IBM Research Report

Reconfiguration Aware Resource Allocation in Contingency Planning in Distributed Service Delivery

Vinayaka Pandit
IBM Research Division
India Research Lan
Bangalore - 560045. India.

Sreyash Kenkre
IBM Research Division
IBM India Research Lab
Bangalore - 560045. India.

Krishnasuri Narayanam
IBM Research Division
IBM India Research Lab
Bangalore - 560045. India.

Srinivas Karthik
IBM Research Division
IBM India Research Lab
Bangalore - 560045. India.

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

Abstract

Remote delivery of services using geographically distributed service delivery locations has emerged as a popular and viable business model. Examples of services delivered in this manner are software services, business process outsourcing services, customer support centers, etc. The very nature of services and the fragile nature of the business environments in some of the delivery locations accentuates the need for business continuity. A key aspect of enabling business continuity is, at the time of a disruptive event, ability to reroute the services delivered from affected locations to unaffected locations while meeting their resource requirements. Such rerouting is called recourse. We highlight the need for recourse aware resource allocation. We study this problem from a computational viewpoint, present a new recourse aware resource allocation heuristic, and experimentally compare this to traditional resource allocation methods.

1 Introduction

Business continuity is one of the most important aspects of remote service delivery. Essentially, this entails a commitment of a service provider to the service seeker that at least certain critical services will be delivered round the clock irrespective of the uncertainties in the remote operational environment. In this paper, we consider the basic problem of resource assignment when the service provider has an additional commitment of business continuity. We model business continuity measures as recourse actions. We motivate the need for "recourse aware" resource assignment by highlighting the deficiencies of traditional resource assignment heuristics in the presence of recourse actions. We present the computational complexity of the recourse aware resource assignment problem, novel heuristics to compute recourse aware resource assignments, and experimental results.

1.1 Business Continuity in Service Delivery

Recently, emerging economies like India, China, Brazil, etc. have emerged as popular destinations to deliver software services, back-office services, remote infrastructure management, and so on. Most of these are remote service deliveries in the sense that the services are delivered to customers spread throughout the world. A major portion of such service delivery is enabled by setting up of large-scale, geographically distributed infrastructures consisting of heterogenous resources. See [7] for more insight on the acceptance and feasibility of such a global delivery models from the point of view of service provider. It also presents an overview of the typical heterogenous, geographically distributed infrastructure network set up by such service providers.

One of the important aspects of services is that they cannot be stored and served when the demand arrives. A service has to be essentially served when it arrives. This, coupled with the concerns of the sourcing organization that the service delivery is happening from remote, uncertain environments makes a compelling case for a service provider to ensure highest levels of business continuity (in this case it means continuity of service delivery). See [8] for a client perspective of global sourcing.

Typical disruptions in the geographies that we mentioned are localized in nature. They could be any of the following: strikes, societal unrest, urban flooding, natural disasters, below par supply of utilities like power and water, etc. When a disruption happens at a location, the part of the organization's infrastructure located there become unavailable. Therefore, services that are being delivered from the affected locations have to be rerouted to unaffected locations. The rerouted location must be able to offer the required combination of resources for the service delivery to take place. Such a reroute action is called "recourse".

Importance of Recourse Aware Allocation: Traditionally, resource allocation in organizational setting is done without taking into account recourse requirements. The resource allocation method typically optimizes some notion of resource utilization or other business performance metrics. For instance, resource allocation may be treated as a "bin packing" problem and employ "best fit" policy [1, 2]. However, resource allocation under a disruption free environment could be very different from the optimal allocation when frequent disruptions have to be dealt with via recourse actions. Therefore, recourse aware allocation is important in situations where contingency mechanisms have to be invoked frequently.

Recourse Aware Resource Allocation Problem: The services required by each customer has to be allocated to a single location from where the required combination of resources will be deployed. But, the allocation of location to the customers should be such that, efficient recourse actions exist, at least for a pre-identified set of disruptive scenarios. We assume the following simplified set-up. At a given point in time, the service delivery organization knows the remaining capacity of different resources at different locations (after the allocations to the existing customers) and it also knows from the historical data the most common disruption patterns (in terms of affected locations). Now, the would like to batch the set of all new customers won over in the last quarter and allocate a base location for them (from where the services required by the customer will get their required combination of resources). It would like to do the allocation in such a way that the expected cost of rerouting at the time of disruptions is minimized.

Outline: We present an abstract formulation of the problem in Section 2 and present the computational complexity of the resource allocation problem in Section 3. In Section 4, we present an algorithmic approach to compute efficient resource allocation. In Section 5, we present experimental results.

2 Problem Formulation

In this section, we formulate the problem of computing a resource allocation that is amenable for efficient business continuity contingency planning.

In the context of a service delivery system, there are three important components to be modeled. They are *resource infrastructure network*, *service accounts*, and *scenarios*. The resource infrastructure network is used to model the set of all the resources that a service delivery organization uses to deliver its services. The service accounts represent the different services being delivered to different customers. Essentially, service accounts represent the customer accounts in the business world and model the

resource requirements of the customer accounts. The scenarios are used to model the different possible disruptions that may occur to the resource infrastructure network.

Formally, the resource infrastructure network comprises of resources belonging to a finite set of resource types $\mathcal{T} = \{T_1, T_2, \ldots, T_r\}$. We use t as an index to the resource types. The resources in the organization are distributed geographically over a set of locations $\mathcal{L} = \{L_1, L_2, \ldots, L_m\}$. We use i as an index to the locations. Associated with each location L_i is its capacity profile given by $(c_{i,1}, c_{i,2}, \ldots, c_{i,r})$ where $c_{i,t}$ denotes the capacity of the resource type T_t available at location L_i. Furthermore, for each pair $L_{i1}, L_{i2} \in \mathcal{L}$, we are also given $d_{i1,i2}$, the distance between L_{i1} and L_{i2}. In our model, we assume that the cost of movement between L_i and L_j are same in both the directions. However, these assumptions can easily be relaxed.

The service accounts in the system are given by the set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. We use h as an index to the service accounts. Each service account J_h is specified by its resource requirement profile: $((u_{h,1}, l_{h,1}), (u_{h,2}, l_{h,2}), \ldots, (u_{h,r}, l_{h,r}))$. $u_{h,t}$ is the "normal requirement" and means that J_h requires $u_{h,t}$ units of the resource type T_t during normal operations. $l_{h,t}$ is the "critical requirement" and means that J_h requires $l_{h,t}$ units of resource type T_t to ensure continuity of service delivery; furthermore $l_{h,t} \leq u_{h,t}$. Associated with each service account J_h is o_h, its *overhead factor*. This overhead factor captures the overhead involved in starting the service delivery from the alternate location when it is rerouted. In the context of the software service delivery example, it may include overheads like getting network ports and access controls enabled, arranging for secure access to the seats, transferring project servers (if need be), etc. In the rest of the paper, we use the terms service accounts and jobs interchangeably.

We now formalize the notion of resource assignment. Since each service account is a project, it needs to obtain all its resources from the same location. Therefore, an assignment has to map each service account to a location from where it obtains all its requirements. Formally, an assignment is a mapping of service accounts to the locations, i.e, $A : \mathcal{J} \rightarrow \mathcal{L}$. Let $A^{-1}(L_i)$ denote the set of service accounts assigned to location L_i, i.e, $A^{-1}(L_i) = \{J_h \in \mathcal{J} \,|A(J_h) = L_i\}$. A valid assignment is one in which the capacities of the resource types at each location are not violated. Formally, $\forall L_i \in \mathcal{L}$ and $\forall\, T_t \in \mathcal{T}$, $\sum_{J_h \in A^{-1}(L_i)} u_{h,t} \leq c_{i,t}$.

For business continuity planning, a key input is the set of *scenarios* which model different disruptions that could happen. Formally, the set of scenarios is given by $\mathcal{S} = \{S_1, S_2, \ldots, S_p\}$. We use k as an index to the scenarios. Each scenario S_k is a subset of $\mathcal{L}$. The meaning of S_k is that the resources located at the locations in S_k are not available. Therefore, for a given scenario S_k, the set of service accounts that need to be rerouted is given by $A^{-1}(S_k) = \cup_{L_i \in S_k} A^{-1}(L_i)$. They need to be rerouted to one of the locations in $\mathcal{L} \setminus S_k$. When a job J_h is rerouted to location L_i it means that a resource profile of $(l_{h,1}, l_{h,2}, \ldots, l_{h,r})$ is allocated to J_h at location L_i.

Formally, we need an assignment A_{S_k}, such that, $\forall J_h \in A^{-1}(S_k)$, $A_{S_k}(J_h) \in \mathcal{L} \setminus S_k$ and $\forall J_h \in \mathcal{J} \setminus A^{-1}(S_k)$, $A_{S_k}(J_h) = A(J_h)$. The new assignment A_{S_k} should be a valid assignment, i.e, for all pairs $T_t \in \mathcal{T}$ and $(L_i \in \mathcal{L} \setminus S_k)$ it should be true that $(\sum_{J_h \in (A_{S_k}^{-1}(L_i) \cap A^{-1}(S_k))} l_{h,t} + \sum_{J_h \in A_{S_k}^{-1} \setminus A^{-1}(S_k)} u_{h,t}) \leq c_{i,t}$. The cost of A_{S_k} given by $Cost(A_{S_k}) = \sum_{J_h \in A^{-1}(S_k)} o_h \cdot d_{A(J_h), A_{S_k}(J_h)}$ where o_h is the overhead factor of

a rerouted job J_h. Note that the quantity $d_{A(J_h),A_{S_k}(J_h)}$ captures the distance between the original and rerouted locations of the service account J_h. Our cost function models the cost of enabling the service delivery from the rerouted locations. It makes a simplifying assumption that it is the product of the job's overhead factor and the distance of rerouting. In presence of even more specific information of the costs involved, the cost function can be modified to take such information into account. In planning literature, such reallocations are called as "recourse".

We now define the *recourse aware resource allocation* problem referred to as the RECONNECT problem. Input consists of a set of resource types $\mathcal{T}$, a set of locations $\mathcal{L}$, capacity profiles for the locations $\forall L_i \in \mathcal{L}$, set of service accounts $\mathcal{J}$, set of resource profiles and overhead factors for the service accounts $J_h \in \mathcal{J}$, and a set of scenarios $\mathcal{S}$. Goal is to compute a valid assignment A for normal operations such that $\sum_{S_k \in \mathcal{S}} Cost(A_{S_k})$ is minimized. Here, for a scenario S_k, A_{S_k} denotes the reassignment corresponding to the scenario S_k.

3 Complexity of the RECONNECT Problem

One may wish to design an efficient and optimal algorithm for the RECONNECT problem. But, we show that such an algorithm is unlikely to exist. In particular, we establish the NP-Completeness and the hardness of approximation of the RECONNECT problem via a reduction from the well known dominating set problem [5].

The Dominating Set Problem: The input consists of a graph $G = (V, E)$ and an integer g where V is the set of nodes and E is the set of edges incident on V. The decision problem is to output whether or not there exists a dominating set in G of size g. A subset of vertices $D \subseteq V$ is said to be a dominating set if, for any vertex $v \in V$, either $v \in D$ or v is adjacent to a vertex in D. The dominating set is NP-hard to approximate within a ratio of $\Omega(\log |V|)$ [9]. Given an instance of the dominating set problem, we give a polynomial time reduction to the RECONNECT problem.

Our reduction is such that the resulting RECONNECT instance has just one resource type T_1. Therefore, the capacity profile of a location will just be a number that indicates its capacity of the resource type T_1. Further, each service unit will be a tuple (u, l) denoting its normal and critical requirement of the resource type T_1.

Reduction: Let $G = (V, E)$ and g be the input to the dominating set problem. There is one location for each node in the graph, i.e, $\mathcal{L} = \{L_v | v \in V\}$. For each edge $e = (u, v)$, we set $d_{u,v} = 1$. For other pairs which do not have an edge between them $d(u, v)$ is set to be the length of the shortest path connecting u and v; note that $d(u, v) \geq 2$ for such pairs. The capacities at $L_v, v \in V$ is 1. We create $(n-g)$ jobs with the service profile $(1, 1)$. The overhead factor for each job is 1. Our set of scenarios is: $\mathcal{S} = \{S_1, S_2, \ldots, S_{|V|}\}$ where scenario S_k corresponds to the location L_k (location corresponding to node k in V) being unavailable. This completes our construction. It is easy to see that this can all be done in linear time and therefore, our reduction runs in polynomial time.

Lemma 1. *There is a dominating set of size g in $G = (V, E)$ if and only if there is an initial allocation A to the* RECONNECT *problem of cost exactly $n - g$.*

Proof. Suppose there is a dominating set of size g, say D. Consider an allocation which assigns exactly one job to each of the locations in $V - D$ and no job to any of the locations in D. Note that the locations in D have spare capacity of 1. Therefore, for each of the scenarios corresponding to $(V - D)$, we can reroute the service account assigned to the location to one of the locations in the dominating set. This is always possible by the definition of the dominating set. Cost of such reassignments is 1 and over $(n - g)$ locations, total cost is $(n - g)$. For the scenarios corresponding to the locations in the dominating set, there is no need to reroute as no service accounts are assigned to them. Thus, the reassignment cost across all the scenarios is $(n - g)$.

Similarly, we show that if there is an assignment of cost exactly $(n-g)$, then, there is a dominating set of size g. Observe that at most one job can be assigned to each location. Therefore, in any allocation, there should be $(n - g)$ locations with one job each and g locations which are not allocated any job. Let B be the set of locations with no job. For each of the scenario in $(V - B)$, the cost of rerouting is at least 1. Furthermore, for a location $v \in (V - B)$ which does not have a neighbor in B, the rerouting cost is at least 2. Therefore, unless B is a dominating set, cost of the reallocations across all scenarios in $V - B$ has to be at least $(n - g) + 1$. □

In our reduction, the distances satisfy triangle inequality. Furthermore, it is easy to see that our reduction is approximation preserving (refer to [3]). So, the RECONNECT problem is at least as hard as the dominating set problem. We have:

Theorem 1. *The* RECONNECT *problem is NP-Complete even when the distances satisfy triangle inequality (i.e, form a metric) and unless $P = NP$, it is NP-hard to approximate within a ratio of $\Omega(\log n)$ where n is the number of locations.*

4 Algorithmic Approach

As proved in 3, the RECONNECT problem is a difficult problem from the point of view of computing optimal recourse aware resource allocation. In this section, we develop an algorithmic approach to solve the problem in practice.

4.1 Relative Importance of Resource Types

Note that, since the system requires the rerouting to take place in case of each scenario, we assume that there is sufficient residual capacity even after allocating all the jobs to the locations. For each resource type T_t, let C_t represent the total capacity of resource type t across all locations, i.e, $C_t = \sum_{L_i \in \mathcal{L}} c_{i,t}$. Similarly, let R_t represent the total demand for the resource type t across all the jobs, i.e, $R_t = \sum_{J_h \in \mathcal{J}} u_{h,t}$. The residual capacity for t is given by $B_t = C_t - R_t$. We assume $B_t > 0$. The ratio $r(t) = \frac{B_t}{C_t}$ represents how tight the supply of the resource type T_t is. Lower the ratio, more scarce is the resource type. The weightage of a resource type t is given by

$$w(t) = \frac{r(t)}{\sum_{t1 \in \mathcal{T}} r(t1)} \tag{1}$$

We now compute an ordering on the jobs which represents the overall scarcity of its resource requirements. The weightage of a service account J_h is given by

$$W(h) = \sum_{t \in \mathcal{T}} u_{h,t} r(t) \tag{2}$$

4.2 Location Allocation Heuristic

The set of jobs in $\mathcal{J}$ are considered in the decreasing order of their weights $W(J_h)$. Our heuristic for assigning a location to a job J_h is the following: assign it to a location L_i such that, in the scenarios that contain L_i, the average cost of rerouting J_h from L_i is minimum. When we consider the jobs in the decreasing order of their weightages, we say that J_h suffers *contingency deficiency* if the following condition holds (given the allocation for the previous jobs in the order): for every choice of location for J_h there is some scenario $S_k \in \mathcal{S}$ under which there is no viable rerouting option. In other words, no matter which location it is allocated to, it cannot be rerouted under some scenario. If that happens, a given choice is evaluated by adding a large penalty for every scenario under which rerouting is not possible. The details of the complete algorithm are presented in Algorithm 1.

5 Experimental Results

In this section, we present experimental results that study the potential benefits of recourse aware resource allocation in comparison to traditional methods.

5.1 Simulation Engine

We have built a simulation engine that can simulate distributed service delivery organizations. Most remote service delivery organizations can be conceptualized *hierarchically*. For example, an organization can be thought as distributed in multiple cities, each city consists of multiple campuses, each campus consists of multiple buildings, each building has multiple floors, and finally each floor consists of office spaces. So, the distance between various points have a hierarchical nature. Our simulation engine can generate such organizational structure. But, for simplicity, we present results with a *flat* organization in which there is only one layer of geographical locations. This is a convenient abstraction if we treat all the distances below a hierarchical level, say a campus, as equal to zero. In this case, the flat representation just considers a distance metic over all the campuses. In most settings, the flat representation is good enough to the hierarchical one.

One naive way of implementing a simulator of service delivery would be to independently and randomly generate each of the three major components: infrastructure, service accounts, and scenarios. But, the resulting data would be quite meaningless as the service accounts that an organization decides to serve typically depends on its infrastructure. Similarly, whether a scenario is of interest or not depends on what impact it has on the overall infrastructure network. Therefore, our simulator is designed to reflect the correlations between various components as briefly described below.

input : Resource types $\mathcal{T}$, Locations $\mathcal{L}$, Capacity profiles for the locations, distances $d_{i,j}$s between the locations, Service accounts $\mathcal{J}$, Resource profiles of the service accounts, and Scenarios $\mathcal{S}$
output: A recourse aware resource allocation $A : \mathcal{J} \rightarrow \mathcal{L}$

```
1  ∀T_t ∈ 𝒯, compute C_t, B_t, R_t, and r(t) as described in Section 4.1;
2  ∀T_t ∈ 𝒯, compute w(t) as shown in Equation 1;
3  ∀J_h ∈ 𝒥, compute W(h) as shown in Equation 2;
4  Order the jobs in the decreasing order of their weightages. After reordering, J_1
   corresponds to the job with highest weightage and so on;
5  for h ← 1 to | 𝒥 | do
6      curChoice = NULL; curCost = ∞ ;
7      for i ← 1 to | ℒ | do
8          if (It is feasible to allocate J_h to L_i) then
9              cost = 0;
10             for every scenario S_k such that L_i ∈ S_k do
11                 For all J_h is allocated to L_i, is it possible to reroute J_h under
                   S_k? ;
12                 If Yes, cost += Cost of Optimal Rerouting of J_h under S_k;
13                 Else, cost += Z where Z is a large penalty for contingency
                   deficiency;
14             end
15             if (cost <curCost) then
16                 curCost = cost; curChoice = L_i;
17             end
18         end
19     end
20     Allocate J_h to curChoice;
21 end
```

Algorithm 1: The main algorithm

We observe that there are a few resource types (example: Email, WAN, Power Systems), referred to as *common type* which are required by almost all the service accounts. Then, there are resource types, referred to as *special type* which are required by only a few service accounts (example: LANs with limited access and security features, secured seats, etc.). At each location, the simulator generates instances of all the common type resources and sets high capacity for them. As for the special type resources, it only generates a subset of them and sets relatively lower capacity. Moreover, not all the resource types of the organization are required by all service accounts (example: purely call handling account may not need printers, copiers, etc.). Further, it is important to note that the service accounts taken on by the organization is highly correlated with its infrastructure network. So, the simulator generates the resource requirements of the service accounts as follows. Each service account has an associated inherent size. It picks small subsets of both common and special type resources. Its requirement for the common type resources is set proportional to its inherent size. For the special type

resources, its requirement is set as per a normal distribution whose mean is determined by its inherent size. We have verified that the profile of the infrastructure networks and the service accounts generated this way are quite similar to real-life data (which are highly confidential and no company can make it available for public dissemination).

5.2 Scenario Generation

We construct both rule based scenarios and scenarios which are constructed by carefully analyzing the infrastructural network for bottlenecks. Example of rule based scenarios is one where $\mathcal{S}_1 = \{\{L_1\}, \{L_2\}, \ldots, \{L_m\}\}$, i.e, set of all possible scenarios in which exactly one location is not available. In real-life, rule based scenarios help an organization to test its preparedness for contingency. We consider two rule based scenario sets $\mathcal{S}_1$, $\mathcal{S}_2$ where $\mathcal{S}_1$ is defined as above and $\mathcal{S}_2$ consists of all possible scenarios in which exactly two locations are not available. Apart from the rule based scenarios one can also construct scenarios that are specific to the infrastructural network. Specifically, we construct *moderate* and *difficult* scenarios based on their impact on the residual capacities of the resource types in the network. The moderate scenarios are created as follows: starting from the entire network, keep marking a random location as unavailable till the residual capacity of any one of the resource types falls below 80% of its original capacity. We generate many such scenarios. The difficult scenarios are created similarly except that the threshold for completing a scenario is 60% instead of 80%.

5.3 Geographical Distribution of the Locations

One way of generating the locations on a map would be to just locate them at random locations on a grid. But, it is easy to see that such an approach does not stress test the allocation heuristics for the following reasons: with respect to a location, number of locations at different distances is fairly well spread out. Therefore, likelihood of ever rerouting jobs from a location to its farthest point is very small. But, a re-look at Figure **??** highlights the fact that there are two clusters which are separated by large distance. The best fit allocation fails because, in some scenarios it has to reroute across the clusters. Therefore, we generate the locations as follows (it also mimics real-life cases pretty well): we consider a small number of clusters which are separated by a large distance; we generate locations only within these clusters. This forces a job to pay a high cost every time it has to be rerouted to a location outside its clusters.

5.4 Experimental Comparisons

We have used our simulation engine to construct service delivery organizations of different sizes to conduct our experiments. Simple and very small organizations with just a handful of locations and jobs were analyzed manually to ensure the soundness of the approach. We then generated medium sized and large sized organizations for our experimental study. The medium sized organizations consisted of roughly a dozen locations and up to 250 jobs. The large sized organizations consisted of two or three dozen locations and in the range of 500 jobs.

We have implemented the following resource allocation heuristics: recourse aware allocation heuristic presented in Section 4, first-fit algorithm which assigns resources by following the first-fit bin-packing heuristic, similarly the best-fit heuristic for bin-packing, and finally just a random allocation strategy of allocating a random location with required capacity of resource types.

We generated large number of instances of medium and large sized organizations on which all the four allocation procedures were run. Due to lack of space, we will only present a summary of the experimental comparison of the four methods. What we have captured in these summaries is representative of the results observed across all the experiments. We consider two parameters for comparison: maximum number of jobs that need to be rerouted under one of the scenarios and the actual average cost of rerouting under all the scenarios of interest. We report the results split across the different rule based scenarios, moderate scenarios, and difficult scenarios.

Table in Figure 1 shows the comparison of the different methods with respect to the number of jobs that need to be rerouted. Here, "RA" refers to the recourse aware allocation heuristic, "FF" refers to the first-fit heuristic, "BF" refers to the best-fit heuristic, and "Random" refers to the random allocation method. In the BF heuristic, a job is allocated to a location which minimizes the fragmentation, i.e, a location at which the remaining capacity after the allocation is minimized. In the FF heuristic, locations are ordered based when a job was first assigned to them. The FF heuristic allocates a job to the earliest location in the order where it can fit in. Under the "Scenario Type" column, "Rule: x loc" refer to the rule based scenarios which include all scenarios in which exactly x location(s) are not available. Similarly, Table in Figure 2 shows the comparison with respect to the actual cost of rerouting. In both tables, we normalize the entries with respect to RA. For example, if an entry in a cell in Figure 2 has an entry f, it means its average reroute cost was f times the average reroute cost of the RA strategy (across all the different experiments).

From the table in Figure 1, we see that the average number of jobs that need to be rerouted for the bin-packing heuristics could be smaller than the corresponding number for the recourse aware allocation. But, the crucial observation is to look at the cost of rerouting in Figure 2. Note that the recourse aware allocation consistently has lower cost than all the other strategies. Even with smaller number of average jobs to reroute, the BF and FF strategies could take take up to a factor of 1.6 times more than the cost of RA. Clearly, the RA strategy performs very well in terms of the cost of rerouting. But, the curious case is that of the Random strategy. Its performance is surprisingly better than the bin-packing strategies. Preliminary observation suggests that the bin-packing algorithms do a good job in terms of the minimizing the number of jobs that need to be rerouted. But, they suffer from lack of efficient reroute options in some of the scenarios and end up paying higher cost on an average.

6 Conclusions

In this paper, we argued the importance of recourse aware allocation of service delivery organizations. We presented an abstract formulation of the problem and studied its complexity. We presented a novel recourse aware resource allocation heuristic. We

Scenario Type	RA	FF	BF	Random
Rule: 1 loc	1.0	1.0	1.0	1.0
Rule: 2 loc	1.0	1.0	1.0	1.0
Moderate	1.0	0.9	0.9	1.05
Difficult	1.0	0.93	0.93	1.01

Figure 1: Comparison w.r.t. the number of jobs to be rerouted

Scenario Type	RA	FF	BF	Random
Rule: 1 loc	1.0	1.7	1.6	1.4
Rule: 2 loc	1.0	1.3	1.29	1.33
Moderate	1.0	1.58	1.64	1.08
Difficult	1.0	1.12	1.14	1.16

Figure 2: Comparison w.r.t. the average cost of rerouting

presented initial simulation based results that demonstrate the potential benefits of recourse aware resource allocation. We leave open the problem of developing provably near-optimal algorithm for the `RECONNECT` problem. Another interesting direction is to develop challenging real-life benchmark datasets for service operations that reflect the typical contingency scenarios faced in geographies like India, China, Brazil, Russia, etc.

References

[1] G. Antoniol, M. Penta, and M. Harman. Search-based techniques for optimizing software project resource allocation. In *Proceedings of the Genetic and Evoluationary Computation (GECCO)*, pages 1425–1426, 2004.

[2] E. Coffman, M. Garey, and D. Johnson. *Approximation algorithms for bin packing: a survey*, pages 46–93. PWS Publishing Co., 1997.

[3] P. Crescenzi. A short guide to approximation preserving reductions. In *IEEE Conference on Computational Complexity*, pages 262–273, 1997.

[4] B. Dietrich and T. Harrison. Serving the services business. *Operations Research and Management Science*, 2006.

[5] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[6] J. Graham and D. Kaye. *A Risk Management Approach to Business Continuity*. Rothstein Associates Inc., 2006.

[7] S. Jalona and A. Chandrakar. Evolution of it services delivery model. Infosys White Paper available at http://www.infosys.com/global-sourcing/white-papers/pages/index.aspx, 2008.

[8] Keane White Paper. Going global with application outsourcing. Report is available at http://www.keane.com/resources/pdf/WhitePapers/WP-GGAO.pdf, 2011.

[9] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *ACM Symposium on Theory of Computing (STOC)*, pages 475–484, 1997.

[10] Wipro Report. Wipro Business Continuity - "Plan B". Report is available http://www.wipro.com/documents/Wipro_Business_Continuity.pdf.