing documents and other materials and can be accessed through mobile browsers of the clients. After becoming registered members, the client users can also upload and maintain their personal information, receive customized information, and participate in forum discussions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

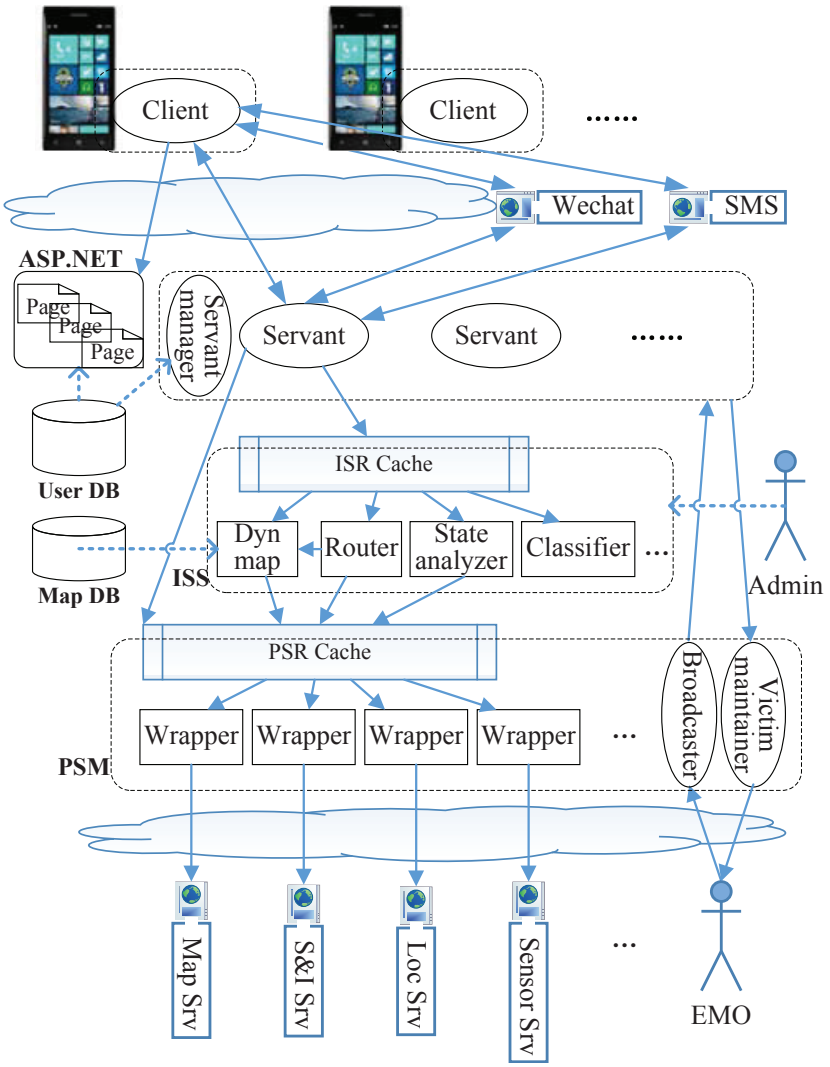


Fig. 2. The framework of Rescue Wings (ISS: intelligent support services; ISR: intelligent service requests; PSR: public service requests; PSM: public services manager; EMO: Emergency management organization).

The second system is the emergency server application, which is the key part of Rescue Wings. It can be further divided into the following three parts:

- A population of servants maintained by a *Servant Manager*. Each servant is responsible for communicating with and supporting a client user.
- The *intelligent support services (ISS)*, which provides a set of intelligent services for the servants and an ISR cache queuing unprocessed requests.
- The *public services manager (PSM)*, which contains a set of wrappers responsible for communicating with public services, a cache for queuing public service requests (PSR), a *Broadcaster* that directly sends important messages from the emergency management organizations (EMO) to the servants, and a *Victim maintainer* that maintains victim information in a view of the EMO. Most of the public services, such as Baidu Map API and China WeatherWS, are accessed via the HTTP protocol.

Fig. 2. presents the basic framework of Rescue Wings. Normally, Rescue Wings runs an instance of the emergency application on a master server, and provides emergency support for some individual requests. When dealing with a large-scale emergency event, Rescue Wings often initializes a new application instance on one or more servers, and the components of the application can reside on different physical or virtual servers.

3.1 Mobile Client Applications

A Rescue Wings client has two modes: normal mode and emergency mode. Under the normal mode, the user can visit the regular web site of Rescue Wings through the client browser. There are three ways to switch the client to the emergency mode:

1) The client actively sends a request for help (RFH) to the Rescue Wings emergency server. This is a one-button operation of the mobile device.
2) The Rescue Wings automatically identify the client in a disaster-stricken area, i.e., identify the client user as a (potential) disaster sufferer.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

3) The client actively requests to be a rescuer in an emergency event.

Fig. 3 presents the main interface of a client application in the emergency mode, which follows the guidelines of user interface design of emergency systems [31]–[33]. The listbox at the top of the screen scrolls the important messages from the Rescue Wings server. The icon buttons at the bottom are respectively for quickly turning on/off the voice prompts, dialing frequently used telephone numbers, searching for important targets (e.g., the closest exit, shelter, victims, etc.), opening the common tools, and changing the basic settings of the application. The center part is a pivot control, which consists of three items (pages):

- The *Escape Map* page for displaying the map of the disaster area, which dynamically marks and updates important objects such as victims, rescuers, escape paths, exits, dangerous zones, etc. Typically, the location of the user is set as the center.
- The *Live Video* page for displaying real-time video of environments around the important targets in the disaster area.
- The *Professional Support* page for communicating with the professionals who can provide help through textual, graphical, or voice messages.

Between the Rescue Wings clients and servants, some short messages and signals are transmitted through the SMS of the telecommunication service provider, and other data such as long messages, pictures, videos are mainly transmitted through Wechat, a third-party value-added data service. We have developed different versions of the Rescue Wings client application to support major platforms including Android, iOS, and Windows Phone. The client can invoke programs including recorders, cameras, accelerometers, compasses, gyroscopes, etc., which are increasingly commonly used in today's smart phones. We have also customized a Rescue Wings mobile phone equipped with additional sensors including thermometers, barometers, light sensors, smoke sensors, pulsimeters, sphygmomanometers, which can collect important information about users and their surrounding environment.

Fig. 3. The main interface of the Rescue Wings client application.

3.2 Servant Functions

When a client is identified as a disaster-stricken user, a servant is constructed for the client based on its user profile, and uses the frame-based approach [34] to maintain the user instance which contains a set of basic properties from the user profile, a set of state properties inferred from the basic properties (as summarized in Table 1), as well as real-time information collected by the client application. As a discrete state property, *MoveState* is used for describing the move state of the user, the values of which include *Fast*, *Normal*, *Slow*, *Very_slow*, *Trapped*, and *Unknown*. The *RIA* property is for describing the degree of the user requiring immediate (emergency) assistance, the values of which include *Not_required*, *Very_low*, *Low*, *Normal*, *High*, *Very_high*, and *Perhaps_too_late*.

In general, not every basic property will be available for a user instance. For example, the *PhysCondition* property is composed by sub-properties such as *Heart_rate* and *Body_Temperature*, the values of which need to be acquired by the client application through dedicated sensors of the device (such as the Rescue Wings customized phone). But a majority of users do not own such special devices, thus the servant also maintains a list of available sensors of the client device, and determines which sensor data it can obtain remotely from the client.

For a user identified as a sufferer, the servant first adds a record to *Victim maintainer*; During the operation, all RFH and other critical information of the user will be passed to the EMO. Another major task of the servant is to obtain the user's real-time location, and maintain the user's movement path in its *Movement* property. The path is dynamically updated and analyzed to infer the *MoveState* of the user. Based on the values of *MoveState* and other related properties, the servant determines its further actions for supporting the user, as depicted in the case study in Section 1. If the servant detects some abnormal states of the user or the environment, it will report to the EMO on behalf of the user.

For a user identified as a rescuer, the behavior pattern of the servant is similar to that of a sufferer servant, because that the most threats are posed to sufferers and rescuers in the same manner in a disaster. The main difference is that the emergency mode of a rescuer is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

TABLE 1
The properties of the user instance maintained by a servant.

Category	Type	Properties
Basic	Simple (scalar) Composite	UserID, Name, Age, Gender, Eyesight, Weight, Length, PhysDisable, Hearing, Disease, Injured Surrounding, Location, Movement, PhysCondition
State	Simple	MoveState, RIA

classified into three submodes: *Searching* (for one or more victims), *Rescuing* (for victims on-the-spot), and *Escaping* (with victims), and the servant provides further targeted services according to the current submode of the rescuer.

3.3 Rescue Support Services

To support client users in evacuation or rescue operations, the servants need a number of services, most from ISS and a few from PSM. Moveover, many services provided by ISS need to access public services from PSM.

In an emergency event, one of the busiest components of Rescue Wings may be the *Broadcaster*, an agent that provides active service for broadcasting messages from the EMO to the servants. Considering the limited bandwidth under emergency situation, the messages are ranked into three levels according to their importance:

Level 1. The message will be sent only once, and no feedback is needed.

Level 2. The message is relatively important and needs feedback to confirm its reception. If there is no feedback, the servant will send the message for at most three times.

Level 3. The message is very important. During the validity the servant will continually send the message until the user confirms its reception.

The underlying congestion control and loss recovery mechanisms are implemented by SMS and Wechat [35].

Dyn map is an intelligent component for providing dynamic map services. After an emergency server is set up, *Dyn map* loads a set of underlying maps about the disaster affected area (mainly from Baidu map service or its caches in Rescue Wings), and constructs a dataset of focused targets, e.g., escape routes, exits, shelters, and hazard sources, in relation to the area. During the disaster operation, the *Dyn map* continually gets the changes of the disaster area and the targets by:

1) Actively requesting data from public services (such as those of GIS and road monitoring systems).
2) Receiving data from the sensors of Rescue Wings clients.
3) Accepting manual inputs of the Rescue Wings administrators.

Consequently, the *Dyn map* dynamically updates the maps and targets, based on which the servants can construct and update the surrounding maps for the users. If a user moves out of the current region, its servant actively requests the service from the *Dyn map* and sends the new map to the client. On the other hand, if the environment changes remarkably, the *Dyn map* actively sends the changes to related servants to update the users' surrounding maps. To reduce network traffic congestion, only incremental changes of cartographic data are transmitted from the server to the clients [36].

Services of another important intelligent component, *Router*, are used for producing escape or rescue routes on the maps of disaster affected areas. The core of *Router* is a set of algorithms including:

- A kth-shortest path algorithm [37], which produces routes in some simple graphs abstracted from the map data.
- A heuristic constrained shortest path algorithm [38], which produces efficient parametric shortest routes under hard constraints.
- A heuristic stochastic shortest path algorithm [39], which produces efficient routes in an uncertain environment.
- A set of heuristic algorithms for producing efficient composite routes passing through multiple intermediate nodes [40], which are mainly used for a rescuer with multiple targets.

When given one source position and one or more target positions, *Router* automatically determines which of the above algorithms will be used based on the situation of current region. Moreover, when computing the costs of the paths, *Router* takes into consideration not only the path lengths, but also the route conditions, the user's speed, the expected traffic on the paths, as well as the (potential) threats along the paths.

In some cases, the servants request the services of *Router* by passing the exact positions of the targets. In other cases, the servants only give the types of targets (such as victims, shelters, hazard sources, etc.), and *Router* first searches for the positions of the targets by calling for the services of *Dyn map*, and then uses built-in algorithms to produce routes. Whenever possible, *Router* will recommend more than one route in response to a request, in order to enhance the decision support without increasing the response time.

State Analyzer provides a service that receives the user movement path from a servant, and infers the move state of the user. The inference result, together with some other properties of the user, will be sent to the *Classifier* to infer the RIA index of the user. Both *State Analyzer* and *Classifier* use the Takagi-Sugeno type neuro-fuzzy techniques networks for state inference [41].

With the increasing number of servants, the number of requests and the computational load of ISS services increase rapidly. Consequently, the number of requests to public services also increases sharply. Next we describe

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

the method for scheduling requests to ISS and PSM.

4 Service Requests Scheduling

4.1 The Basic Scheduling Scheme

During an emergency, the request arrival rate can increase dramatically, which leads to an overload and ultimately the thrashing effect [42]. A number of request scheduling algorithms have been proposed (e.g., [42]–[46]). In Rescue Wings, any request for *Update_Targets* (updating the focused targets) is preferentially sent to available service instances of the *Dyn map*, and all the requests for other ISS services are scheduled using a priority based scheme [46] in combination with a frequent-request-sequence-pattern (FRSP) based approach. That is, the requests are grouped by *UserID* of the servants. Each request waiting to be processed is queued in the *ISR Cache* and assigned a priority according to the RIA value of the user, as described in Table 2.

If the total number of cached requests is less than a threshold (typically set as 200 in the ISS), the requests will be sent to the available services in decreasing order of $p(t + t_0)$, where p is the priority of the request, t is the waiting time of the request so far (in seconds), and t_0 is a constant typically set as 5. Suppose a request A with priority p_A has been waiting for t_A seconds and another request B with priority p_B has been waiting for t_B seconds, under this scheme A is to be processed before B when:

$$t_A > \frac{p_B}{p_A}(t_B + t_0) - t_0 \tag{1}$$

Otherwise, the requests belonging to the FRSP are picked out and scheduled by the heuristic algorithm described in the next subsection, and other requests remain in the *ISR Cache* and wait to be processed in decreasing order of $p(t + t_0)$.

In PSM, there is no obvious FRSP, and requests in the *PSR cache* are scheduled using the above priority-based scheme.

4.2 The FRSP Scheduling Problem

FRSP identifies those requests which frequently arrive and/or need to be served in a specified sequence. Currently there are two typical FRSP in Rescue Wings:

1) $UpdateSurroundingMap \rightarrow QueryRoute \rightarrow AnalyzeState \rightarrow AnalyzeRIA$;
2) $QueryRoute \rightarrow AnalyzeState \rightarrow AnalyzeRIA$.

The second request sequence is a subsequence of the first one. We note that, if a servant calls for an $UpdateSurroundingMap$ service, in almost all cases it will call for the next three services in sequence. Thus we treat each individual request for $UpdateSurroundingMap$ as a sequence of the first FRSP.

Scheduling of the requests belongs the FRSP can be considered as a permutation flowshop scheduling problem [47]: Available services are regarded as machines, each group is regarded as a job, and the problem is to find an optimal request schedule that minimizes the makespan $C(\pi_n, m)$, i.e., the completion time of the last job (request) π_n on the last machine (service) m. But a difference of our problem is that, for a job derived from a request for $UpdateSurroundingMap$, the starting time is larger than zero.

The expected processing time of each service operation is estimated based on the following empirical equations.

- $AnalyzeState$:

$$t_1 = t_1^*(1 + a_1 l_P)^{C_P} \tag{2}$$

- $AnalyzeRIA$:

$$t_2 = t_2^*(a_2 - a_3 n_A) \tag{3}$$

- $QueryRoute$:

$$t_3 = t_3^*(1 + a_4 n_T)^{C_R} \tag{4}$$

where t_1^*, t_2^* and t_3^* are three basic processing times (in minutes) related to the computational speed of the emergency server, l_P is the length (i.e., the number of points) of the input path, n_A the number of available properties of the target user, n_T the number of target positions, C_P the complexity of the graph abstracted for the surrounding environment around the path, C_R the complexity of the graph containing the target positions, and a_1–a_4 are the coefficients whose typical value ranges and value setting in Rescue Wings are given in Table 3.

4.3 A Heuristic Scheduling Algorithm

We propose a very efficient scheduling algorithm, named PBBO-LS, which combines biogeography-based optimization (BBO) [48] with greedy local search. BBO is a metaheuristic optimization method inspired by the island biogeography, where a solution is analogous to an island, and its fitness is analogous to the habitat suitability index (HSI) of the island. Central to BBO is the equilibrium theory of biogeography, which indicates that high HSI islands have high species emigration rates and low HSI islands have high species immigration rates. Here we use a simple linear migration model, where the immigration rate λ and the emigration rate μ are calculated based on the ranking of fitness (where n is the population size and i is the ranking of the island):

$$\lambda_i = \frac{i}{n} \tag{5}$$

$$\mu_i = 1 - \frac{i}{n} \tag{6}$$

PBBO-LS first initializes a population of solutions (request sequences) to the scheduling problem, and then continually evolves them using three operators:

- *Migration*, which migrates subsequences probably from high-quality solutions to low-quality ones based on their immigration and emigration rates.
- *Mutation*, which disturbs a solution by randomly choosing and reversing some subsequences.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

TABLE 2
The relationship between the priorities of requests and the RIA property of the user.

RIA	Not_required	Very_low	Low	Normal	High	Very_high	Perhaps_too_late
Priority	0.1	0.25	0.4	0.6	0.8	1.0	0.5

TABLE 3
The parameter settings of the FRSP Scheduling Problem (where a_4 is used when exact target positions are given, while a_4' is used when only target types are given).

Parameter	a_1	a_2	a_3	a_4	a_4'
Typical range	[0.03,0.1]	[2,5]	[0.03,0.3]	[0.12,0.3]	[0.12,0.3]
Value in Rescue Wings	0.06	2.87	0.052	0.18	0.21

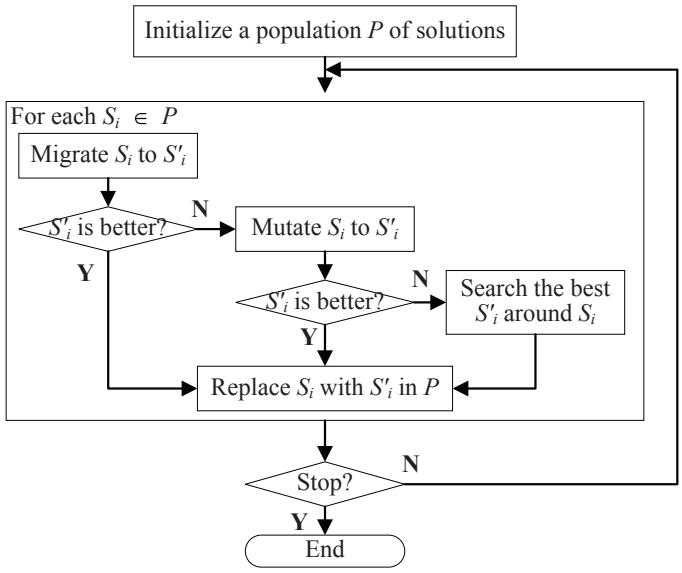


Fig. 4. The algorithm flowchart of PBBO-LS.

- *Local search*, which drops a random subsequence from the solution, and then reinsert its element into the remaining sequence based on a greedy heuristic.

At each generation, PBBO-LS first performs migration on each solution S_i; if the solution is not improved, it then tries to mutate S_i to a better solution; If the mutation also fails, it then tries to improve S_i by local search. Whenever a better solution S_i' is found, it replaces S_i in the next generation, as illustrated by Fig. 4.

We have compared PBBO-LS with some state-of-the-art methods [49]–[51] on a set of test problems, which demonstrates that PBBO-LS is very competitive in both computational cost and solution accuracy. The simulation results and the psuedo-code of PBBO-LS are given in the Appendix.

5 Experiments in Rescue Drills

Rescue Wings has been tested in several rescue drills. Here we present its application in two typical fire drills.

5.1 The First Fire Drill

The first drill was held in a high school for cultivating and testing the abilities of teachers and students in fire disasters. The drill started at 1:00 pm and lasted for 1.5 hours in a school day. A total of 15 classes participated in the drill. We selected a class of 32 students and three teachers, and equipped them with mobile phones which have been installed with Rescue Wings Clients.

The application process of Rescue Wings in the drill can be summarized as follows:

1) At the beginning of the drill, Rescue Wings initialized an emergency server, remotely started the emergency mode of the 35 client applications, and automatically constructed a servant for each client.
2) The servants sent alerts to their clients, and used the LBS to obtain the positions of the users. The results showed that except a teacher (denoted by T1), all the other 34 users were in the fire building.
3) During 1:00∼1:03 pm, the servants kept track of the clients, and judged that one teacher (denoted by T2) and 18 students were escaping towards two exits, 6 students were moving in wrong directions, and one teacher (denoted by T3) and 8 students were trapped. Among the trapped users, 5 students sent RFH to the Rescue Wings server. The teacher T1 moved towards the fire building, meanwhile requested Rescue Wings for disaster information and manually set the state to "Serving as a rescuer".
4) In response, the servants of the 6 students in wrong directions sent new alerts, and marked the dangerous areas and recommended escape routes on the client maps; For the 9 trapped users, Rescue Wings selected 3 among them, opened the cameras of their devices and tried to take real-time pictures of the scene; Moreover, Rescue Wings found out the student closest to T1, calculated a route from T1 to this student, and sent the route and the information about the student to T1.
5) During 1:04∼1:10 pm, teacher T2 and 17 students successfully escaped. T1 entered into the building and his state was updated as "Searching for victims". Among the remaining 15 students, 7 were escaping towards the exits and 8 were identified to be trapped.
6) At 1:11 pm, the fire brigade arrived. T2 also set his state to "Serving as a rescuer", and thus his servant queried the information of trapped students and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

the surrounding environment. Afterwards, T2 and other three fire rescuers entered into the building and moved to the trapped students.

7) During 1:11∼1:16 pm, 7 students escaped by themselves; By 1:29 pm, all the remaining students successfully escaped with the help of the rescuers.

The rescue task for the chosen class supported by Rescue Wings was completed in 29 minutes. However, the average evacuation time of the other 14 classes was 52 min, and the total evacuation time of the school was 79 min. The result showed that Rescue Wings helped to improve the evacuation and rescue efficiency greatly.

During the operation, Rescue Wings had sent 1261 messages (including 71 voice alerts) to the users. There are total 2625 invocations of active services of the servants. The most used service is "GetLocation", the number of requests for which is 2388; The other four most important services are "MarkMap", "RecommendRoute", "RFH", and "TakePicture". However, the most used ISS services requested by the servants are those services defined in the FRSP, the number of requests for which accounts for about 89% of that for all the ISS services, as shown by the variation curves in Fig. 5(a).

After the drill, we reproduce the requests for services in a simulation environment, and test the request scheduling performance of the following three methods:

- The basic scheduling scheme which does not use FRSP (Section 4.1).
- The basic scheduling scheme together with a discrete particle swarm optimization (PSO) algorithm [50] for FRSP scheduling.
- The basic scheduling scheme together with a modified differential evolution (DE) algorithm [51] for FRSP scheduling.

Fig. 5(b) compares the variation of the number of cached requests (NCR) of the four methods during the drill. As we can see, the basic scheduling scheme without FRSP performs badly; among the three FRSP scheduling algorithms, PBBO-LS results in the smallest NCR and thus provides the most efficient response.

5.2 The Second Fire Drill

The second drill dealt with a fire following an explosion in a chemical factory. The area covered about 6,000 square meters, and the drill started at 4:00 pm and ended at about 6:00 pm. Among more than 450 participants, 116 workers were users of Rescue Wings. The process of the drill can be divided into the following four stages:

1) From 4:00 to 4:03 pm, the fire broke out in a building, Rescue Wings initialized an emergency server, identified 15 workers in the building, started the emergency mode of their clients and constructed servants for them.
2) From 4:04 to 4:07 pm, the servants communicated with the clients to assist evacuation. Besides, Rescue Wings identified other 21 users coming to fight against the fire and included them to the emergency.
3) From 4:08 to 4:22 pm, the fire caused secondary explosion and spread to the whole factory area. Up to 4:13 pm, Rescue Wings identified and included the remaining 80 users. Since then, Rescue Wings assisted the users in the larger scale evacuation, and did not encourage anyone to search & rescue others.
4) At 4:23 pm, the fire brigade arrived. From 4:23 to 6:00 pm, Rescue Wings kept track of the users and reported their status to the fire brigade, and continued to assist the users in evacuation.

The drill lasted for about two hours. Nonetheless, up to 5:00 pm, 114 users of Rescue Wings had escaped or been rescued. Rescue Wings lost touch with the other two users. It had tried to get their positions and status until 6:00 pm. Finally we found that one had escaped but lost her client device, and the other was rescued at the end of the drill and her device was broken.

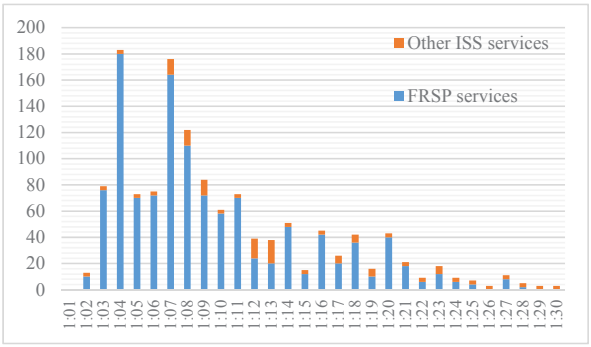
According to statistics, among the successful evacuees, the average escape path length calculated by Rescue Wings was 568.5 meters, the average actual escape path length of the Rescue Wings users was 616.3 m, and their average escape time was 11.71 min. In comparison, the average actual escape path length and escape time of the other evacuees were respectively 830 m and 17.73 min. Among the trapped ones, the average rescue path length and average rescue time of the Rescue Wings users were respectively 375.9 m and 33.55 min, which are also less than 503.7 m and 39.53 min of the others. As a result, the rate of successful escape (denoted by R_{SE}) of the evacuees using Rescue Wings clients was 90.35%, which was much higher than 46.73% of the others.

The variation of service invocations and the comparison of different scheduling methods in the drill are presented in Fig. 6(a) and (b). In this application, the number of requests for FRSP services accounts for about 91% of that for all the ISS services, and PBBO-LS also exhibits the best scheduling performance.

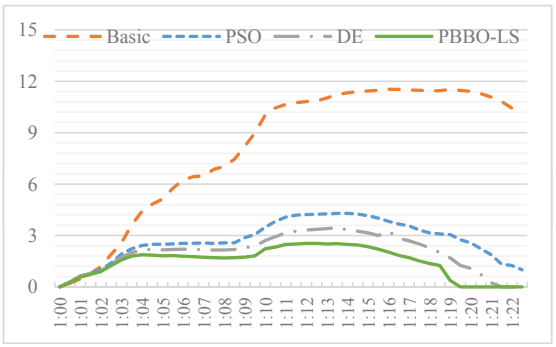
6 Application in Ya'an Earthquake

At 08:02 Beijing Time (00:02 UTC) on April 20, 2013, a 7.0-magnitude earthquake occurred Ya'an district in southwest China, spreading over 19 cities and 115 countries. About 1.99 million people were affected, with 458 thousand resettled, 13.4 thousand injured and 196 killed by the time 18:00, 26 April (http://news.sina.com.cn/c/2013-04-27/090126968052.shtml). National and local governments started the emergency response plan within 15 minutes, established the disaster relief command center, and activated a set of public emergency services in two hours [52]. Rescue Wings immediately started an emergency server and launched services for rescue support. During a 96-hour operation, Rescue Wings had communicated to 320 stricken users (sufferers) and 93 rescuers, and provided a very effective support in evacuation and rescue. The section reports the process and then analyzes result of the application of Rescue Wings in this real-world rescue operation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

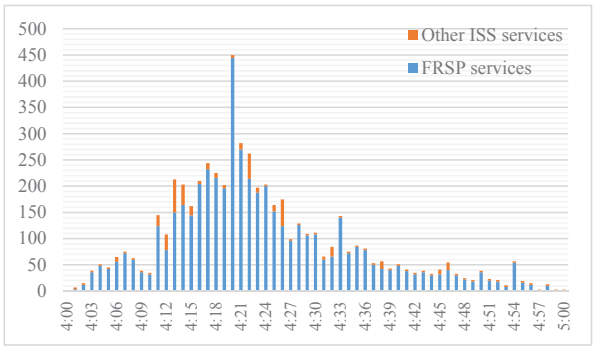


(a) Number of ISS services requested by the servants

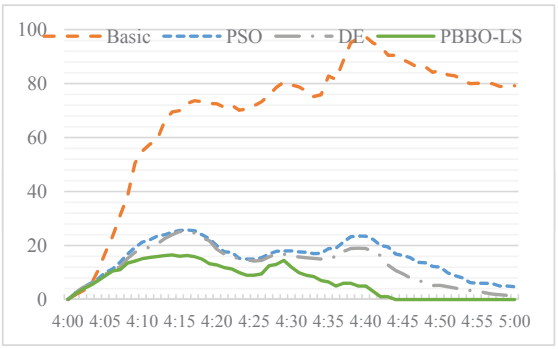


(b) Number of cached requests resulted by the different scheduling methods

Fig. 5. Statistics on the application of Rescue Wings in the first rescue drill.



(a) Number of ISS services requested by the servants



(b) Number of cached requests resulted by the different scheduling methods

Fig. 6. Statistics on the application of Rescue Wings in the second rescue drill.

6.1 Application Process

In general, the application process of Rescue Wings in Ya'an Earthquake can be divided into five stages.

Stage 1. At 8:08, April 20, Rescue Wings center got an emergency message from the China Earthquake Networks Center and immediately initialized an emergency server. At 8:11, it also received an urgent notice from the army's emergency operations center, and thus affirmed the event and broadly defined the affected area.

Stage 2. Up to 8:35, Rescue Wings had identified 268 client devices in the affected area, and started the emergency mode and constructed servants for them. Moreover, at 8:32, a rescue team equipped with 6 client devices set out from Chengdu and arrived at the stricken area after about 90 minutes. During the first four hours of the emergency, the tasks of Rescue Wings mainly included alerting the affected users, receiving and processing RFHs, and recommending emergency shelters and evacuation routes.

Stage 3. From 12:00, April 20 to 6:00, April 21, Rescue Wings had connected with 375 client devices, including 305 sufferers, 22 registered rescuers, and 49 volunteers serving as rescuers. The main task of Rescue Wings was to provide help to trapped sufferers, including sending them self-rescue information and gathering their information for supporting the rescuers. In particular, during the night of April 20, 32 Rescue Wings CSRs had communicated to and provided support for 106 users.

Stage 4. From 6:00, April 21 to 6:00, April 22, dozens of aftershocks continued in the disaster area, and a number of users previously had escaped or been rescued were in danger again. In this stage the main tasks of Rescue Wings included acquiring real-time disaster information from public services, sending new alerts, and providing support for sufferers and rescuers. Moreover, since the morning of April 21, the command center discouraged volunteers from coming into the disaster area, and thus Rescue Wings forwarded the message to the users outside the disaster area, and did not create servants for users requesting for serving as voluntary rescuers.

Stage 5. Up to the morning of April 22, among Rescue Wings users, most sufferers had been reported "Action completed". In the remaining time of the rescue operation, the main task of Rescue Wings was to assist the rescuers in searching and rescuing other victims. At 8:00, April 23, the servants of all the sufferers and of most rescuers were destructed, and the Rescue Wings server mainly served as a platform for the communication and information sharing among the users.

The distributions of Rescue Wings users at 11:00 and 23:00 of April 20-22 are presented in Fig. A.3 in the Appendix.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing

6.2 Results and Discussion

During operation of about 72 hours (from 8:08, April 20 to 8:00, April 23), Rescue Wings had sent 18,862 messages to the users, involved 15255 invocations of servant services, 57197 requests of ISS services (about 89% FRSPservices) and 3047 public services, and used 3197MB network traffic (2728MB inner traffic and 469MB outer traffic). The variation curves of the messages, service invocations, and network traffic are presented in Fig. 7(a)–(c) respectively. As we can see, the number of service requests and the network traffic quickly reached a peak during the first two hours of the emergency event, and then gradually decreased until a dense aftershock sequence arrived. During the last two stages of the operation, the number of service requests and the traffic dropped rapidly due to the decrease number of online Rescue Wings users.

We also conduct a simulation reproducing the service requests in the event to test the scheduling performance of the other three methods used in Section 5, and present the comparative result in Fig. 7(d), which shows that PBBO-LS always reaches the smallest NCR in the event.

The main roles of Rescue Wings in the rescue operation are threefold:

1) For sufferers trapped immediately by the earthquake or aftershocks, Rescue Wings clients provided location information to and kept communication with the command center and rescuers, which greatly facilitated the rescue processes.
2) For sufferers escaped by themselves, Rescue Wings clients receives important messages (such as risk warning and route guidance) from the center server, which can be crucial to the escapers.
3) For rescuers, Rescue Wings helped to search and locate the victims, collect real-time information about them, and communicate with them when necessary, which greatly improved the rescue efficiency.

6.2.1 Support for Sufferers

There were 320 Rescue Wings users identified as disaster sufferers, but 4 had their devices lost or damaged and one has been missing in the event. Among the remaining 315 sufferers:

- 9 had been trapped immediately by the earthquake and thus had to wait for rescue. By keeping communication with the Rescue Wings center and rescuers, all of them had been successfully rescued, with a median time of 27.2 min (between the receipt of the first RFH and the success of rescue).
- For the other 306 sufferers, 287 had successfully escaped by themselves, with $R_{SE} = 93.8\%$, the median escape path of 262 m, and the median escape time of 3.9 min. During the escape, 18 were slightly wounded and 1 was heavily wounded.
- 19 sufferers tried to escape but failed (5 slightly wounded and 1 heavily wounded during the escape), and thus had to wait for rescue. With the help of Rescue Wings, all of them had been finally rescued with a median time of 22.8 min.

After the operation, we have collected and analyzed information of other 925 disaster sufferers who are not Rescue Wings users. Those sufferers have a similar age distribution as Rescue Wings sufferers (most are young and middle-aged). Among them:

- 23 had been trapped immediately by the earthquake, and the median rescue time was 46.3 min, which was about 170% of that with Rescue Wings.
- 450 had escaped by themselves, with $R_{SE} = 50\%$, the median escape path of 339 m, and the median escape time of 5.2 min. 89 were slightly wounded and 15 heavily wounded, among which 62 and 9 were respectively wounded after 4 min (later than the median escape time of Rescue Wings users). This showed that the rescue time saved by Rescue Wings was a key factor in decreasing the injury rate.
- 452 failed to escape, among which 446 had been rescued (195 slightly wounded and 65 heavily wounded) with a median time of 50.5 min, about 221% of that with Rescue Wings.

Fig. 8 compares the distribution of sufferers of the two groups. In general, the sufferers with Rescue Wings support had significant advantages over those without in terms of both the high success rates and low casualties.

6.2.2 Support for Rescuers

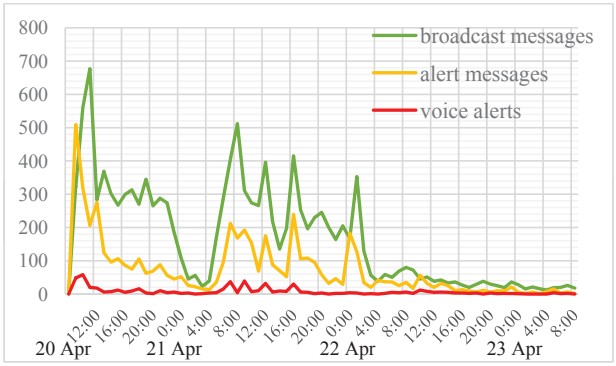
The 93 Rescue Wings rescuers can be divided into two classes: 61 worked independently or with other Rescue Wings rescuers (denoted as C1), and 32 collaborated with other 26 rescuers who were not Rescue Wings users (denoted as C2). We have also investigated 292 rescuers who were not Rescue Wings users and did not collaborated with Rescue Wings users (denoted as C3), and compared their rescue efficiency with rescuers in C1 and C2. During the first 72 hours of the rescue operation, there were 136, 125, and 573 victims rescued by the rescuers of C1, C2, and C3 respectively.

Table 4 presents the efficiency of different classes of rescuers. As we can see, there is no obvious difference between the rescue efficiencies of C1 and C2: The average number of victims rescued per rescuer was around 2.2, the median rescue path length per victim was about 465 m, and the median rescue time per victim was 32∼33 min. However, C3 performs much worse: its average number of victims rescued per rescuer was 1.89, and the rescue path length and rescue time per victim were 635 m and 45.6 min respectively. This also validated the effectiveness of Rescue Wings services in supporting the rescuers in searching and rescuing.

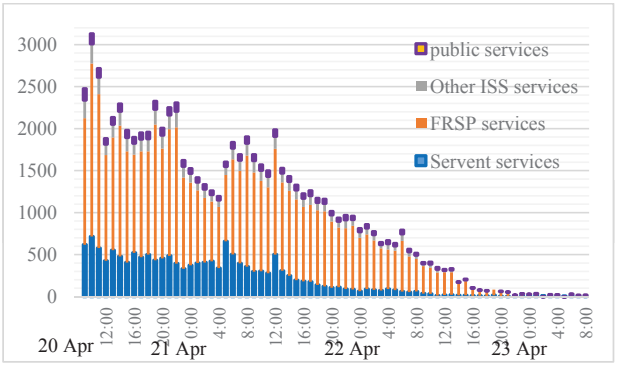
7 Conclusion

The paper presents Rescue Wings, a service-oriented system for assisting sufferers and rescuers in disaster rescue operations. It consists a population of servants for communicating with the client users, a set of internal

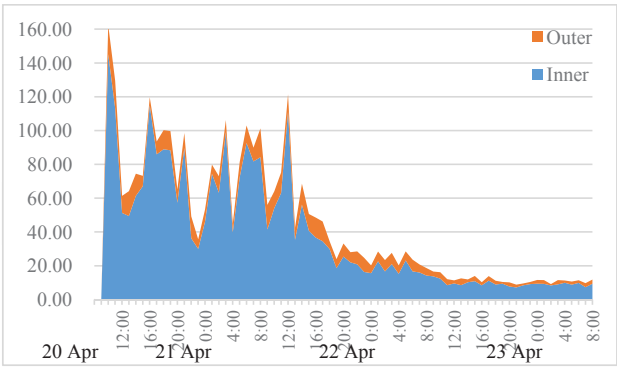
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2401598, IEEE Transactions on Services Computing



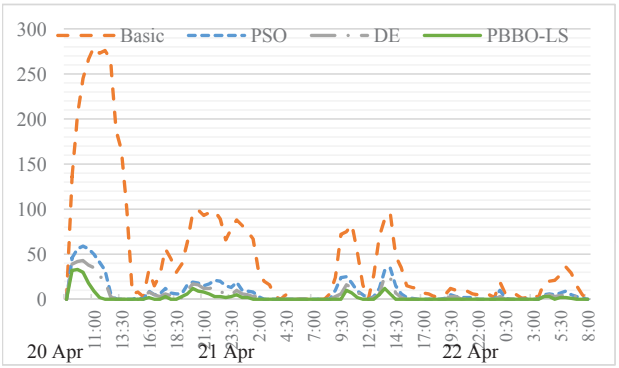
(a) Number of messages/alerts sent by the servants



(b) Number of invocations of Rescue Wings services

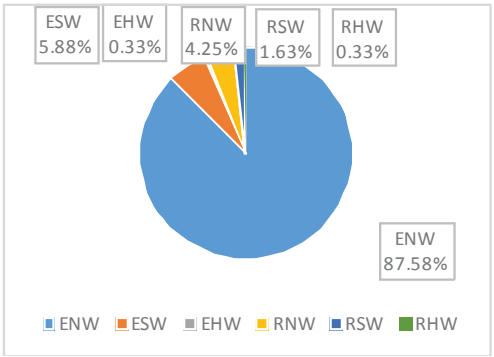


(c) Network throughput between Rescue Wings server and clients (inner) and between Rescue Wings server and public services (outer)

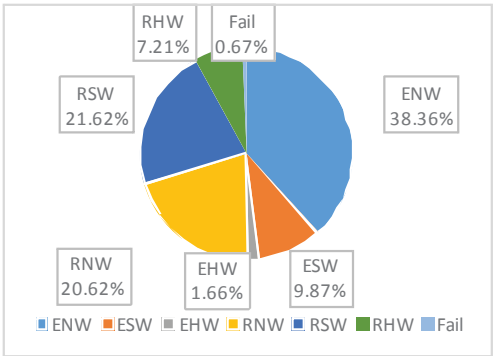


(d) Number of cached requests resulted by the different scheduling methods

Fig. 7. Statistics on the application of Rescue Wings in Ya'an earthquake, 2013.



(a) The distribution of the sufferers that are Rescue Wings users



(b) The distribution of the selected sufferers that are not Rescue Wings users

Fig. 8. The comparison of escape/rescue effects of sufferers in Ya'an earthquake, 2013. (ENW: escaped without wounded; ESW: escaped but slightly wounded; EHW: escaped but heavily wounded; RNW: rescued without wounded; RSW: rescued but slightly wounded; RHW: rescued but heavily wounded; Fail: failed to escape or be rescued)

TABLE 4
Rescue efficiency of different classes of rescuers in Ya'an earthquake, 2013.

Class	Rescuers	Number of victims rescued			Rescue efficiency		
		Total	Slightly wounded	Heavily wounded	Victims rescued per rescuer	Median rescue path length per victim	Median rescue time per victim
C1	61	136	68	30	2.23	468 m	32.6 min
C2	58	125	53	21	2.16	462 m	33.5 min
C3	292	553	252	130	1.89	635 m	45.6 min

intelligent support services, and a number of wrappers for accessing public services. For efficiently scheduling the requests for Rescue Wings services, we develop a new heuristic algorithm based on BBO and local search, which outperforms some state-of-the-art scheduling methods. Rescue Wings has been tested on two rescue drills, and applied to the 2013 Ya'an Earthquake. The results show that the system contributes greatly in assisting the users and improving the rescue efficiency.

Rescue Wings are now attracting more and more users, which makes us pay more attentions in improving the scalability and diversity of services for disaster rescue support. We are currently developing more customized and personalized functions for the Rescue Wings client applications, and promoting the existing service scheduling rules and algorithms to support more concurrent requests without degrading the response time. We also plan to extend the Rescue Wings services by utilizing/integrating sensing services of the Internet of Things [53], [54]. We expect that Rescue Wings could have a broader application and play a more important role in future disaster rescue operations.

ACKNOWLEDGMENT

This work was supported in part by grants from National Natural Science Foundation (Grant No. 61020106009, 61105073, 61272075, 61379023, 61473263) of China.

REFERENCES

[1] A. Weiser and A. Zipf, "Web service orchestration of OGC web services for disaster management," in *Geomatics Solutions for Disaster Management*, ser. Lecture Notes in Geoinformation and Cartography, J. Li, S. Zlatanova, and A. Fabbri, Eds. Springer Berlin Heidelberg, 2007, pp. 239–254.

[2] M. Gloria, V. Lersi, G. Minei, D. Pasquariello, C. Monti, and A. Saitto, "A semantic Web services platform to support disaster and emergency management," in *4th Biennial Meeting of International Environmental Modelling and Software Society*, 2008, pp. 1524–1532.

[3] D. Rosenkrantz, S. Goel, S. Ravi, and J. Gangolly, "Resilience metrics for service-oriented networks: A service allocation approach," *IEEE Transactions on Services Computing*, vol. 2, no. 3, pp. 183–196, 2009.

[4] M. Farnaghi and A. Mansourian, "Disaster planning using automated composition of semantic OGC web services: A case study in sheltering," *Computers, Environment and Urban Systems*, vol. 41, no. 1, pp. 204–218, 2013.

[5] C. Oxendine, M. Sonwalkar, and N. Waters, "A multi-objective, multi-criteria approach to improve situational awareness in emergency evacuation routing using mobile phone data," *Trans. GIS*, vol. 16, no. 3, pp. 375–396, 2012.

[6] H. Artail, K. Fawaz, and A. Ghandour, "A proxy-based architecture for dynamic discovery and invocation of web services from mobile devices," *IEEE Transactions on Services Computing*, vol. 5, no. 1, pp. 99–115, 2012.

[7] K. M. Sim, "Agent-based cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 564–577, 2012.

[8] E. Amir, S. McCanne, and R. Katz, "An active service framework and its application to real-time multimedia transcoding," in *Proceedings of the ACM SIGCOMM'98 Conference on Applications, technologies, architectures, and protocols for computer communication*. New York, NY, USA: ACM, 1998, pp. 178–189.

[9] R. Karrer and T. Gross, "Location selection for active services," *Cluster Computing*, vol. 5, no. 3, pp. 265–275, 2002.

[10] C. Toregas, R. Swain, C. ReVelle, and L. Bergman, "The location of emergency service facilities," *Operations Research*, vol. 19, no. 6, pp. 1363–1373, 1971.

[11] P. Kolesar and W. E. Walker, "An algorithm for the dynamic relocation of fire companies," *Operations Research*, vol. 22, no. 2, pp. 249–274, 1974.

[12] J. H. Johnson Jr, "A model of evacuation–decision making in a nuclear reactor emergency," *Geographical Review*, vol. 75, no. 4, pp. 405–418, 1985.

[13] N. Altay and W. G. Green III, "OR/MS research in disaster operations management," *European Journal of Operational Research*, vol. 175, no. 1, pp. 475–493, 2006.

[14] W. Wallace and F. DeBalogh, "Decision support systems for disaster management," *Public Administration Review*, vol. 45, pp. 134–146, 1985.

[15] D. Mendonca, G. E. Beroggi, and W. A. Wallace, "Decision support for improvisation during emergency response operations," *International Journal of Emergency Management*, vol. 1, no. 1, pp. 30–38, 2001.

[16] K.-M. N. Bryson, H. Millar, A. Joseph, and A. Mobolurin, "Using formal MS/OR modeling to support disaster recovery planning," *European Journal of Operational Research*, vol. 141, no. 3, pp. 679–688, 2002.

[17] E. Rolland, R. Patterson, K. Ward, and B. Dodin, "Decision support for disaster management," *Operations Management Research*, vol. 3, no. 1-2, pp. 68–79, 2010.

[18] S. Asghar, D. Alahakoon, and L. Churilov, "Categorization of disaster decision support needs for the development of an integrated model for DMDSS," *International Journal of Information Technology & Decision Making*, vol. 7, no. 1, pp. 115–145, 2008.

[19] N.-B. Chang, Y. Wei, C. Tseng, and C.-Y. Kao, "The design of a GIS-based decision support system for chemical emergency preparedness and response in an urban environment," *Computers, Environment and Urban Systems*, vol. 21, no. 1, pp. 67–94, 1997.

[20] M.-P. Kwan and J. Lee, "Emergency response after 9/11: the potential of real-time 3D GIS for quick emergency response in micro-spatial environments," *Computers, Environment and Urban Systems*, vol. 29, no. 2, pp. 93–113, 2005.

[21] B. Nicolai, G. Jin, K. Jiang, and C. Winer, "Disaster assessment with parallel image processing for GIS based local area disaster decision support system," in *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application*. New York, NY, USA: ACM, 2010, pp. 1–4.

[22] Z. Hu, X. Li, Y. Sun, and L. Zhu, "Flood disaster response and decision-making support system based on remote sensing and GIS," in *IEEE International Geoscience and Remote Sensing Symposium*, July 2007, pp. 2435–2438.

[23] M.-H. Hsu, A. Chen, L.-C. Chen, C.-S. Lee, F.-T. Lin, and C.-J. Huang, "A GIS-based decision support system for typhoon emergency response in taiwan," *Geotechnical and Geological Engineering*, vol. 29, no. 1, pp. 7–12, 2011.

[24] A. Rasekh and A. Vafaeinezhad, "Developing a GIS based decision support system for resource allocation in earthquake search and rescue operation," in *Computational Science and Its Applications*, ser. Lecture Notes in Computer Science, B. Murgante, O. Gervasi, S. Misra, N. Nedjah, A. A. Rocha, D. Taniar, and B. Apduhan, Eds. Springer Berlin Heidelberg, 2012, vol. 7334, pp. 275–285.

[25] N. Hashmi, D. Myung, M. Gaynor, and S. Moulton, "A sensor-based, web service-enabled, emergency medical response system," in *Proceedings of the 2005 Workshop on End-to-end, Sense-and-respond Systems, Applications and Services*. Berkeley, CA, USA: USENIX Association, 2005, pp. 25–29.

[26] V. Tanasescu, A. Gugliotta, J. Domingue, R. Davies, L. Gutierrez-Villarias, M. Rowlatt, M. Richardson, and S. Stincic, "A semantic web services GIS based emergency management application," in *The Semantic Web*, ser. Lecture Notes in Computer Science, I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, Eds. Springer Berlin Heidelberg, 2006, vol. 4273, pp. 959–966.

[27] A. Jones and O. Thiebaut, "A web-service application for ubiquitous response to emergency situations in hazardous environments," in *Innovations in Information Technology*, Nov 2006, pp. 1–5.

[28] A. Smirnov, T. Levashova, M. Pashkin, A. Krizhanovsky, A. Kashevnik, A. Komarova, and N. Shilov, "Web-service based distributed system for decision support in emergency situations," in *IEEE Military Communications Conference*, Oct 2007, pp. 1–7.

[29] J. Li, X. Tang, Z. Liu, and M. Duan, "Application of WebGIS for government emergency management based on web service," in *International Conference on Information Engineering and Computer Science*, Dec 2009, pp. 1–4.
[30] Baidu map. [Online]. Available: http://map.baidu.com/
[31] I. Rauschert, P. Agrawal, R. Sharma, S. Fuhrmann, I. Brewer, and A. MacEachren, "Designing a human-centered, multimodal GIS interface to support emergency management," in *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*. New York, NY, USA: ACM, 2002, pp. 119–124.
[32] L. Chittaro, F. Zuliani, and E. Carchietti, "Mobile devices in emergency medical services: User evaluation of a PDA-based interface for ambulance run reporting," in *Mobile Response*, ser. Lecture Notes in Computer Science, J. Loffler and M. Klann, Eds. Springer Berlin Heidelberg, 2007, vol. 4458, pp. 19–28.
[33] Y. B. Salman, H.-I. Cheng, and P. E. Patterson, "Icon and user interface design for emergency medical information systems: A case study," *International Journal of Medical Informatics*, vol. 81, no. 1, pp. 29–35, 2012.
[34] M. Minsky, "A framework for representing knowledge," in *Computation & intelligence*, G. F. Luger, Ed. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1995, pp. 163–189.
[35] S. Kasera, S. Bhattacharyya, M. Keaton, D. Kiwior, S. Zabele, J. Kurose, and D. Towsley, "Scalable fair reliable multicast using active services," *IEEE Network*, vol. 14, no. 1, pp. 48–57, 2000.
[36] D. Mioc, F. Anton, C. Gold, and B. Moulin, "'time travel' visualization in a dynamic Voronoi data structure," *Cartography and Geographic Information Science*, vol. 26, no. 2, pp. 99–108, 1999.
[37] H. Maruyama, "A new kth-shortest path algorithm," *IEICE transactions on information and systems*, vol. 76, no. 3, pp. 388–389, 1993.
[38] C. C. Ribeiro and M. Minoux, "A heuristic approach to hard constrained shortest path problems," *Discrete Applied Mathematics*, vol. 10, no. 2, pp. 125–137, 1985.
[39] J. L. Bander and C. C. White, "A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost," *Transportation Science*, vol. 36, no. 2, pp. 218–230, 2002.
[40] Y. Zheng, C. Xu, and J. Xue, "A simple greedy algorithm for a class of shuttle transportation problems," *Optimization Letters*, vol. 3, no. 4, pp. 491–497, 2009.
[41] Y.-J. Zheng, H.-F. Ling, S.-Y. Chen, and J.-Y. Xue, "A hybrid neuro-fuzzy network based on differential biogeography-based optimization for online population classification in earthquakes," *IEEE Trans. Fuzzy Syst.*, 2015, doi:10.1109/TFUZZ.2014.2337938.
[42] H.-U. Heiss and R. Wagner, "Adaptive load control in transaction processing systems," in *17th International Conference on Very Large Databases*, Barcelona, Spain, September 1991.
[43] D. Dyachuk and R. Deters, "Scheduling of composite web services," in *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, ser. Lecture Notes in Computer Science, R. Meersman, Z. Tari, and P. Herrero, Eds. Springer Berlin Heidelberg, 2006, vol. 4277, pp. 19–20.
[44] Y.-D. Lin, C.-M. Tien, S.-C. Tsao, R.-H. Feng, and Y.-C. Lai, "Multiple-resource request scheduling for differentiated QoS at website gateway," *Computer Communications*, vol. 31, no. 10, pp. 1993–2004, 2008.
[45] D. Garcia, J. Garcia, J. Entrialgo, M. Garcia, P. Valledor, R. Garcia, and A. Campos, "A QoS control mechanism to provide service differentiation and overload protection to internet scalable servers," *IEEE Transactions on Services Computing*, vol. 2, no. 1, pp. 3–16, Jan 2009.
[46] R. Khazankin, D. Schall, and S. Dustdar, "Adaptive request prioritization in dynamic service-oriented systems," in *IEEE International Conference on Services Computing*, July 2011, pp. 9–15.
[47] M. Pinedo, *Scheduling Theory, Algorithms, and Systems*, 2nd ed. Prentice Hall, 2002.
[48] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, 2008.
[49] S. K. Iyer and B. Saxena, "Improved genetic algorithm for the permutation flowshop scheduling problem," *Comput Oper Res*, vol. 31, no. 4, pp. 593–606, 2004.
[50] C.-J. Liao, C.-T. Tseng, and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Comput Oper Res*, vol. 34, no. 10, pp. 3099–3111, 2007.
[51] U. K. Chakraborty and K. P. Turvey, "Floating-point to integer mapping schemes in differential evolution for permutation flow shop scheduling," *International Journal of Bio-Inspired Computation*, vol. 2, no. 3, pp. 183–204, 2010.
[52] J. Yang, J. Chen, H. Liu, K. Zhang, W. Ren, and J. Zheng, "The Chinese national emergency medical rescue team response to the Sichuan Lushan earthquake," *Nat. Hazards*, vol. 69, no. 3, pp. 2263–2268, 2013.
[53] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 223–235, 2010.
[54] J.-S. Leu, C.-F. Chen, and K.-C. Hsu, "Improving heterogeneous SOA-based IoT message stability by shortest processing time scheduling," *IEEE Transactions on Services Computing*, vol. PrePrints, 2013.

Yu-Jun Zheng is an associate professor, Ph.D. supervisor in Zhejiang University of Technology, China. He received the Ph.D. degree from Institute of Software, Chinese Academy of Sciences in 2010. He is an IEEE member and an ACM member, and his research interests include bio-inspired computing and operations research. He has published over 60 scientific papers in international journals and conferences. In 2014, he received the runner-up of IFORS Prize for Development due to the work of emergency engineering rescue tasks scheduling in disaster relief operations in China.

Qing-Zhang Chen is a professor, Ph.D. supervisor in Zhejiang University of Technology, China. He received the B.S., M.S., and Ph.D. degrees from Hefei University of Technology in 1982, 1986, and 2007, respectively. He is a director of China Computer Education Association and a senior member of China Computer Federation. His research interests include computer networks and their applications, and he has published over 60 scientific papers.

Hai-Feng Ling is an associate professor in PLA University of Science & Technology, China. She received the Ph.D. degree from Nanjing University in 2011. Her research interests include operational management and decision support. She has received more than 20 awards of scientific and technological progress of PLA for her contributions in operational operations.

Jin-Yun Xue is a professor in Jiangxi Normal University, and a Ph.D. supervisor both at Institute of Software, Chinese Academy of Sciences and Wuhan University. He received his B.S. degree in Mathematics from Najing University in 1970. From 1985 to 1988, he worked as a visiting scholar at Cornell University. After June 1995, he spent 10 months as a visiting scholar at Santa Clara University. His research interests include software engineering and information systems.