Qualitative Economic Model for Long-Term IaaS Composition

Sajib Mistry[1], Athman Bouguettaya[1], Hai Dong[1](✉), and Abdelkarim Erradi[2]

[1] School of Science, RMIT University, Melbourne, Australia
{sajib.mistry,athman.bouguettaya,hai.dong}@rmit.edu.au

[2] Department of Computer Science and Engineering,
Qatar University, Doha, Qatar
erradi@qu.edu.qa

Abstract. We propose a new qualitative economic model based optimization approach to compose an optimal set of infrastructure service requests over a long-term period. The economic model is represented as a temporal CP-Net to capture the provider's dynamic business strategies in qualitative service provisions. The multidimensional qualitative preferences are indexed in a k-d tree to compute the preference ranking of a set of incoming requests. We propose a heuristic based sequential optimization process to select the most preferred composition without the knowledge of historical request patterns. Experimental results prove the feasibility of the proposed approach.

1 Introduction

An Infrastructure-as-a-Service (IaaS) provider offers Virtual Machines (VMs) as services in a cloud market [2]. An IaaS service (VM) is a configuration of functional or resource attributes, such as CPU, memory, and network units, and Quality of Services (QoSs) attributes, such as availability, throughput, response time and price [2]. Typical IaaS providers, such as Amazon, Windows Azure and Rackspace preconfigure their services and set the prices of those services [2]. However, there exists a different IaaS model where consumers are allowed to create custom IaaS requests (custom configurations of the functional and non-functional attributes) [6]. The prices of the services are either set by the provider [2] or the consumers are allowed to quote their own prices for the services [18].

The long-term IaaS composition is defined as to select an optimal set of custom consumer requests that maximizes the profit of the provider [10]. In our previous research, the prices of each unit of resource and QoS attributes are advertised by the provider. Hence, two identical requests generate the same level of revenue. However, they can be distinguished from their usage behaviors, such as under-utilization and over-utilization [11]. We have proposed an quantitative economic model for the IaaS provider that predicts consumers' service usage behaviours and calculates the operation costs of the requests [10].

According to [18], the functional and QoS attributes in a long-term consumer request are variable over a time period, qualitative in nature and closely related

© Springer International Publishing Switzerland 2016
Q.Z. Sheng et al. (Eds.): ICSOC 2016, LNCS 9936, pp. 317–332, 2016.
DOI: 10.1007/978-3-319-46295-0_20

with the price. For example, a consumer may prefer an IaaS service that has less response time in the first year. While in the second year, the consumer may find response time is less important and quote lower price to save the cost as much as possible. The proposed quantitative economic models [10] fail to capture the business strategies of the provider to evaluate such long-term requests.

A qualitative IaaS economic model should capture the long-term business strategies of the provider. For example, the mobile carrier companies normally create different plans for different types of consumers [9]. Sometimes they offer mobile phones in low prices as a business strategy. Similarly in the cloud market, if a provider finds its CPU units are more fault-tolerant than the hard disks, it may prefer CPU-intensive service requests more than the space-intensive requests to avoid probable Service Level Agreement (SLA) violations. The acceptance or rejection of an incoming request should follow an strategy as accepted requests are committed for the whole period. Partial service provisioning is treated as a SLA violation. We assume the provider operates with fixed amount of resources [5]. We focus on the deterministic arrivals of the incoming requests, i.e., all the requests are known at the start of the composition.

Existing qualitative economic models represent the consumers' qualitative preferences [13,15,18]. They do not consider the following issues related to the long-term composition from the provider's perspective:

– **Dynamic temporal semantic preferences:** In a typical short-term composition, the qualitative preferences remain static during the composition time [13,15,18]. The relative ordering of the provider's preferences may vary in the long-term period. For example, a provider may prefer providing CPU based services over Network based services in the first year and prefer the opposite in the second year. It creates a set of temporal segments of preferences in the economic model. The semantics of preferences may not be static during the whole period of composition. It is relative to the competitions in the market [5,9]. For example, 10ms response time is treated as a high QoS in this year, but it may become a moderate QoS in the next year because of an upgrade of the hardware in the market.
– **Temporal mismatch between the service request and the provider's preferences:** The long-term service requests may not have exact temporal match with the temporal segments of the economic model. For example, the provider has a preference on provisioning CPU intensive services in January and Network intensive services in February. If a service request spans from the middle of January to the middle of February, any economic model could not be applied directly for evaluating the merit of the requests.

We propose a novel approach to compose requests using the provider's long-term qualitative economic model. We represent the economic model as Temporal CP-Nets (TempCP-Net), which is a collection of dynamic CP-nets [4] spanning over the composition time segments. CP-Nets [4] is a compact and intuitive formalism for representing and reasoning with conditional preferences under the ceteris paribus ("all else being equal") semantics. The dynamic semantics of the preferences are indicated using a Conditional Preference Table (CPT) [4] of the TempCP-Net. The temporal mismatch between the service request and the provider's preferences is solved through the semantic temporal segmentation of the requests which preserves the inherent dependencies among the attributes in the original long-term request. Moreover, the induced preference graph [15] from TempCP-Net is indexed in a multidimensional k-d tree [3] to effectively match with the multidimensional attributes of the consumer requests. We transform the TempCP-net into a k-d tree as nodes in the preference graph could be considered as points and k-d tree is widely used for multidimensional point query in different applications [1]. The k-d tree is treated as an objective function to compute the global preference ranking of a composition. Hence, we transform the IaaS composition as a preference maximization optimization problem.

We consider both the global and local optimization approaches to select the optimal composition considering k-d tree represented TempCP-net as the objective function. A typical Dynamic Programming (DP) [8] based global approach considers all the requests at the time of optimization. This may pose a scalability issue in the runtime due to comparisons among a large number of candidate solutions. We devise a heuristic based local optimization approach to accept or reject requests in each segment so that local decisions are collectively converged to an acceptable approximate global optimal composition.

2 Related Work

An economic model for profit maximization of cloud service providers is proposed in [10]. The profit maximization based IaaS composition in stochastic arrival of requests is proposed in [10]. An economic model of federated clouds is described in [5]. The model evaluates the cost of using resources from a cloud federation and develops a resource management core for the profit maximization. A CP-Net based economic model is proposed for the service composition from the consumers' perspective in [18]. The consumer preferences are fine grained using Weighted CP-Net (WCP-Net) in [16]. Service selection from incomplete user preferences are proposed in [15]. Such models do not consider temporal changes in the providers' preferences for a long-term period.

An integer programming formulation for IaaS composition is proposed in [11]. Heuristic algorithms are proposed to determine whether a new request can be admitted without impacting accepted requests in [17]. Meta-heuristic optimization is proposed for stochastic incoming requests in [10]. In operations research, the restaurant reservation problem is solved using Dynamic Programming techniques (DP) [8]. The DP is used to optimize the IaaS service scheduling in [5]. Markovian Decision Process (MDP) is used as a machine learning technique to solve sequential iterative optimization in [14]. As we do not have the history of previous incoming requests, we devise a heuristic based sequential optimization technique using only the current set of incoming requests.

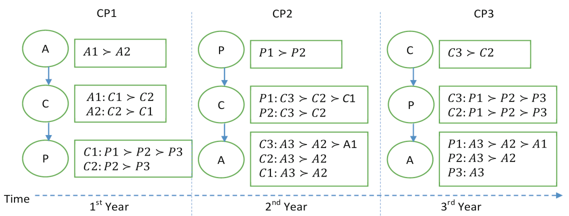
3 Motivation: A Qualitative IaaS Economic Model

Let us assume, a new IaaS provider starts offering virtual CPU services associated with QoS of availability for simplicity. It can provide maximum 100 CPU units and 100 % availability. The provider's qualitative preferences on CPU, availability, and price can be interpreted into three semantic levels - high, moderate, and low, as shown in Fig. 1(a). The provider now has different preference ranks based on its annual goals in a three-year period. For the first year, the provider prefers to provide high quality services with relatively lower prices, to build its reputation in the market. Hence, the provider decides that "availability" of a service is the most important attribute, followed by "CPU" and "price". For the second year, the provider expects to provide services with higher prices and relatively lower resources and QoS to maximize the profit. Thus "price" decides "CPU" and "availability". For the third year, the provider considers that the ageing infrastructure may cause problematic CPUs and lower availability. Therefore the provider's preference is that providing relatively lower "CPU" has a higher priority than relatively higher "price" and lower "availability". The interpretation of the semantic levels remains static in the three years for simplicity.

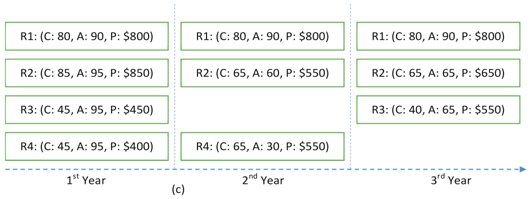
The CP-Net can elegantly represent these qualitative preferences. For example, an arc from "CPU" to "availability" means the preference of "availability" depends on the preference of "CPU" units. The provider's economic preferences are captured in a Temporal CP-Net denoted as TempCP-Net, which is a collection of time-period based CP-Nets. As the provider has annual preferences, the three-year TempCP-Net is a set {(CP1, Year 1),(CP2, Year 2) (CP3, Year 3)}, in which each subcomponent corresponds to an annual preference (Fig. 1(b)). CP1 captures the first-year reputation building strategy in Fig. 1(b). Hence, the "high" availability has a higher priority than the "moderate" availability, i.e., $A1 \succ A2$. Note that, the "low" availability ($A3$) is not in the provider's preference in CP1. The choice of availability dictates the choice of CPU units. If the "high" availability ($A1$) is chosen, the provider prefers to provide the "high" CPU units than the "moderate" CPU units ($C1 \succ C2$). However, if the "moderate" availability ($A2$) is chosen, the provider prefers to provide the "moderate" CPU units than the "high" CPU units ($C2 \succ C1$). This is because a moderate QoS may not increase the reputation as expected, thus packaging it with lower CPU units may increase the probability to reduce SLA violations than packaging with higher CPU units. Finally, the price of the service is chosen based on the selection of the levels of availability and CPU units. As this is a reputation building phase, the provider will not charge "high" price ($P1$) while providing "moderate" CPU units ($C2 : P2 \succ P3$). In CP1, the most preferred service provision is $(A1, C1, P1)$ and the least preferred choice is $(A2, C1, P3)$. Similarly, CP2 and CP3 capture the profit maximization and risk management strategies in the second and third years respectively. In CP2, the most preferred service provision is $(P1, C3, A3)$ and the least preferred service is $(P2, C2, A2)$ expressing the preference on the higher price. In the third year, the most preferred service provision is $(C3, P1, A3)$ and the least preferred service is $(C2, P3, A3)$.

Attribute	Symbol	Semantic	Meaning
CPU (C)	*C1*	High	71 to 100 units
	C2	Moderate	46 to 70 units
	C3	Low	1 to 45 units
Availability (A)	*A1*	High	71% to 100%
	A2	Moderate	31% to 70%
	A3	Low	1% to 30%
Price (P)	*P1*	High	More than $1000
	P2	Moderate	$500 to $999
	P3	Low	$1 to $499

(a)



(b)


R1: (C: 80, A: 90, P: $800)
R2: (C: 85, A: 95, P: $850)
R3: (C: 45, A: 95, P: $450)
R4: (C: 45, A: 95, P: $400)
R1: (C: 80, A: 90, P: $800)
R2: (C: 65, A: 60, P: $550)
R4: (C: 65, A: 30, P: $550)
R1: (C: 80, A: 90, P: $800)
R2: (C: 65, A: 65, P: $650)
R3: (C: 40, A: 65, P: $550)
1st Year
2nd Year
3rd Year

(c)

Requests	Preference Ranking			
	1st Year	2nd Year	3rd Year	Total
{R1}	2	N/A	N/A	N/A
{R2}	2	9	9	20
{R3}	9		4	13
{R4}	9	4		13
{R3,R4}	2	4	4	10
Other Combinations	SLA Violation	SLA Violation	SLA Violation	SLA Violation

(d)

Fig. 1. (a) Semantic representation of preferred service attributes, (b) A TempCP-Net, (c) Incoming requests, (d) The preference ranking table

Let us assume, four different requests, $\{R1\}, \{R2\}, \{R3\},$ and $\{R4\}$ arrive at the beginning of the composition (Fig. 1(c)). For simplicity, a request is specified in annual segments. In Fig. 1(c), $(C : 80, A : 90, P : \$800)$ is the first year segment of $\{R1\}$, which means that the consumer requires a VM with 80 CPU units and 90 % availability and is able to pay $800 for this service in the first year. The annual requirements of $\{R2\}, \{R3\}$ and $\{R4\}$ are described in the same way. As there are four requests, the optimal composition will be selected from $2^4 = 16$ combinations of the requests in the brute-force approach. The TempCP-net (Fig. 1(b)) provides the objective function for the optimal IaaS composition selection. The preference ranking (lower values indicate higher preference ranks) of the combinations of the requests is retrieved through the matching between the TempCP-net and the combinations of the requests. For example, the first year segment of $\{R1\}$ falls into the "high" CPU units, "high" availability, and "moderate" price in Fig. 1(a). It is the 2nd ranked preference in the first year. However, the second year segment of $\{R1\}$ requires the "high" availability in the "moderate" cost, which is out of the preference (N/A) in CP2. The total preference rank of each request is tabulated under "Total" in Fig. 1(d). As two segments of $\{R1\}$ are out of the preference ranking, its total rank is N/A. $\{R3\}$ and $\{R4\}$ are combined into a request $\{R3, R4\}$. We cannot consider the other combinations due to the constraint of maximum 100 CPU units. According to the total qualitative ranks, $\{R3, R4\}$ is the optimal composition.

A heuristic based sequential optimization process may produce the global solution in fewer number of comparisons. Let us assume, the sequential optimization operates in the right to left year sequence (3rd, 2nd and 1st). In the 3rd year $2^3 = 8$ comparisons are performed and only the highest ranked $R3$ is accepted. The 9th ranked $R2$ violates the constraint of maximum 100 CPU units when it is combined with $R3$. As $R1$ (N/A ranking) and $R2$ are already rejected, they are not considered in the subsequent optimization. The local optimizations in the 2nd and 1st year accepts the remaining $R4$ in the solution. Hence, the optimal solution $\{R3, R4\}$ is produced in 10 comparisons.

4 The Temporal CP-Net Based Economic Model

We require not only an intuitive tool for structuring the provider's preferences, but also a support for an efficient optimization process. We assume a set of functional and non-functional attributes $V = \{X_1, ..., X_n\}$ with finite domains $\{D(X_1), ..., D(X_n)\}$ and semantic domains $\{S(X_1), ..., S(X_n)\}$. Typical functional attributes are CPU (C), Network bandwidth (NB), and Memory (M), and QoS attributes are Availability (A), Response time (RT), Throughput (TP) and Price (P). The numerical value x_n in $D(X_n)$ is mapped into a semantic value s_n in $S(X_n)$ using a mapping table, $s_n = Sem\_Table(X_n, x_n)$. Figure 1(a) is such a semantic table that maps 70–100 units of CPU as a "high" CPU value. In the long-term, the preference order and related semantics of V remain constant for a time period, but they may get changed in the next period. Let us assume, the total composition time, T is divided into m intervals $\{I_1, I_2, ..., I_m\}$ where, $T = \sum_{i=1}^{m} I_i$. In an interval I_k, the IaaS provider can specify a preference ranking of service configurations over complete assignments on V with the semantic domain $Sem\_D^{I_k}(V)$. The set of all service configurations is denoted as O^{I_k} for the interval I_k. A preference ranking is a total order ($\succeq$) over the set of service configurations: $o_1 \succeq o_2$ means that a configuration o_1 is equally or more preferred than o_2. We use $o_1 \succ o_2$ to denote the fact that provisioning service o_1 is more preferred than o_2 (i.e., $o_1 \succeq o_2$ and $o_2 \not\succeq o_1$), while $o_1 \sim o_2$ denotes that the provider's preference is indifferent, i.e., $o_1 \not\succeq o_2$ and $o_2 \not\succeq o_1$.

Direct assessment of a long-term preference relation is generally infeasible due to the exponential size of $O^{I_k} \mid \forall k \in [1, m]$. We represent the long-term economic model in a Temporal CP-Net (TempCP-Net), which is a set of CP-Nets and semantic mapping tables over different intervals defined as TempCP-Net $= \{(CP^{I_k}, Sem\_Table^{I_k}, I_k) \mid \forall k \in [1, m]\}$. A CP-Net can concisely specify a preference relation in a graphical structure. A CP-net in the interval I_k, CP^{I_k} is a directed graph G over V whose nodes are annotated with conditional preference tables $CPT(X_i)$ for each $X_i \in V$. In this paper, we focus on only acyclic CP-Nets. Each conditional preference table $CPT(X_i)$ associates a total order $\succ^i_u$ with each instantiation u of X_i's parents $Pa(X_i) = U$ [13]. For example, in $CP1$, $A = Pa(C)$ and the $CPT(C)$ contains $\{A1, A2\}$ while preferences are made over $\{C1, C2\}$ (Fig. 1(b)). The preference $o \succ \acute{o}$ is a *consequence* of TempCP-Net, iff $o \succ \acute{o}$ holds in all preference orderings consistent with the *ceteris paribus* preference statements ("all else being equal") encoded by the CPTs of the TempCP-Net [16]. The set of consequences $o \succ \acute{o}$ of an acyclic TempCP-Net constitutes a partial order over the service configuration. This partial order can be represented by an acyclic directed graph, referred to as the *induced preference graph*. The nodes of the induced preference graph correspond to the complete assignments to the variables of the network. There is

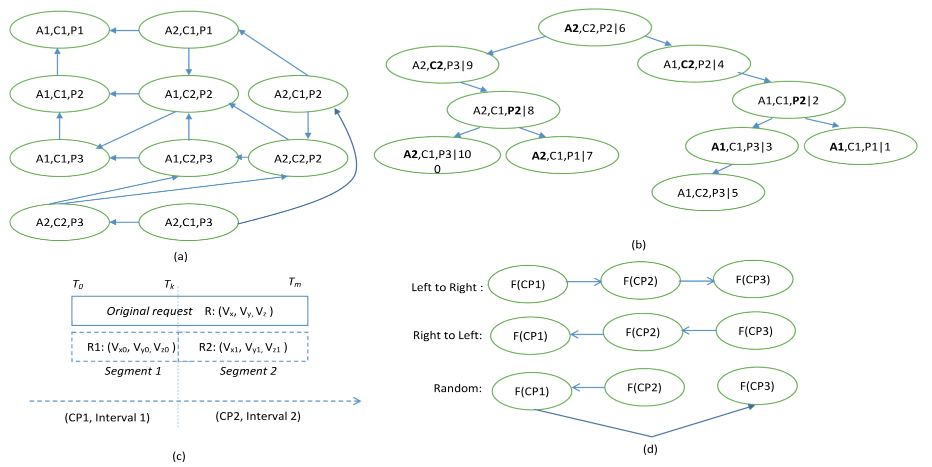


Fig. 2. (a) The induced preference graph of CP1, (b) The k-d tree representation, (c) Temporal semantic segmentation of a request, (d) Some sequential orders for local optimization

an edge from node $\acute{o}$ to node o iff the assignments at $\acute{o}$ and o differ only in the value of a single variable X. Given the values assigned by $\acute{o}$ and o to $Pa(X)$, the value assigned by o to X is preferred to the value assigned by $\acute{o}$ to X. Figure 2 depicts the induced preference graph of $CP1$. There is no outgoing edge from $(A1, C1, P1)$ as it is the most preferred request configuration. Similarly, there is no incoming edge to $(A2, C1, P3)$ as it is the least preferred configuration. As the $CPT(CP1)$ states $A1 \succ A2$, there is an edge from $(A2, C1, P1)$ to $(A1, C1, P1)$ considering the *ceteris paribus* preference statements. The induced preference graph (total ordering) of all the configurations is created using pair wise comparison (ordering queries) of the configurations [13]. If n is the number of attributes in the TempCP-net and q is the number of output configurations in an interval, the time complexity for ordering queries in an interval is $O(nq^2)$.

4.1 The *k*-d Tree Indexing of the Induced Preference Graph

Given a semantic request configuration $Sem\_Req = (s_1, ..., s_n) \mid$ where $s_i \in S(X_i)$, and $X_i \in V$, the induced preference graph enables searching the preference ranking of $(s_1, ..., s_n)$. Such a graph based searching approach requires linearly traversing over the graph (time complexity $O(n)$) [13]. Considering the tuple $(s_1, ..., s_n)$ as a multidimensional vector, we improve the search process using the k-d tree [1]. The k-d tree is a binary tree in which every node is a k-dimensional point (Fig. 2(b)). Every non-leaf node can be thought of as implicitly generating a splitting hyperplane that divides the space into two parts, known as half-spaces. Points on the left and right sides of this hyperplane are represented by the left and right subtree of that node respectively. We use the canonical method to construct the k-d tree [3]:

- The selection of splitting planes follows a cycle as the construction algorithm moves down on the tree. For example, in Fig. 2(b), the root is an "Availability-aligned" plane, the root's children both have "CPU-aligned planes", the root's grandchildren have "Price-aligned" planes, the root's great-grandchildren have again "Availability-aligned" planes, and so on.
- As all the n points are available from the induced preference graph, we insert points by selecting the median of the points being put into the subtree, with respect to their coordinates in the axis being used to create the splitting plane. This would result in a balanced k-d tree construction in $O(n\ log(n))$ times [3]. Each node in the k-d tree is annotated with its respective preference order from the induced graph. For example, the root node $(A2, C2, P2)$ is annotated with the preference ranking 6 in Fig. 2(b).

Starting with the root node, the searching algorithm moves down on the tree recursively, in the same way that it would if the search point was being inserted. If the search point is matched with a node, it returns the annotated ranking value. For example, the search for the request $(A2, C1, P3)$ returns rank 10 using only 4 comparisons. A non-matched search point is discarded in the composition. The time complexity in k-d tree searching of an interval is $O(log(n))$.

4.2 Ranking of the Consumers' Requests Using the TempCP-Net

Let us assume, a consumer divides its service usage time in n intervals and service requirements in the intervals vary from each other. We define the request of consumer u over the composition time T as $R_u = \{(x_i, I_j) \mid x_i \in D(X_i), X_i \in V, \text{and } T = \sum_{j=1}^{m} I_j\}$. A set of N requests is represented as $\bar{R} = \{R_1, ..., R_N\}$. We combine the requests in $\bar{R}$ using the composition rules [10]:

$$\text{Summation rule: } \bar{x_i} = \sum_{i=1}^{N} x_i, \text{where } X_i \in \{C, M, NB, RT, P\} \qquad (1)$$

$$\text{Maximization rule: } \bar{y_i} = max(y_i), \forall i \in [1, N] \text{ where } Y_i \in \{A, TP\}$$

For example, the combined first year request $\{R1 : (C : 80, A : 90, P : \$800), R2 : (C : 85, A : 95, P : \$850)\}$ in Fig. 1(c) is $(C : 165, A : 95, P : 1650)$.

Note that, the intervals in consumer requests may be different from the intervals in the provider's TempCP-Net. In the first case, the starting time and the ending time of an *inclusive* request segment are from the same temporal segment of the TempCP-Net. For example, if a CP-Net in TempCP-net operates between 1st January and 31st January, a request spanning from 4th January to 25th January is an inclusive segment. As a single CP-Net is operating over the request, the request could be directly matched with the induced k-d tree from Sect. 4.1. In the second case, the starting time and the ending time of a *overlapping* request segment are from different temporal segments of the TempCP-Net. As more than one CP-Nets are operating over the request, we divide an overlapping request, R (interval $[T_0, T_m]$) into smaller inclusive segments. In Fig. 2(c), a request R is divided into R_1 and R_2 to match with corresponding $CP1$ and $CP2$. Note that, only attributes with *temporal semantics* require such a segmentation. For example, "Price" has *temporal semantics* in the consumer requests. If the consumer requires 100 units of CPU in 12 months for \$120, it still requires 100 units of CPU in every month but the monthly cost will be \$10. If the attribute X in R has *temporal semantics* and the segmentation is applied in $[T_j, T_k]$, the new value for X is calculated as follows:

$$x_i^{[T_j,T_k]} = x_i^{[T_0,T_m]} \times \frac{|T_k - T_j|}{|T_m - T_0|} \tag{2}$$

We define $Pref(\text{TempCP-Net}, \bar{R}) : V \rightarrow [1, n]$ as the ranking function that finds the preference order of $\bar{R}$ in the k-d trees of the TempCP-Net. As the TempCP-Net is constructed in semantic domains, we transform $\bar{R}$ into the semantic $\acute{\bar{R}} = \{(s_i, I_j) \mid s_i \in S(X_i), X_i \in V, \text{and } T = \sum_{j=1}^{m} I_j\}$ using the Sem_Table in the TempCP-Net. Each temporal segments in $\acute{\bar{R}}$ is matched with the corresponding temporal k-d tree using the matching process in Sect. 4.1. We denote the matching process in interval i as $M^i(s_i)$. Hence, ranking function can be defined as follows:

$$Pref(\text{TempCP-Net}, \bar{R}) = M^1(s_1) + ... + M^i(s_i) + ... + M^m(s_n) \tag{3}$$

5 Optimization Algorithm for IaaS Requests Composition

Given a set of N long-term requests $\bar{R}$ and the IaaS provider's TempCP-Net, the IaaS composition is to find an optimal set $\bar{r} \subseteq \bar{R}$ that minimizes the ranking output $Pref(\text{TempCP-Net}, \bar{r})$ in Eq. 3 (a lower value means a higher rank). There are two approaches to solve the temporal optimization problems: (a) global optimization, (b) sequential local optimization [8]. The global approach considers the entire time period and the input set at the time of composition. A common way of global optimization is the brute-force approach, that attempts all the combinations of requests over the entire composition period and finds the minimum one. The time complexity of this approach is exponential (2^N) which is not applicable in realtime applications. We formulate a dynamic programming (DP) approach that solves the optimization in super-polynomial time ($N^{O(N)}$) [8]. As the size of the input requests is in proportion to the length of the composition time, the global DP approach may not be feasible for a long-term composition. We propose a sequential local approach that divides the total time into segments according to the corresponding time intervals and each segment optimizes the requests that only operates in that interval [14]. This approach has a sequential effect for overlapping requests. Two local optimizations in different intervals may have different accept or reject opinion for the same overlapping request. As the quality of the final composition is dependent on the sequence order, we devise a heuristic based approach to approximate the optimal solution.

5.1 Dynamic Programming Based IaaS Composition

We propose a dynamic programming framework to weigh the benefits of accepting versus rejecting a request. Accepting a request will lead to immediate revenue, but it is possible that this acceptance will diminish future resource utilization for other requests. Dynamic Programming (DP) is an algorithmic paradigm that solves a given complex problem by breaking it into sub-problems (overlapping sub-problems) and stores the results of sub-problems to avoid repeated computation (optimal substructure) [8]. We denote $\bar{R}(N)$ as a set of N requests and $i \in [1, N]$ as the i_{th} request. If $C(\bar{R}(N), k)$ returns the optimal subset of requests of size k, it either accepts the N_{th} request (the k_{th} place is already filled) or rejects it (reduces $\bar{R}(N)$ to $\bar{R}(N-1)$). We formulate the DP as follows:

$$\begin{aligned} \bar{R_1} &= \{N \cup C(\bar{R}(N-1), k-1)\} \\ \bar{R_2} &= C(\bar{R}(N-1), k) \end{aligned} \tag{4}$$

$$C(\bar{R}(N), k) = \begin{cases} \bar{R_1}, \text{ if } Pref(\text{TempCP-Net}, \bar{R_1}) < Pref(\text{TempCP-Net}, \bar{R_2}) \\ \bar{R_2}, \text{ if } Pref(\text{TempCP-Net}, \bar{R_1}) \geq Pref(\text{TempCP-Net}, \bar{R_2}) \\ \{i\} \text{ if } k = 1 \text{ and } Pref(\text{TempCP-Net}, \{i\}) \text{ is minimum} \\ \emptyset \quad \text{if } k = 0 \end{cases}$$

In Eq. 4, $\bar{R_1}$ refers to the set that accepts the N_{th} request and $\bar{R_2}$ refers to the set that rejects the N_{th} request. The base case is defined on $K = 1$ (only one request output) that performs a linear search to find the highest ranked request i. The DP can reduce the re-computations of same sub-problems by constructing a temporary array in the bottom-up manner [8]. The complexity of finding $C(\bar{R}(N), k)$ is $O(N^k)$. The final optimal subset can have at most N requests. Hence, we can find the optimal solution (Sol) through the iterative operation in Eq. 5. The final complexity of the DP based solution is $O(N^{O(N)})$.

$$Sol = C(\bar{R}(N), i), \text{where } Pref(\text{TempCP-Net}, C(\bar{R}(N), i)) \text{ is minimum} \tag{5}$$

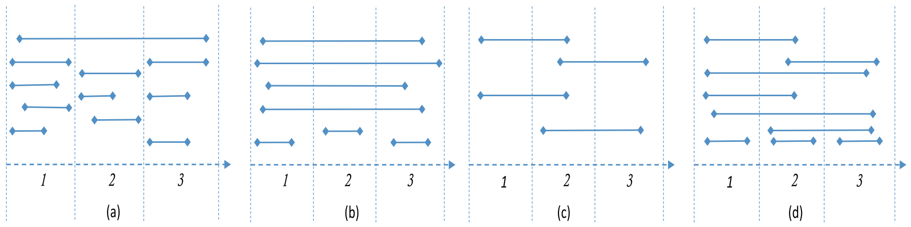


Fig. 3. The key composition scenarios (a) Almost Disjoint (b) Almost Overlapping (c) Chain, (d) Hybrid patterns

5.2 Heuristic Based Sequential Optimization

We use a heuristic based sequential aggregation of local optimizations to approximate the global optimization in reduced time complexity. Each interval in TempCP-Net only considers the request segments within its range and performs DP based optimization by using its own CP-Net as TempCP-Net (TempCP-Net $= CP_i$) in Eq. 4. These local optimizations in the intervals can run in parallel if there is no overlapping requests. However acceptance or rejection decision of overlapping requests can not be taken in parallel. Hence, different sequences of optimization order are performed to produce the optimal composition. For example, the left to right sequence first carries out optimization from the left intervals to the right intervals in Fig. 2(d). Finding the best sequence in sequential optimization is NP-Complete [8]. We build our heuristics by exploring several key composition scenarios:

- Almost Disjoint Pattern: In this pattern, the requests in an interval are mostly disjoint and evenly cover that interval (Fig. 3(a)). An interval that contains such a pattern can be assumed that rejecting an overlapping request may not affect the global ranking, as the overlapped request can be replaced by one or more disjoint requests with high possibility.
- Almost Overlapping Pattern: The requests in an interval are mostly long-overlapping in this pattern (Fig. 3(b)). The local optimization at a certain interval of a sequence may not be changed in other sequences as the acceptance and rejection of most of requests are decided in the first interval.
- Chain Pattern: In this pattern, the requests are short-overlapped and almost evenly distributed over the intervals (Fig. 3(c)). Several sequences of local optimization need to be applied to achieve the optimal result.
- Hybrid Pattern: Both the long-overlapping and short-overlapping requests are almost evenly distributed in this pattern (Fig. 3(d)). The final result should also maintain the ratio of different types of requests.

We formulate the following generic heuristics to find out the optimal sequences of local optimizations for different patterns. We define the overlapping ratio of a request N as follows:

$$O\_Ratio(N) = \frac{\text{Number of operating intervals of } N}{\text{Total number of intervals}} \tag{6}$$

- **Heuristic 1:** If most of the local optimizations reject a long-overlapping request independently, the final output should also reject it. If most of the local optimizations accept the request independently, the final output should also accept it. This heuristic is common in collective decision processes.
- **Heuristic 2:** If two set of requests $\{1, 2, ..., i, .., n\}$ and $\{1, 2, ..., j, ..., n\}$ produce the final ranking x and y respectively in an interval, a local optimization prefers accepting the request i than the request j if $O\_Ratio(j) < O\_Ratio(i)$ and $|x - y| < \tau$. τ is the highest acceptable difference in the ranking set by the provider. The heuristic prefers disjoint requests to the overlapping requests

Algorithm 1. The heuristic based sequential optimization

```
Input: The Request set($\bar{R}$), Acceptance window($l$) and Maximum additive ranking ($p$)
Output: The optimal composition
1: final solution = $\emptyset$
2: Run local dynamic programming based optimization in each of the $n$ intervals in parallel. Store the cumulative appearance frequency of a request in the first $l$ ranking (the acceptance window will be set by the provider). For example, if $l = 5$ and $A$ appears in both 1st rank and 3rd rank, the frequency of A is 2. By default the frequency is 0. Rank the requests based on their frequencies.
3: Add the rankings of a request from each interval. Accept the request and add to the final solution if its additive ranking is less than $p$ (set by the provider).
4: temporary solution = final solution
5: Generate the left to right sequence of intervals $(I_1, I_2, ...., I_m)$.
6: Start dynamic programming based optimization in the first interval. Add new requests in the temporary solution by following heuristic 2. Only add requests if there are available resources. After finishing optimization in an interval, continue adding new requests in the following intervals of the sequence. If the ranking of the temporary solution is greater than the final solution, set final solution = initial solution (update operation).
7: Generate the right to left sequence of intervals $(I_m, I_{n-1}, ....I_1)$ and try to update the final solution by following step 6.
8: Generate a new random sequence of intervals $(I_k, I_{n-1}, ....I_l)$ and try to update the final solution by following step 6. If the ranking of the final solution is improved, start step 8 again. Otherwise, return the final solution.
```

when optimizing an interval. It reduces the effect of sequencing by replacing overlapping requests with disjoint requests without affecting the ranking in individual optimization.

We devise a two-phase based approach (Algorithm 1) to incorporate these heuristics. In the first phase, we filter long-overlapping requests for the acceptance or the rejection (heuristic 1). The first phase is described in step 1 to 3 in Algorithm 1. It sets the final solution with a set of long-overlapping requests which are voted by the intervals independently. In the second phase, we add new requests in the final using heuristic 2 (step 4 to 8). At least three different sequences are generated and the final solution is updated only when the ranking of the solution is improved. The random generation of sequences are stopped when no improvements are made in the final ranking in Algorithm 1.

6 Experiments and Results

A set of experiments are conducted to evaluate the efficiency of the proposed approach. At first we compare the ranking of the sequential optimized composition with the global dynamic programming based composition. Next we compare the time complexity between the two approaches. All the experiments are conducted on computers with Intel Core i7 CPU (2.13 GHz and 4 GB RAM). Java is used to implement the algorithms.

6.1 Simulation Setup

As it is difficult to find a real world IaaS provider's business strategies, we synthetically create 10 different yearly temporal CP-Nets with 12 intervals in each TempCP-Net. Each TempCP-Net has 5 attributes (CPU, Memory Availability, Response time, Throughput and Price) and dependencies among the attributes are randomly generated. The values of an attribute are divided into 10 semantic levels (from high to low). Each CPT is filled with random conditional preferences, each of which is a random order of the attribute semantic values. We

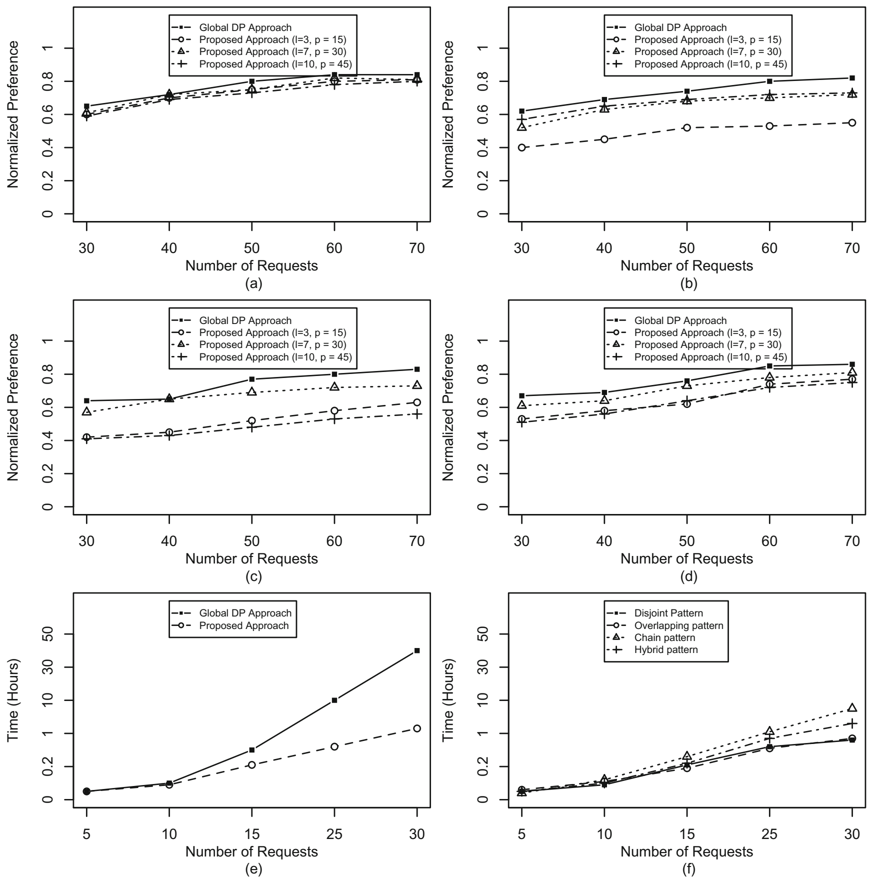


Fig. 4. Accuracy of the global and sequential approach in (a) disjoint pattern, (b) overlapping pattern, (c) chain pattern, (d) hybrid pattern, (e) The global vs sequential time complexity, (f) Time complexity of the sequential approach

create the user requests as a mixure of Google Cluster resource utilization [12], real world cloud QoS performance [7], and randomly generated availability and prices. Google Cluster data include CPU and Memory utilization and allocation time series of 70 jobs over a 1-month period. Real world QoS data [7] include two time series (i.e., response time and throughput) for 100 cloud services over a 6-month period. We randomly pick 70 Google Cluster jobs and make one-to-one mapping with the 100 sets of QoS data. A 6-month request is extended to a 12-month request using random duplication of segments. We create 4 different patterns of the input requests: (a) disjoint pattern: 80 % of the request set is segmented into requests of 1 to 2 month period. (b) overlapping pattern: 80 % of the request set is segmented into requests of 8–12 month period. (c) chain pattern: 80 % of the request set is segmented into requests of 2–8 month period. (d) hybrid pattern: It is a collection of 35 % disjoint requests, 35 % overlapping requests and 30 % of requests which spans in 2 to 8 month period. In each of the patterns, other types of requests are randomly distributed into different intervals.

6.2 Efficiency of the Heuristic Based Sequential Optimization

In the first experiment, we analyse the efficiency of the proposed heuristic based sequential optimization. As the proposed Algorithm 1 needs an acceptance window (l) and additive ranking (p) in its first phase (accepting long-overlapping requests), we use 3 different configurations of acceptance window: (a) conservative ($l = 3, p = 15$), (b) moderate ($l = 7, p = 30$) and liberal ($l = 10, p = 45$). Each of the request patterns (disjoint, overlapping, chain and hybrid) is filled with different numbers of requests ranging from 30 to 70. A request pattern filled with a certain number of requests is executed in 10 different TempCP-Nets using the proposed heuristic approach (with 3 different acceptance windows) and the global DP approach. The outputs are averaged and normalized as $(\frac{1}{ranking})$ (higher values mean top rankings, lower values mean low rankings). Figure 4(a),(b),(c) and (d) depict the performance of the approaches in the disjoint, overlapping, chain and hybrid patterns respectively. In all of the figures, the quality of the output is increased when the number of requests is increased. At least one of the acceptance windows produces close outputs to the DP based approach in higher number of requests. All three acceptance window configurations of the heuristic based approach perform close to the DP based approach in the disjoint and hybrid patterns (Fig. 4(a) and (d)). Among the three, only the output of conservative ($l = 3, p = 15$) configuration is not performing close to the DP based approach in the overlapping pattern (Fig. 4(b)). It may refuse too many possible overlapping requests in the final output composition. In Fig. 4(c), only the output of the moderate ($l = 7, p = 30$) configuration is acceptable and close to the DP approach in higher numbers of requests (Fig. 4(c)).

6.3 Time Complexity Analysis

Although the global DP based approach produces better results than proposed approach, it is not applicable in runtime. The convergence time of the global approach and the proposed approach are close for the smaller numbers of requests in Fig. 4(e). However, the convergence time of the proposed approach is significantly lower than the global approach for a high number of requests (Fig. 4(e)). Figure 4(f) depicts the time complexity of the proposed approach in different patterns. It takes relatively higher time to converge in the chain pattern than the other patterns in Fig. 4(f).

7 Conclusion

We propose the TempCP-Net framework to represent the long-term economic model of an IaaS provider. The proposed model allows a provider to apply qualitative business strategies in composing consumer requests. It is a more natural and simplified composition process than the quantitative approach which focuses the composition on complex resource levels, i.e., operation cost calculation and scheduling. We propose a heuristic based sequential optimization algorithm that does not require the history of different input patterns and corresponding composition decisions. Hence, it will be mostly helpful to start-up IaaS providers. Experimental results show that the proposed approach is applicable in runtime and significantly faster than global DP based approaches. The accuracy of the solution is also acceptable in different input patterns. In the future work, we want to explore the TempCP-Net with the root causes of their different business strategies. We will find correlations between the TempCP-Net and different market factors, such as peer competitions, the supply and demand of the services, and the reputation.

Acknowledgements. This research was made possible by NPRP 7-481-1-088 grant from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

1. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Proceedings of FOCS, pp. 459–468. IEEE (2006)
2. Armbrust, M., Fox, A., Griffith, R.: Above the clouds: a berkeley view of cloud computing. Technical report, University of California, Berkeley (2009)
3. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
4. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: Cp-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. J. Artif. Intell. Res. **21**, 135–191 (2004)
5. Goiri, Í., Guitart, J., Torres, J.: Economic model of a cloud provider operating in a federated cloud. Inf. Syst. Front. **14**, 827–843 (2012)
6. Inc, G.: Compute engine features (2015). https://cloud.google.com
7. Jiang, W., Lee, D., Hu, S.: Large-scale longitudinal analysis of soap-based and restful web services. In: Proceedings of ICWS, pp. 218–225 (2012)
8. Kimes, S.E., Thompson, G.M.: Restaurant revenue management: determining the best table mix. Decis. Sci. **35**(3), 371–392 (2004)

9. Lim, H., Widdows, R., Park, J.: M-loyalty: winning strategies for mobile carriers. J. Consum. Mark. **23**(4), 208–218 (2006)
10. Mistry, S., Bouguettaya, A., Dong, H., Qin, A.K.: Metaheuristic optimization for long-term iaas service composition. IEEE Trans. Serv. Comput. **PP**(99), 1 (2016)
11. Mistry, S., Bouguettaya, A., Dong, H., Qin, A.K.: Predicting dynamic requests behavior in long-term iaas service composition. In: Proceedings of ICWS. IEEE (2015)
12. Reiss, C., Wilkes, J., Hellerstein, J.L.: Google cluster-usage traces: format + schema. Google Inc., Mountain View, CA, USA, Technical report (2011)
13. Santhanam, G.R., Basu, S., Honavar, V.: Web service substitution based on preferences over non-functional attributes. In: Proceedings of SCC, pp. 210–217 (2009)
14. Vien, N.A., Toussaint, M.: Hierarchical monte-carlo planning. In: Proceedings of AAAI, pp. 3613–3619 (2015)
15. Wang, H., Shao, S., Zhou, X., Wan, C., Bouguettaya, A.: Preference recommendation for personalized search. Knowl.-Based Syst. **100**, 124–136 (2016)
16. Wang, H., Zhang, J., Sun, W., Song, H., Guo, G., Zhou, X.: WCP-Nets: a weighted extension to CP-Nets for web service selection. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) ICSOC 2012. LNCS, pp. 298–312. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34321-6_20
17. Wu, L., Kumar Garg, S., Buyya, R.: Sla-based admission control for a software-as-a-service provider in cloud computing environments. J. Comput. Syst. Sci. **78**(5), 1280–1299 (2012)
18. Ye, Z., Bouguettaya, A., Zhou, X.: QoS-aware cloud service composition based on economic models. In: Proceedings of ICSOC, pp. 111–126 (2012)