Modeling Multi-factor Multi-site Risk-based Offloading for Mobile Cloud Computing

Huijun Wu, Dijiang Huang
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Email: {Huijun.Wu, Dijiang.Huang}@asu.edu

***Abstract*—Offloading decisions for computation-intensive applications in mobile cloud computing may involve many decision factors. Important decision factors such as offloading node reliability and privacy protection have not been well studied. Moreover, existing offloading models mainly focus on the one-to-one offloading relation. To address the multi-factor and multi-site offloading mobile cloud application scenarios, we present a multi-factor multi-site risk-based offloading model that abstracts the offloading impact factors as for offloading benefit and offloading risk. The offloading decision is made based on a comprehensive offloading risk evaluation. This presented model is generic and extendible. Four offloading impact factors are presented to show the construction and operation of the presented offloading model, which can be easily extended to incorporate more factors to make offloading decision more comprehensive. The overall offloading benefits and risks are aggregated based on the mobile cloud users' preference. The performance evaluation presents the practicality of the presented solution.**

I. Introduction

In mobile cloud computing, offloading is an important approach to overcome the resources and functionalities constrains of mobile devices. In a mobile cloud offloading model, applications deploy their components on multiple application processing nodes such as mobile smart phones and virtual machines in a cloud. A mobile device can rely on an offloading decision model based on multiple factors such as offloading computational-intensive, time- or energy-consuming application functions.

Many existing mobile cloud offloading models focus on a one-to-one directional offloading model (e.g., [1][2][3]): i.e., offloading from a mobile device to a cloud server (e.g., a VM). Study in [4] had shown the benefits of using one-to-many (i.e., multi-site) offloading models to improve the performance of mobile devices, where a mobile device can offload multiple application functions to multiple computing nodes. However, existing offloading models only consider one offloading factor such as computation overhead, networking delay, or energy consumption. Moreover, they do not consider other important offloading factors such privacy, reliability, etc.

One of important focuses of this work is to model the offloading risks that have not been addressed by previous research. Particularly, we focus on two main sources of risks in mobile cloud computing: privacy breach and offloading reliability. Mobile application components can be offloaded to malicious nodes that can potentially compromise the application's privacy, e.g., the offloaded data and functions may contain private data or expose the purpose of the application. In addition to the privacy-breach risk, another offloading risk is the reliability of offloading targets. For example, the surrogates may be unavailable due to unstable communication link or software crash, etc. In this case, offloading effort has to include offloading recovery strategies. Thus, both privacy breach and reliability risks should be considered in the offloading model in addition to the performance offloading factors described previously.

The presented multi-factor multi-site risk-based offloading model abstracts the offloading impact factors into two categories: offloading benefit and offloading risk. The final offloading decision is made based on the aggregated and normalized benefit and risk evaluations. This presented model is generic and extendible in that the number of factors that are considered in the offloading decision processes varies. Specifically, we present four offloading impact factors to show how the model works, and it can cover more factors to make offloading decision more comprehensive. The offloading benefits include delay reduction and energy saving, and offloading risks factors include privacy breach and reliability of the offloading targeting nodes. The overall offloading benefits and risks are aggregated based on the mobile cloud users' preference. Finally, we design an ant-based algorithm to compute the optimal application partition strategy. In summary, the contributions of presented research are highlighted as follows:

- *Risk-based multi-factor decision*: The proposed approach takes two risk factors into offloading decision making process. The privacy risk and reliability risks have not been studied in previous work, however they are common issues in mobile cloud computing application scenarios. The offloading decision is made by comparing the aggregated risk and potential offloading benefit. The proposed decision approach is not limited to only two types of risk or benefits. The solution can be easily extended to many factors as long as the factors result in either benefits or risks.
- *Multi-site offloading*: The proposed offloading approach picks one or multiple surrogates from a set of candidate sites (here we use *node* and *site* interchangeably), i.e., there could be multiple offloading destinations, where both cloud VMs and mobile nodes can serve as a destination.

The remainder of this paper is organized as follows: Section II discusses recent related work. Section III presents models of mobile cloud applications, and formulates the offloading partition problem, and presents the ant system based algorithm.

Section IV provides some experiment results illustrating the performance of our proposed algorithm as well as algorithm tuning guides. The conclusions are given in section V finally.

II. Related Work

Cuckoo [5] proposed a static offloading framework that is build on Android and requires a separate compile process to generate mobile device and cloud versions of the same application. This framework does not have offloading decision intelligence. Zhang *et al.* proposed a web service based elastic offloading framework [3]. This framework uses a cost model to make offloading decision. The author did not provide an effective method to obtain optimal offloading decision. ThinkAir [2] and MAUI [1] are energy saving frameworks. They profiles the hardware components and make offloading to optimize energy usage of mobile device. These two frameworks only consider the energy issue of offloading and ignore other aspects. Clonecloud [6] and eXCloud [7] are virtual machine related frameworks. Clonecloud mirrors the mobile device application in the cloud. eXCloud migrates Java virtual machine stack to cloud. These two frameworks record the application state on mobile devices and resume the application from the stored state in cloud.

Besides offloading framework, some research work focus on offloading algorithms. Giurgiu *et al.* [8] models application as a dependency graph. 'All step' and 'K step' algorithms were proposed to solve application partition problem. Memory and transferred data size were taken into the consideration. Ou *et al.* [9] discussed system failure in offloading systems. They analyzed system execution time considering system failure and recovery. They did not consider how to make offloading decisions. Wolski *et al.* [10] considered offloading decision problem based on network bandwidth constraints. They assumed that the network reliability is not an issue. However, the network may even not be connected in real mobile cloud computing scenario due to mobility or other reasons. Sinha *et al.* [4] proposed offloading scenarios involving multiple sites. However, their approaches do not consider the privacy and reliability issues during the mobile cloud offloading process.

Although the above researches provided different offloading frameworks and decision models, they mostly focused on a one-to-one mobile devices to cloud servers offloading model, and none of existing solutions consider the privacy and reliability issues during the offloading procedure.

III. System and Models

In our framework, a mobile application is programmed based on component-based design, where components provide functionality via interfaces and may, conversely, consume functionality provided by other components via their interfaces. The application can be presented as a graph $G_{app} = (C, E)$ where a vertex is a component and an edge represents the interaction between components. Let $m = |C|$. In a multi-site offloading scenario, some computation workload is offloaded from an original mobile device and distributed onto candidate surrogate sites. The original site and its candidate surrogate sites form an egocentric network $G_{sur} = (S, L)$, where a node is a site and a link represents a network connection between two sites. Let $n = |S|$.

The multi-site offloading problem can be modeled to find a mapping from the application graph G_{app} to the candidate network G_{sur} to achieve a given optimization objective function. Let matrix X present this mapping, whose element x_{ij} is set to either 1 when component i is assigned to site j or 0 otherwise. Based on X, four more mappings can be defined. Mapping $f_{C\to S}$ maps component i to site j, and $f_{S\to C}$ is the reverse mapping. Mapping $f_{E\to L}$ maps edge $e = (i_1, i_2)$ to link $l = (f_{C\to S}(i_1), f_{C\to S}(i_2))$; and $f_{L\to E}$ maps a link to a vector of edges. These four mappings can easily be expended to accept vector as well.

A. Offloading Benefits

1) Execution Time: The component c is labeled with computation load w_c and vector $\mathbf{w}$ is computation load of all vertexes; the edge $e = (c_1, c_2)$ from c_1 to c_2 is labeled with data transfer load $f_{c_1c_2}$ and matrix $F_{m\times m}$ is data exchange load of all edges. Meanwhile, candidate site s is labeled as a vertex weight u_s and vector $\mathbf{u}$ is weights of all sites; link $l = (s_1, s_2)$ from s_1 to s_2 is labeled as link weight $d_{s_1s_2}$ and matrix $D_{n\times n}$ is weights of all links.

In this article, we define the operator $.*$ as array inner multiplication that multiplies arrays or matrices in element-by-element way, which is different from matrix multiplication. Let $\tilde{\mathbf{u}}$ be the vector that satisfies $\mathbf{u}.*\tilde{\mathbf{u}} = \mathbf{1}$ where $\mathbf{1}$ is the vector whose elements are all 1's. Then the upper bound of total time spent for computation load is sum of workload on every site over its processing capability:

$$t_c = \mathbf{w}^T X\tilde{\mathbf{u}}. \tag{1}$$

Similarly, let $\tilde{D}_{n\times n}$ be the matrix that satisfies $D.*\tilde{D} = \mathbf{1}_{n\times n}$ where $\mathbf{1}_{n\times n}$ is the matrix whose elements are all 1s. The transformation X^TFX redistributes the communication load of $F_{m\times m}$ into a $n\times n$ matrix where element positions are corresponding to $\tilde{D}_{n\times n}$. The upper bound of total time spent on networks for data exchange load is the sum of workload on every link over its throughput:

$$t_n = tr(X^TFX\tilde{D}^T), \tag{2}$$

where the $tr()$ function calculates matrix trace $tr(A_{n\times n}) = \sum_{i=1}^{n} a_{ii}$. So the upper bound of total time is the sum of the computation time and the communication time: $t = t_c + t_n$.

2) Energy Consumption: Energy consumption on mobile devices can be categorized according to hardware modules. The major categories are CPU, radio module including Wi-Fi and Cellular, display, audio device, GPS module and vibration motor [11][12][13][14]. Both CPU and radio power can be modeled as a linear model that consists of two parts: dynamic consumption when hardware module is active and base consumption [15][16]. The dynamic part of CPU power is proportional to utilized processing capability according to [14][13][17][11]:

$$P^{CPU} = \mathbf{1}^T X(\mathbf{u}.*\mathbf{c}^{CPU}), \tag{3}$$

where $\mathbf{c}^{CPU}$ is the coefficient vector for all sites. Let's code E as $\mathbf{e}$, then the dynamic part of radio module power is proportional to the outgoing packet rate [14][13]:

$$D' = D.*(\mathbf{1}_{n\times n} - I).*(\mathbf{c}^{radio}\mathbf{1}^T), \tag{4}$$

$$P^{radio} = \sum_{\forall l \in f_{E\to L}(\mathbf{e})} d'_l, \tag{5}$$

where $\mathbf{c}^{radio}$ is coefficient vector for all sites, I is identity matrix and d'_l is the element of D' corresponding to link l. Let P^{CPU}_{idle} be the static part of CPU power and P^{radio}_{base} be static power of radio module. Then the total power is the sum of the above four parts: $P = P^{CPU} + P^{CPU}_{idle} + P^{radio} + P^{radio}_{base}$.

B. Offloading Risks

1) Privacy and Trust: Information leakage may happen in transportation process in network and/or in computation process on sites. Let's code C as $\mathbf{c}$ and E as $\mathbf{e}$, then combine $f_{C\to S}(\mathbf{c})$ and $f_{E\to L}(\mathbf{e})$ as a vector $\mathbf{v}$ of elements that may leak information. Corresponding to this vector, two state vectors $\mathbf{s}$ and $\mathbf{s}'$ are defined of one unique surrogate network state. Vector $\mathbf{s}$ records what elements leak information and vector $\mathbf{s}'$ records links on which data is exposed. Both $\mathbf{s}$ and $\mathbf{s}'$ are constructed initially as all 0's and then adjusted according to:

- For $\mathbf{s}$, if v_i leaks information, then $s_i = 1$.
- For $\mathbf{s}'$, two situations are considered. First, if a link $v_i \in f_{E\to L}(\mathbf{e})$ leaks information, then $s'_i = 1$. Second, if a site is compromised, then all data transferred on incident links are exposed. Let $v_i \in f_{E\to L}(\mathbf{e})$ and $v_i = (v_j, v_k)$, and either v_j or v_k leaks information, then $s'_i = 1$.

Let function $p_1(v_i)$ be the probability of leakage occurring for element v_i. The probability of state $\mathbf{s}$ is the probability product of all independent leakage events:

$$p_{1\mathbf{s}} = \prod_{\forall s_i=1} p_1(v_i) \prod_{\forall s_i=0} (1 - p_1(v_i)). \tag{6}$$

The amount of exposed data is considered as the information leakage consequence. The impact of state $\mathbf{s}'$ is data sum of all unique leakage events:

$$q_{1\mathbf{s}'} = \sum_{\forall s'_i=1} \{f_{\tilde{\mathbf{e}}}\}, \tag{7}$$

where $f_{\tilde{\mathbf{e}}}$ are values picked from F corresponding to edges in $\tilde{\mathbf{e}} = f_{L\to E}(v_i)$; and $\{f_{\tilde{\mathbf{e}}}\}$ is set of data loss in all unique leakage events.

All valid $\mathbf{s}(\mathbf{s}')$ vectors form a state space H and $\sum_{\mathbf{s}\in H} p_{1\mathbf{s}} = 1$. The risk can be computed based on the expected data loss:

$$r_1 = \sum_{\mathbf{s}(\mathbf{s}')\in H} p_{1\mathbf{s}} q_{1\mathbf{s}'}. \tag{8}$$

2) Reliability: The surrogates may be unavailable due to device mobility or failures in network or on sites. Similarly, vector $\mathbf{v}$ is defined of elements that may be unavailable and state vectors $\mathbf{s}$ and $\mathbf{s}'$ are defined as well. Vector $\mathbf{s}$ records what elements are unavailable and vector $\mathbf{s}'$ records nodes on which workload is lost. Analogically to the section III-B1, we can calculate the risk r_2 introduced by surrogates' unavailability:

$$p_{2\mathbf{s}} = \prod_{\forall s_i=1} p_2(v_i) \prod_{\forall s_i=0} (1 - p_2(v_i)), \tag{9}$$

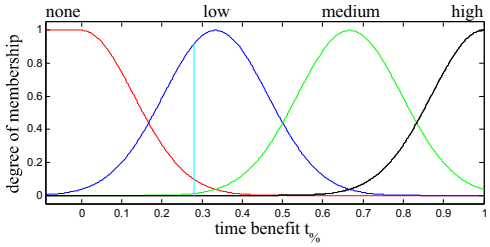

Fig. 1. Time benefit member functions.

TABLE I. BENEFIT INFERENCE RULES

Weight	Time benefit	Energy benefit	Final benefit
...	...	...	...
0.85	low	medium	medium
...	...	...	...

$$q_{2\mathbf{s}'} = \sum_{\forall s'_i=1} \{w_{\tilde{\mathbf{c}}}\}, \tag{10}$$

$$r_2 = \sum_{\mathbf{s}(\mathbf{s}')\in H} p_{2\mathbf{s}} q_{2\mathbf{s}'}, \tag{11}$$

where $p_2(v_i)$ be probability of unavailability occurring for element v_i; $w_{\tilde{\mathbf{c}}}$ are values picked from $\mathbf{w}$ corresponding to components in $\tilde{\mathbf{c}} = f_{S\to C}(v_i)$, and $\{w_{\tilde{\mathbf{c}}}\}$ is a set of computation loss of all unavailability events.

C. User Preference

We classify benefits from each source into several levels, such as *none*, *low*, *medium*, and *high*, so that each type of benefits can be processed together. To obtain the overall offloading benefit, user preference is used to aggregate time benefit and energy benefit. We use a fuzzy-based approach [18], where the overall benefit can be calculated by fuzzy inference in three steps: fuzzification, inference and defuzzification.

1) The time benefit is firstly normalized by $t_{\%} = (t_{orig} - t)/t_{orig}$ where t_{orig} is execution time when no offloading is done. The membership degree of $t_{\%}$ to a specific benefit level is determined by membership function. For example, the membership degrees of $t_{\%} = 0.28$ are: $\mathrm{mf}^{time}_{none}(0.28) = 0.0983$ (*none*), where $\mathrm{mf}^{time}_{none}$ is member function of time benefit for level '*none*', and similarly 0.9193 (*low*), 0.0120 (*medium*), and 0 (*high*) according to membership functions shown in Fig. 1. Similarly, the energy benefit is fuzzified as well.
2) The firing degree of a rule is calculated based on the chosen conjunction operation and membership degrees of both time and energy benefit. For example, the chosen operation is product t-norm $T_p(a, b, \ldots) = ab\ldots$, and the membership degree of energy benefit for '*medium*' is 0.5642. Then, the firing degree is calculated by $T_p(0.9193, 0.5642) = 0.9193 \times 0.5642 = 0.5187$ for the rule shown in TABLE I.
The final benefit of a rule is estimated based on the conjunction of its member function, its firing degree, and its weight. For instance, the final benefit for the rule in TABLE I is: $T_p(0.5187, 0.85, \mathrm{mf}^{final}_{medium}) = 0.5187 \times 0.85 \times \mathrm{mf}^{final}_{medium} = 0.4409 \times \mathrm{mf}^{final}_{medium}$

where $\mathrm{mf}_{medium}^{final}$ is member function of the final benefit for '*medium*'.

3) The aggregation of final benefit is calculated based on the chosen disjunction operation and all outputs of rules. For example, the chosen operation is minimum t-norm $T_{min}(a, b, \ldots) = \{a, b, \ldots\}$. Then, the final benefit result function is $\mathrm{rf}(x) = T_{min}(\ldots, 0.4409 \times \mathrm{mf}_{medium}^{final}, \ldots)$. The final benefit b value is the center of $\mathrm{rf}(x)$: $b = \frac{\int \mathrm{rf}(x) x \mathrm{d}x}{\int \mathrm{rf}(x) \mathrm{d}x}$.

Similarly to benefit aggregation, fuzzy inference is also applied to aggregate offloading risks and the final risk is denoted as r. Both b and r are within range $[0, 1]$.

D. Risk-Based Decision

The final benefit aggregated by user preference depends on mapping matrix X and the final risk. Thus b and r in previous section are actually $b(X)$ and $r(X)$. The offloading problem is to find mapping X to maximize the aggregate benefit, and meanwhile, the constraint is satisfied that the aggregate risk is smaller than the benefit:

$$\max\ b(X), \tag{12}$$

$$\text{s.t. } b(X) > r(X). \tag{13}$$

The offloading decision considers trade off of benefit and risk. The offloading is allowed only when the offloading motivation or benefit overwhelms the offloading risk. To solve the above problem, we design an ant-algorithm based approach in Alg. 1.

Algorithm 1 Algorithm overview.

```
Initialization
repeat
    Solution construction
    Local search
    Pheromone update
until stopping criteria is reached
```

The previous problem can be represented by graph $G = (\Theta, \Lambda)$ where $\Theta = \{C, S\}$. Edges Λ connect components to sites. An ant at component vertex chooses site vertex as next vertex to go and leaves pheromone on the trail.

The algorithm maintains a counter γ and a pheromone matrix τ. At the beginning, γ is set to 0, and τ_{ij} is set to 1 where $1 \le i \le m$ and $1 \le j \le n$. Then, three steps are executed in a loop until the result is reached. γ increases by 1 each round in loop. τ is used to guide the transition in solution construction step, and is updated in pheromone update step. Two stopping criteria are: the counter reaches γ_{max}, and the result stagnates. When either criterion is met, the algorithm stops.

1) Solution Construction: The solution construction is shown in Alg. 2. The ant goes from component i to site j with the probability:

$$p_{ij}(\gamma) = \frac{\tau_{ij}(\gamma)}{\sum\limits_{1 \le k \le n} \tau_{ik}(\gamma)}. \tag{14}$$

Algorithm 2 Algorithm: solution construction.

```
Set X according to predefined assignment, and put unmovable component index into set I
while |I| ≤ m do
    Choose i uniformly at random,
        where 1 ≤ i ≤ m and i ∉ I
    Choose j randomly with probability in (14),
        where 1 ≤ j ≤ n
    x_ij ← 1
    I ← I ∪ {i}
end while
```

The ant algorithm is based on the following philosophy: high pheromone accumulated on the edge indicates this assignment is a potential good assignment; the ant chooses a potential good edge intentionally. However, the ant does not give up the choices of other edges.

Algorithm 3 Algorithm: local search.

```
I ← ∅
for 1 to α do
    Choose i uniformly at random,
        where 1 ≤ i ≤ m and i ∉ I
    J ← ∅
    while |J| ≤ n do
        Choose j uniformly at random,
            where 1 ≤ j ≤ n and j ∉ J
        X' ← X, and change assignment from i to j in X'
        if b(X) < b(X') and b(X') > r(X') then
            X ← X'
        end if
        J ← J ∪ {j}
    end while
    I ← I ∪ {i}
end for
```

2) Improvement Procedure: The neighborhood X' is obtained by changing one assignment in the solution X. The improvement procedure searches the neighborhood area and finds the valid local optimal solution as shown in Alg. 3. The local search improves X through iterations. The parameter α controls the iteration times. The high α means the local search algorithm will try to explore more neighbors. The value range of α is $[1, m]$.

3) Pheromone Update: The elements of pheromone matrix are updated as:

$$\tau_{ij}(\gamma + 1) = \tau_{ij}(\gamma) + \Delta\tau_{ij}(\gamma), \tag{15}$$

where

$$\Delta\tau_{ij}(\gamma) = \begin{cases} \eta\lambda_\gamma & \text{if } x_{ij} = 1 \text{ in current } X_\gamma \\ \eta^* & \text{if } x_{ij} = 1 \text{ in best } X_{best} \\ \eta\lambda_\gamma + \eta^* & \text{if } x_{ij} = 1 \text{ in } X_\gamma \text{ and } X_{best} \\ 0 & \text{otherwise} \end{cases}. \tag{16}$$

Two parameters η and η^* control the search scheduling. η is set to 1 at the beginning and varies in the run, while η^* is fixed in the whole process. In two cases, pheromone updates in alternative way that is different from (16):

1) When the best result ever found X_{best} has been improved by current solution X_γ, the η is reset to 1 and all τ_{ij} are reset to 1. The resetting remove historical information and intensify the search around new direction X_{best}.
2) When current solution X_γ reaches X_{best}, which means the focus on X_{best} is to too high, η is increased by 1 and all τ_{ij} are reset to new η to diversify the search.

The risk constraint in the problem is treated by parameter λ. When the solution X does not obey risk constraints, the positive feedback on pheromone is removed by paying penalty:

$$\lambda_\gamma = \begin{cases} 0 & \text{if } b(X_\gamma) < r(X_\gamma) \\ 1 & \text{otherwise} \end{cases}. \quad (17)$$

IV. Evaluations

We evaluate our ant based algorithm by simulation in MATLAB. We generate 50 test cases for evaluation. In each case, parameters for time, energy, privacy and reliability factors are generated. For time factor, one application graph and one surrogate graph are generated. The application component number is set to 5 and the candidate surrogate site number is set to 3. The workload **w** and site processing ability **u** are generated uniformly at random in range $[1, 100]$ and $[1, 10]$ separately. The data exchange amount F and network throughput D are uniformly random distributed in range $[1, 10]$ and $[1, 100]$ separately. This parameters are set in order to meet the assumption that the application is computation intensive as the average time spent on computation is much greater than data exchange. For energy factor, the coefficients of mobile devices are similar to [15]. The CPU coefficients of dynamic and static parts are random in range $[4, 6]$ and $[1, 1.5]$ separately. The RF coefficients of dynamic and static parts are random in range $[2, 3]$ and $[1, 1.5]$ separately. For privacy and reliability factors, the event probabilities are random in range $[0, 0.2]$. These events are assumed to happen in low probability in real situations.

A. η^* impact

In the proposed algorithm, there are two parameters η^* and α for performance tuning. The parameter η^* indicates the search direction in the whole solution space. In every iteration of searching, the best solution ever found is always assigned more pheromone, which guides the ant to go to that direction in high probability in following iterations. The performance impact of η^* is shown in Fig. 2, which shows the algorithm performance when α values are fixed at 1, 3 and 5. In the figure, the y axis represents the average case number and the x axis represents the iteration sequence number when the solution X is found. Every point (x, y) in the figure represents x cases reach their final solutions in the y^{th} iteration. The lines in the figure represent the case numbers distributed in iteration number range $[1, 50]$. Since the total case number is 50, the area below each line should be 50.

In all three situations of $\alpha = 1, 3, 5$, the line $\eta^* = 5$ is higher than $\eta^* = 3$ and the line $\eta^* = 3$ is higher than $\eta^* = 1$ in iteration range $[1, 20]$. And meanwhile, the line $\eta^* = 5$ is lower than $\eta^* = 3$ and the line $\eta^* = 3$ is lower than $\eta^* = 1$ in iteration range $[25, 50]$. When η^* raises, more cases that are solved originally in higher iteration numbers $[25, 50]$ will be solved in less iterations, which leads to the case number increases in range $[1, 20]$. When η^* increases, more pheromone will accumulate on the links that belong to the best solution X_{best}. This accumulated pheromone guides the ant to pick those links more often than other links, so that the ant goes to the direction of the X_{best}. The higher η^* is, the better sense of direction the ant has, which avoid ant's heading in random direction and save iterations.

By comparing figures of the same α values, we also find that the more cases reach solutions in small iteration from line $\eta^* = 1$ to line $\eta^* = 3$ than from line $\eta^* = 3$ to line $\eta^* = 5$. This is obvious in range $[1, 10]$: the incremental case number between line $\eta^* = 1$ and line $\eta^* = 3$ is larger than that between line $\eta^* = 3$ and line $\eta^* = 5$. The incremental case number decreases when the η^* increases, which will finally lead to the limit situation when increasing η^* does not make iteration number smaller. When η^* is too high, the algorithm will always search around the point X_{best} in solution space and often hit it again. When algorithm hits the X_{best} again, the relative high value of η^* is decreased by the way of increasing η in case 2 of pheromone update step in algorithm to guarantee the search diversity.

B. α impact

While parameter η^* guides the global search, the parameter α controls the local search. In Fig. 2a,2b and 2c, the case number from line $\alpha = 1$ to $\alpha = 3$ and from line $\alpha = 3$ to $\alpha = 5$ increases in range $[1, 12]$ and decreases in range $[23, 50]$, which means some cases that are solved originally in high iteration numbers are solved in less iterations. This is obvious in iteration range $[1, 12]$ where the line $\alpha = 3$ is above $\alpha = 1$. Similar phenomenon is also seen in Fig. 2d,2e,2f and Fig. 2g,2h,2i. When α increases, the algorithm searches the near solution around the picked X. The near solution is the solution that has only one different assignment of component to site. Since the component number is limited, the largest α value is bounded by component number. The higher the α is, the more neighbors the algorithm explores in each iteration, and the earlier the algorithm finds the best solution in the dedicated area, which leads to improvement of performance from iteration number aspect. However, the local search costs more time when we increase local search quality. Although the iteration number is decreased, the time spent for each iteration is increased because high α requires more attempts to explore neighbors.

V. Conclusions

In this paper, we proposed a multi-factor multi-site risk-based offloading decision model that is generic and extendible. The offloading decision is made based on a comprehensive offloading risk evaluation. To show how the model works, we used four offloading impact factors, including two benefit factors and two risk factors. We used fuzzy inference to aggregate the overall offloading benefits and risks based on the mobile cloud users preference, and used ant-based algorithm to calculate the assignment from application components to surrogate sites. The performance evaluation presents the practicality of the presented solution.

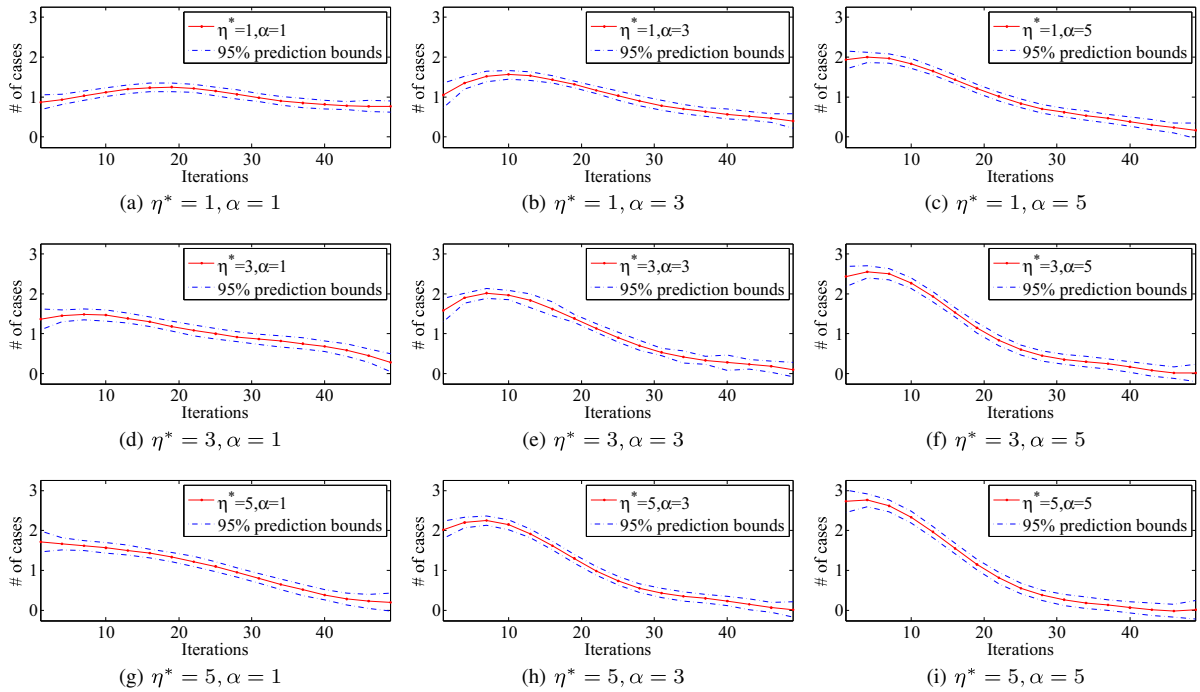

Fig. 2. Performance tuning.

ACKNOWLEDGMENT

The authors would like to thank NSF #1239396 grant to support the research on the MIDAS project.

REFERENCES

[1] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[2] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proceedings of IEEE INFOCOM*, 2012, pp. 945–953.

[3] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 270–284, 2011.

[4] K. Sinha and M. Kulkarni, "Techniques for fine-grained, multi-site computation offloading," in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 2011, pp. 184–194.

[5] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," in *Mobile Computing, Applications, and Services*. Springer, 2012, pp. 59–79.

[6] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[7] R. K. Ma, K. T. Lam, and C.-L. Wang, "excloud: Transparent runtime support for scaling mobile applications in cloud," in *International Conference on Cloud and Service Computing (CSC)*. IEEE, 2011, pp. 103–110.

[8] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: enabling mobile phones as interfaces to cloud applications," in *Middleware*. Springer, 2009, pp. 83–102.

[9] S. Ou, Y. Wu, K. Yang, and B. Zhou, "Performance analysis of fault-tolerant offloading systems for pervasive services in mobile wireless environments," in *IEEE International Conference on Communications (ICC)*. IEEE, 2008, pp. 1856–1860.

[10] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*. IEEE, 2008, pp. 1–8.

[11] D. Shin, K. Kim, N. Chang, W. Lee, Y. Wang, Q. Xie, and M. Pedram, "Online estimation of the remaining energy capacity in mobile systems considering system-wide power consumption and battery characteristics," in *18th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2013, pp. 59–64.

[12] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "Appscope: Application energy metering framework for android smartphone using kernel activity monitoring," in *USENIX ATC*, 2012.

[13] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. ACM, 2010, pp. 105–114.

[14] W. Jung, C. Kang, C. Yoon, D. Kim, and H. Cha, "Devscope: a nonintrusive and online power analysis tool for smartphone hardware components," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. ACM, 2012, pp. 353–362.

[15] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system." in *SOSE*, 2013, pp. 494–502.

[16] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the effectiveness of model-based power characterization," in *USENIX Annual Technical Conf*, 2011.

[17] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 317–328.

[18] Q. Ni, E. Bertino, and J. Lobo, "Risk-based access control systems built on fuzzy inferences," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 250–260.