云计算中TSP问题求解服务的定价机制

曾栩鸿 曾国荪
(同济大学计算机科学与技术系 上海 200092)
(国家高性能计算机工程技术中心同济分中心 上海 201804)

摘 要 旅行商问题(TSP)是一个典型的路径优化问题,在城市交通规划、物流运输、通信网络设置等领域都存在类似的问题和应用。但是,TSP问题的求解是NP难的,当问题规模很大时,必须借助大规模并行计算环境,例如云计算平台,以较大的计算开销来获得可行解。以TSP问题为具体实例,研究云计算服务的定价机制。一般情况下,定价机制要满足公平、灵活、动态、自适应。从公平合理角度来看,影响计算服务定价的因素主要有两方面:一是求解问题的难度,包括计算时间复杂性、空间复杂性、输入输出数据规模等;二是求解服务质量,即服务契约,包括可以作为服务等级协定指标的求解精度、响应时间、资源要求等。由此,提出了一种新的云计算中的服务定价机制:CloudPricing。该机制给出了服务定价的一般和具体原则,并给出了相应的定价公式。针对TSP问题求解,进行了具体的定价实例分析,这对云计算中NP难问题求解服务的定价有参考意义。

关键词 云计算,TSP问题,求解服务,定价机制,服务等级协定

Pricing Mechanism of TSP Solving Service in Cloud Computing

ZENG Xu-hong ZENG Guo-sun
(Department of Computer Science and Technology, Tongji University, Shanghai 200092, China)
(Tongji Branch, National Engineering & Technology Center of High Performance Computer, Shanghai 201804, China)

Abstract The traveling salesman problem(TSP) is a typical path optimization problem which has similar problems and applications in urban transportation planning, logistic transport and communication network settings. However, TSP is a NP hard problem. When problem scale is very large, large scale parallel computing environment such as cloud computing platform is needed. In this paper, we illustrated cloud service pricing mechanism with TSP. Generally, pricing mechanism should be fair, flexible, dynamic and flexible. To be fair and reasonable, there are two main aspects to be considered when pricing a service. One is the difficulty of solving the problem including time complexity, space complexity and quantity of data the application input and output. The other is the quality of service including precision of the result, response time and whether the service is provided in peak time or not which can be served for Service Level Agreement between service provider and customer. Next, we proposed principles of pricing the service and pricing formula. Finally, a case study aiming at pricing solving TSP service was given, which has a reference value for pricing NP hard problem in cloud computing environment.

Keywords Cloud computing, TSP problem, Solving service, Pricing mechanism, SLA

1 引言

随着互联网的发展,网络计算、P2P计算、网格计算、云计算的推广应用,共享和使用计算资源变得越来越普遍。人们越来越希望有一种如同通过自来水管获取水、通过电线获取电源一样的快捷的计算环境。在效用计算、网格计算等的基础上发展起来的云计算顺应了这种趋势,并在工业界和学术界得到了广泛的认可。所谓云计算就是一种计算平台或者应用模式,云中集聚大量服务器或应用软件或存储设备,用户通过访问云,就可以方便地获取自己需要的服务,如数据访问、特定计算等。

如何为云计算服务定价,正成为日益重要的问题。恰当合理的定价策略不仅有助于云服务提供商占领市场,帮助其实现企业价值,更能有效地促进供求双方的良性互动,推动云计算产业的长远发展。现在云服务提供商普遍使用简单的固定定价机制对用户收费。例如,Amazon的云平台按计算使用量、存储量和数据转移量固定地进行收费。其不足主要体现在:固定定价机制往往不能满足不同用户的千差万别的需

到稿日期:2011-01-22 返修日期:2011-04-17 本文受863项目(2007AA01Z425),973计划课题(2007CB316502),国家自然基金项目(90718015),NSFC-微软亚洲研究院联合资助项目(60970155),教育部博士点基金项目(20090072110035),上海市优秀学科带头人计划项目(10XD1404400),高效能服务器和存储技术国家重点实验室开放基金项目(2009HSSA06)资助。

曾栩鸿(1987-),男,硕士生,主要研究领域为云计算、并行计算;曾国荪(1964-),博士,教授,博士生导师,主要研究领域为异构计算、并行计算、信息安全。

求，不同的用户对价格的敏感程度不一样。另外，现有的定价机制一般都从服务提供商的角度为云服务定价，并不能客观、公平、公正地反映云服务的真实价值。

针对现行定价机制的缺点，国外学者开展了大量研究。针对效用计算需求不确定、周期短等特点，Paleologo 提出了 Price-At-Risk 的定价策略[8]。Buyya 等人提出了云服务市场的概念，构建了交易服务的云服务市场模型，用户、中间商、服务提供商基于 SLA 交易云服务[11]。Yeo 等人提出了自动计量的价格策略，在高峰期收取较高的价格，在低峰期收取较低的价格，使得愿意支付较高价格的用户可以在高峰期仍取得好的服务，而其他用户可以在低峰期以较低的价格取得云服务[13]。Henzinger 对云计算环境进行建模，根据每次作业的数据传输、计算、持续时间来考虑价格，提出了启发式调度算法[12]。Mihailescu 基于联邦云环境，提出了一种防策略的动态定价机制，通过实验证明，动态定价机制比固定定价策略在效率等方面都更有优势[5]。这些定价机制往往只从服务提供商的角度考虑定价，并不能完全满足服务定价对公平、公正的要求。

传统高性能程序往往只能达到 15%到 20%的效率值，而利用云计算环境下数据中心强大的计算能力可使得这些程序有更高的效率。云计算为 NP 问题的求解提供了新的环境，用户甚至可以使用蛮力法在云计算环境下求解 NP 问题。旅行商问题 TSP 是一个 NP 问题，TSP 问题可以应用在现实生活中的很多地方，比如电路板钻孔进度的安排、基因测序和机器人控制等、邮路问题、装配线上螺帽问题和产品的生产安排问题等。

本文通过对 TSP 问题的模型及其特征的分析，提出基于问题规模和服务等级协定(SLA)的服务定价的一般和具体原则，并在这些原则的基础上从公平合理的角度提出一种云计算环境下的定价机制。

2 TSP 问题的描述

TSP 问题用文字可简单描述为：假设有多个城市，并已知城市之间的旅行代价，要求寻找经过每个城市一次且仅一次而最终回到起始城市旅行代价最小的路径。

TSP 问题可用图论有关概念来描述。在给出 TSP 问题的准确定义之前，先给出图论中的相关概念。

定义 1(图)　一个图可定义为一个二元组 $G=(V,E)$，其中 $V=\{v_1,v_2,\cdots,v_n\}$ 是一个有限非空集合，称为 G 的顶点集，n 为 G 的顶点数，记 $|V|=n$；$E=\{e_1,e_2,\cdots,e_m\}$ 是由 V 中的顶点组成的序对构成集合，称为 G 的边集，即 $e_j\in E,1\leqslant j\leqslant m,e_j\in V\times V$，$m$ 为边数，记 $|E|=m$。

定义 2(通路)　图 $G=(V,E)$ 的一条通路 path 是指一个有限非空序列 path$=v_0e_1v_1e_2v_2\cdots e_kv_k$，它的项交替为顶点和边，$v_i\in V,e_i\in E$，使得对于 $1\leqslant i\leqslant k$，e_i 关联于 v_{i-1} 和 v_i。顶点 v_0 和 v_k 分别称为 path 的起点和终点。

定义 3(连通图)　在图 $G=(V,E)$ 中，如果存在从顶点 v_i 到顶点 v_j 的一条通路 path，$v_i,v_j\in V$，则称 v_i 和 v_j 是连通的。如果 G 中任意两顶点都是连通的，那么 G 被称作连通图，否则称为非连通图。

定义 4(回路)　图 $G=(V,E)$ 的一条回路 path 是指一个有限非空序列 path$=v_0e_1v_1e_2v_2\cdots e_kv_k$，它的项交替为顶点和边，$v_i\in V,e_i\in E$，使得对于 $1\leqslant i\leqslant k$，e_i 关联于 v_{i-1} 和 v_i。顶点 v_0 和 v_k 分别称为 path 的起点和终点，且满足 $v_k=v_0$。

定义 5(关联)　设 $e_k=(v_i,v_j)$ 是图 $G=(V,E)$ 中的一条边，则称 e_k 与 $v_i(v_j)$ 关联。

定义 6(度)　在图 $G=(V,E)$ 中，一个顶点 v 的度是与它相关联的边的条数。

定义 7(赋权图)　赋权图可定义为一个三元组 $G=(V,E,W)$，其中 $V=\{v_1,v_2,\cdots,v_n\}$ 是 G 的顶点集，$E=\{e_1,e_2,\cdots,e_m\}$ 是 G 的边集，$1\leqslant j\leqslant m$，m 为 G 的边数，记 $|G|=m$，$W=\{w_1,w_2,\cdots,w_m\}$ 称为 G 的赋权集，非负实数 $w_j\in W$ 为对应于 $e_j\in E$ 上的权值。

赋权图在实际问题中非常有用。根据不同的实际情况，权数的含义可以各不相同。例如，可用权数代表两地之间的实际距离或行车时间，也可用权数代表某工序所需的加工时间等。

定义 8(欧拉图)　图 $G=(V,E)$ 的一条回路 path$=v_0e_1v_1e_2v_2\cdots e_kv_k$ 中，$v_k=v_0$，若图 G 中存在能够遍历完所有的边而没有重复的回路，即对于 path 中任意两条边 e_i 与 e_j，$e_i\neq e_j(i\neq j)$，则 G 为欧拉图。

定义 9(哈密顿回路与哈密顿图)　若图 $G=(V,E)$ 中存在能够遍历完所有顶点而没有重复的回路 path$=v_0e_1v_1e_2v_2\cdots e_kv_k(v_k=v_0)$，即对于 path 中任意两点 v_i 与 v_j，$v_i\neq v_j(i\neq j$ 且 i、j 都不等于 $k)$，则称 path 为哈密顿回路，图 G 为哈密顿图。

如果用图论来描述 TSP 问题，就是已知赋权图 $G=(V,E,W)$，找出总权值最小的哈密顿回路。其中 $V=\{v_1,v_2,\cdots,v_n\}$ 为顶点集，这里表示 n 个城市的集合；$E=\{e_1,e_2,\cdots,e_m\}$ 为各顶点相互连接组成的边集。每一条边 e_i 都存在与之对应的权值 w_i，实际应用中 w_i 可以表示距离、费用、时间、油量等。

定义 10(TSP 问题)　假设 $G=(V,E,W)$ 为赋权连通图，$V=\{v_1,v_2,\cdots,v_n\}$ 表示所有城市的集合，$E=\{e_1,e_2,\cdots,e_m\}$ 表示城市之间边的集合，$W=\{w_1,w_2,\cdots,w_m\}$ 表示城市与城市之间旅行代价的集合。若对于所有城市的一个访问顺序为 $T=v_{t1}e_{t1}v_{t2}e_{t2}v_{t3}\cdots v_{ti}e_{ti}\cdots v_{tn}e_{tn}v_{t1}$，其中 $ti\in I(I=\{1,2,3,\cdots,n\})$，$w_{ti}$ 为边 e_{ti} 对应的权值，$|e_{ti}|=w_{ti}$，则旅行商问题的数学模型为在图 G 中找出一个满足代价最小的哈密顿回路。

$$\min \text{Cost}=\sum w_{ti},1\leqslant i\leqslant n$$

例 1　下面给出 TSP 问题的一个实例。假设 $G=(V,E,W)$ 为赋权连通图，$V=\{v_1,v_2,\cdots,v_{10}\}$ 表示所有城市的集合，E 表示城市之间边的集合。下面可以看到这是一个完全图，每一对城市之间都有直接的路径相连，其中 $e_{i,j}$ 表示 v_i 到 v_j 的边。W 表示城市与城市之间旅行代价的集合，$w_{i,j}$ 表示 v_i 到 v_j 的边的权值，W 表示成邻接矩阵，如图 1 所示。

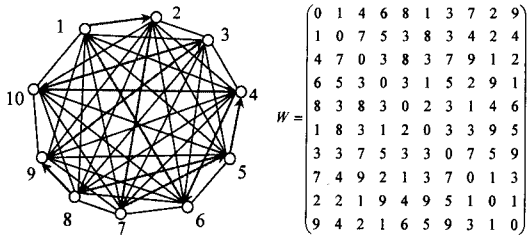

图 1　TSP 问题的例子

通过计算可求得对于所有城市的最佳访问顺序为 $T=v_1$

$e_{1,2}\ v_2\ e_{2,7}\ v_7\ e_{7,5}\ v_5\ e_{5,8}\ v_8\ e_{8,9}\ v_9\ e_{9,3}\ v_3\ e_{3,10}\ v_{10}\ e_{10,4}\ v_4\ e_{4,6}\ v_6\ e_{6,1}\ v_1$（见图1），最后得到的最短周游代价为15。

3 TSP问题的求解方法

TSP问题常用的求解方法有枚举法、分枝定界法、线形规划法、动态规划法等，其求解时间为指数复杂度，问题规模很大时难以获得最优解，因此就产生了巡回路径构造算法、巡回路径优化算法和智能算法等近似算法。前者算法简单，计算量小，大多数情况下求得的解能满足要求。下面分析TSP问题的复杂性，并给出解的正确性和精确性的定义。

3.1 TSP问题的复杂性分析

3.1.1 计算复杂性分析

假设TSP问题中的图$G=(V,E,W)$，$V=\{v_1,v_2,\cdots,v_n\}$表示所有城市的集合，$E=\{e_1,e_2,\cdots,e_m\}$表示城市之间边的集合，$W=\{w_1,w_2,\cdots,w_m\}$表示城市与城市之间旅行代价的集合。其中n表示城市的数目，m表示边数。根据图的不同形状和结构，不同的计算复杂性讨论如下：

(1) $m<(n-1)$，这种情况下图G不是连通图，不满足TSP问题的要求，不需要考虑。

(2) $(n-1)\leqslant m<n(n-1)/2$，说明存在城市和城市不能直接到达的情况，假设$v_1,v_2,\cdots,v_n$的度分别为$d_1,d_2,\cdots,d_n$，因为不管遍历的顺序如何，都需要对每个城市都遍历一遍，都需要在$\sum d_i(1\leqslant i\leqslant n)$条可能的回路中选择最短的一条，对每种可行路径，都需要做$n-1$次加法，所以它的计算复杂性为$P_C(G)=(n-1)\sum d_i(1\leqslant i\leqslant n)$个加法操作。

(3) $m=n(n-1)/2$，即图G为完全图时，每一对城市之间都有一条路径，要找到一条遍历所有城市的回路，需要从$(n-1)!$种情况中找出最短路，对每种可行路径，都需要做$n-1$次加法，因此其计算复杂性为$P_C(G)=(n-1)(n-1)!/C_0$个加法操作。

(4) $m>n(n-1)/2$，说明城市和城市之间有重边存在，这种情况发生在城市和城市之间有不止一条路径或者一种交通方式的时候。假设$v_1,v_2,\cdots,v_n$的度分别为$d_1,d_2,\cdots,d_n$，因为不管遍历的顺序如何，都同样需要对每个城市都遍历一遍，都需要在$\sum d_i(1\leqslant i\leqslant n)$条可能的回路中选出最短的一条，对每种可行路径，都需要做$n-1$次加法，所以它的计算复杂性为$P_C(G)=(n-1)\sum d_i(1\leqslant i\leqslant n)$个加法操作。

3.1.2 空间复杂性分析

假设TSP问题中的图$G=(V,E,W)$，$V=\{v_1,v_2,\cdots,v_n\}$表示所有城市的集合，$E=\{e_1,e_2,\cdots,e_m\}$表示城市之间边的集合，$W=\{w_1,w_2,\cdots,w_m\}$表示城市与城市之间旅行代价的集合。其中n表示城市的数目，m表示边的数目。对于TSP问题中的图G有多种存储方法，利用不同的数据结构，每种存储方法的空间复杂性则不同：

(1)邻接矩阵存储。当城市和城市之间没有重边的时候，可以用邻接矩阵来存储图G，这时需要n^2个单元来存储城市和城市之间是否有边、其权值为多少等信息，所以它的时间复杂性$P_T(G)=n^2$。

(2)邻接表存储。邻接表会为每个顶点都建立一个单链表，而且每条边都会在邻接表中出现两次。因此它需要$n+2m$个单元来存储所有信息，时间复杂性$P_T(G)=n+2m$。

3.1.3 NP完全问题的证明

结论1 若TSP问题中的图G为完全图，则求解TSP问题的解是NP完全的。

Richard M. Karp在1972年证明了哈密顿环问题是NP完全问题，TSP问题是哈密顿环问题的特殊情况，这也意味着TSP问题是NP难问题，第一次从科学的角度解释了寻找最优周游路线的计算复杂度。

3.2 TSP问题解的正确性和精确性定义

因为TSP问题是NP难问题，所以难于求得最优解，实际应用中往往只好求次优解。为了分析解的精确性，我们首先给出以下概念。

定义11(可行解) 假设TSP问题对应的图为$G=(V,E,W)$，存在一个解为$S=v_{t1}e_{t1}v_{t2}e_{t2}v_{t3}\cdots v_{ti}e_{ti}\cdots v_{tn}e_{tn}v_{t1}$，$S$中所有的顶点组成的集合$V_S=\{v_{t1},v_{t2},\cdots,v_{ti},\cdots,v_{tn}\}\subseteq V$，$S$中所有的边组成的集合$E_S=\{e_{t1},e_{t2},\cdots,e_{ti},\cdots,e_{tn}\}\subseteq E$。若$S$满足：(1)$S$中包括$V$中所有顶点，即$V_S=V$；(2)$S$中除起点外没有重复的顶点，即对$V_S$中任意两点$v_{ti}$与$v_{tj}$，$v_{ti}\neq v_{tj}$，$i\neq j$且$i,j$都不等于1，则$S$为图$G$的哈密顿回路，称解$S$为可行解。

定义12(错误解) 假设TSP问题对应的图为$G=(V,E,W)$，存在一个解为$S=v_{t1}e_{t1}v_{t2}e_{t2}v_{t3}\cdots v_{ti}e_{ti}\cdots v_{tn}e_{tn}v_{t1}$，$S$中所有的顶点组成的集合为$V_S=\{v_{t1},v_{t2},\cdots,v_{ti},\cdots,v_{tn}\}$。若$S$满足以下任意条件之一：(1)$S$中不包括$V$中所有顶点，即$V_S\subset V$且$V_S\neq V$；(2)$S$中除起点外有重复的顶点，即存在$V_S$中的两点$v_{ti}$与$v_{tj}$，$v_{ti}=v_{tj}$，$i\neq j$且$i,j$都不等于1，则称解是错误解。

从错误解的定义可知，若解中没有周游到某一城市，或对某一城市遍历了一次以上，则此解为错误解。

结论2 可行解一定不是错误解，错误解一定不是可行解。

证明：由可行解和错误解的定义显然可以得证。

定义13(最优解) 假设TSP问题对应的图为$G=(V,E,W)$，若存在一个解$S=v_{t1}e_{t1}v_{t2}e_{t2}v_{t3}\cdots v_{ti}e_{ti}\cdots v_{tn}e_{tn}v_{t1}$为可行解，且满足周游代价Cost最小，即$\mathrm{Cost}_{\min}=\min\sum w_{ti}$，$1\leqslant i\leqslant n$，$|e_i|=w_i\in W$，则称$S$为最优解。

在例1中，TSP问题的最优解为1-2-7-5-8-9-3-10-4-6-1，周游代价为15。

定义14(最差解) 假设TSP问题对应的图为$G=(V,E,W)$，若存在一个解$S=v_{t1}e_{t1}v_{t2}e_{t2}v_{t3}\cdots v_{ti}e_{ti}\cdots v_{tn}e_{tn}v_{t1}$为可行解，且满足周游代价Cost最大，即$\mathrm{Cost}_{\max}=\max\sum w_{ti}$，$1\leqslant i\leqslant n$，$|e_i|=w_i\in W$，则称$S$为最差解。

在例1中，TSP问题的最优解为1-4-9-6-2-3-8-7-10-5-1，周游代价为78。

定义15(解的精度) 假设TSP问题的最优解的周游代价为$\mathrm{Cost}_{\min}$，任一可行解的周游代价为Cost_{fea}，则称该可行解的精度定义为$\rho=\mathrm{Cost}_{\min}/\mathrm{Cost}_{fea}$。由最差解的定义知最差解的精度为$\rho_{\mathrm{worst}}=\mathrm{Cost}_{\min}/\mathrm{Cost}_{\max}$。特别地，规定错误解的精度为0。

结论3 最差解的精度≤可行解的精度≤最优解的精度。

证明：由解的精度和最差解、最优解、可行解的定义显然得证。

4 解的正确性检测

用户从服务提供商得到问题的解后，只有当这个解正确

并且精度等方面满足用户的期望时，用户才需要为此次服务付费，因此需要验证解的正确性。假设用户 C 提出计算请求，云服务提供商 R 提供服务。C 向 R 提交输入参数 x（x 中包含求解问题所需的所有信息），R 计算后将得到的解 $f_R(x)$ 返回给用户，C 需要正确解 $f_C(x)$ 来进行验证。可能出现以下 4 种情况。

(1)C 知道 $f_C(x)$，R 提供 $f_R(x)$。

对于某一测试样例 x，C 可以通过自己计算或从其它可信或已经验证过的服务提供商得到样例的最优解 $f_C(x)$。例如，在 TSP 问题中，如果城市的数目不多，C 可以自己先算出最佳周游路线，然后用小规模的 TSP 问题作为测试样例，为求解大规模 TSP 问题时选择服务提供商提供依据。C 向 R 提交一系列测试样例后，R 计算后向 C 返回服务解 $f_R(x)$。通过比较 $f_R(x)$ 与 $f_C(x)$，用户 C 可以验证 R 是否提供了正确解。

(2)C 知道 $f_C(x)$，R 不提供 $f_R(x)$。

在某些情况下，服务提供商 R 可能不愿意直接提供测试样例的实际解 $f_R(x)$。例如，R 提供的是收费服务，但 R 不能确定 C 是否会在它的应用程序中使用免费验证服务。在这种情况下，R 可能会提供哈希过的结果 $g(f_R(x))$，这里 g 是由 C、R 共同约定的不可逆哈希函数。C 无法用 $g(f_R(x))$ 求解出 $f_R(x)$，因为 g 是不可逆的。MD5 等哈希函数适用于这种功能验证，满足一对一映射和不可逆性。在这种情况下，用户可以用已知的 $f_C(x)$ 和函数 g 计算出 $g(f_C(x))$，并比较 $g(f_C(x))$ 与 $g(f_R(x))$，用户 C 同样可以验证 R 是否提供了正确解。

(3)C 不知道 $f_C(x)$，R 提供 $f_R(x)$。

C 可能不能提前知道 $f_C(x)$，在这种情况下，C 仍可以用间接的方法验证 R 的服务。例如，C 需要从 R 得到天气预报的服务，C 可以记录 R 的预测结果，然后观测实际的天气情况后评估 R 的预测结果的准确性。另外，C 也可以向几个服务提供商提交测试样例，然后比较它们得到的服务解是否一致。

(4)C 不知道 $f_C(x)$，R 不提供 $f_R(x)$。

这是功能验证中最困难的一种情况，即 C 不知道 $f_C(x)$，R 仅提供经过哈希函数处理过的解。我们仍可以将这种情况转化为情况(2)和(3)中讨论的情况，做法是由 C 向几个服务提供商提交测试样例，且这几个服务提供商使用的是一样的不可逆哈希函数 g，这样 C 可以比较它们得到的 $g(f_R(x))$。

以上讨论了功能验证时可能遇到的 4 种情况，功能验证提供了达成服务协定前的一种检测方法。当然，功能验证不能完全解决现实情况下客户和服务提供商的信任问题。例如，R 可能在功能验证成功后不为用户提供足够精确的服务。为了解决这个问题，需要引入服务评级机制等方法，这里不做进一步讨论。

5 云计算中的服务定价机制：CloudPricing

5.1 一般定价原则

云计算下任何可用的计算资源都以服务的形态存在，向用户提供尽力而为的多租赁的服务，用户按使用付费。服务提供商普遍采用即付即用的计费模式，因为价格可以调节服务的供求关系，从而影响用户和云服务提供商，如何为服务设定合理的价格变得越来越重要。通过设定合理的价格，服务提供商能吸引足够的用户以达到利润目标，同时更有效地提供服务来满足用户的需求。一般情况下，定价机制要公正、公平、公开，在云计算环境下还需要灵活、动态、自适应。如何设定弹性、公平、动态、可适应的定价机制，使不同需求的用户有不同的选择，对云计算商业化的健康发展有着重要的意义。

5.2 具体定价原则

从公平合理角度来看，影响计算服务定价的因素主要有两方面：一是求解问题的难度，包括计算时间复杂性、空间复杂性、输入输出数据规模等；二是求解服务质量，即服务契约，包括求解精度、响应时间、资源要求等。由此下面给出服务定价的具体原则。

原则 1：错误解定价为 0。

待求解的 TSP 问题中，若云服务提供商提供的解中没有遍历到所有城市或对某一城市遍历了多次等，则将这个解定义为错误解，用户不应为错误解付费，所以将错误解定价为 0。

原则 2：问题规模越大，复杂性越高，求解代价越高，定价越高。

待求解的 TSP 问题中城市的数目越多，问题规模越大，求解代价越高，因此定价越高。时间复杂性和空间复杂性越高，则计算此 TSP 的代价越高，定价越高。

原则 3：解的精度越高，定价越高。

待求解的 TSP 问题中，若云服务提供商提供的解是可行解且总代价越低，说明越接近最优解，精度越高，则定价越高。

原则 4：输入输出的数据量越大，定价越高。

在许多高性能计算程序中，往往需要输入输出大量的数据，对网络带宽有较高的要求。输入输出的数据量越大，占用的带宽越多，定价越高。由于输入输出的数据量都比较小，因此可以忽略 TSP 问题中输入输出的数据量对价格的影响。

原则 5：响应时间越短，定价越高。

从服务等级协定(SLA)的角度考虑，当服务未能完全满足用户需求时，可采取惩罚定价的方法。响应时间即用户从提交问题到从云服务提供商得到问题解的总时间。若对于同一问题规模的 TSP 问题，云服务提供商 A 和云服务提供商 B 提供的解都是可行解，且解的精度相同，但 A 的响应时间比 B 短且 A，B 的响应时间都超过用户期望的时间阈值 t_E 时，则 A 的定价比 B 高。若其它条件一样，且 A，B 的响应时间都小于用户期望的时间阈值 t_E，则 A 与 B 服务的价格相同。

原则 6：在用户高峰期收取较高的价格，在用户低谷期收取较低的价格。

在高峰期和低谷期收取不同的价格，可以使得一部分原本打算在高峰期请求服务的用户改变策略，在价格更经济的低谷期请求服务，使得云服务尽可能地得到利用。高峰期较高的价格使得愿意支付较高价格的用户可以随时得到服务，从而保证负载不致于过大，且服务提供商又能获得更多的利润，而用户可以根据自己的实际需要选择合适的时间段请求服务。

其它原则，如不同地区电费差异、长期服务和一次性服务的差异、故障时间、灾难恢复和正常服务时间的奖励对价格的影响，在此不详述。

5.3 CloudPricing 定价的计算公式

假设 TSP 问题中的图 $G=(V,E,W)$，n 表示城市的数

目，m 表示边的数目。$P_C(G)$为该问题的时间复杂性，$P_S(G)$为该问题的空间复杂性，$P_D(G)$为该问题的输入输出数据量。单位计算复杂性定价为 c，单位空间复杂性定价为 s。定义解是否为可行解的标志为 b_{fea}，当解为可行解时，$b_{fea}=1$；解为错误解时，$b_{fea}=0$。引入这个标志相当于采用了经济学中的保证定价法，即保证用户得到可行解后，用户才需要对此次服务付款。如果得到了错误解，则不需对此次服务付款，对用户来说是一个非常有利的保险。定义区分用户是在高峰期还是低谷期请求服务的因子为 b_{peak}。当用户在高峰期请求服务时，$b_{peak}=2$；用户在低谷期请求服务时，$b_{peak}=1$。最差解的精度为 ρ_{worst} $(0\leqslant\rho_{worst}\leqslant 1)$，解的精度为 ρ。为保证云服务提供商的利益，构造单调函数 $g(\rho)=1-0.5(1-\rho)/(1/\rho_{worst}-1)$，$\rho_{worst}\leqslant\rho\leqslant 1$ 使得在取得最差解时 $g(\rho)=0.5$，在取得最优解时 $g(\rho)=1$。此次服务的响应时间为 t，t_E 为用户可以期望的响应时间阈值。对于时间复杂性、空间复杂性等因素在定价时的权重的权衡，引入 λ_1，λ_2，λ_3 等比例因子，可以根据实际需要调节。对于响应时间 t 超过用户期望的时间阈值 t_E 的情况，引入因子 λ_4，当 $t>t_E$ 时，t 越大，价格越低。

综合以上讨论，我们提出基于服务质量的定价函数如下：

$$P(G,c,s,d,t)=\begin{cases} b_{fea}b_{peak}(\lambda_1 cP_C(G)+\lambda_2 sP_S(G)+\lambda_3 dP_D(G))\\ \quad(1-0.5\dfrac{1-\rho}{1/\rho_{worst}-\rho}),t\leqslant t_E\\ b_{fea}b_{peak}(\lambda_1 cP_C(G)+\lambda_2 sP_S(G)+\lambda_3 dP_D(G)-\lambda_4\dfrac{t}{t_E})\\ \quad(1-0.5\dfrac{1-\rho}{1/\rho_{worst}-\rho}),t>t_E \end{cases} \quad (1)$$

从 Amazon 等云服务提供商收取费用时的实际情况来看，一般 λ_1 要大于 λ_2，即对计算的收费比对存储的收费高。从这个式子中可以看出，当解为错误解，即 $b_{fea}=0$ 时，定价为 0，满足原则 1。时间复杂度和空间复杂度越高，即 $P_T(G)$ 和 $P_S(G)$ 越大时，定价也相应越高，满足原则 2。解的精度越高，即 ρ 越高，定价也越高，满足原则 3。输入输出的数据量越大，即 $P_D(G)$ 越大，定价越高，满足原则 4。由于惩罚机制的存在，当响应时间超过时间阈值，即 $t\geqslant t_E$ 时，超过的时间越多，此次服务的定价越低，满足原则 5。当高峰期时，用户被收取较高的费用，反之则被收取较低费用，满足原则 6。综上，这个定价函数在反映问题本质即时间复杂性和空间复杂性的基础上，较好地反映了求解问题难度和求解服务质量对价格的影响，即求解难度和问题服务质量越高时，用户被收取较高的费用；服务质量特别差时，用户甚至不需对此次服务付费。

5.4 具体定价举例

对于第 2 节中的例 1，假设单位计算复杂度的单价为 $c=1$，借鉴云服务提供商 Amazon 等在提供云服务时存储对价格的影响相对较小的考虑，假定空间复杂度的单价为 $s=0.2$。用户期望的响应时间为 $t_E=1$s，$\lambda_1=\lambda_2=1$，由于 TSP 问题中数据输入输出量都较小，设定 $\lambda_3=0$，即输入输出数据量对价格并不产生影响。TSP 问题的求解往往对响应时间要求不高，因此设定 $\lambda_4=0.05$。下面给出按照式(1)计算的具体定价，如表 1 所列。

从表中可以看出：

(1) $b=0$，即在方案 1 中，当解是错误解时，服务提供商返回给用户的周游顺序为 1-4-3-8-2-5-9-10-7-1，没有遍历到城市 6，因此 $b=0$，定价为 0，满足定价原则 1，保证用户使用云计算服务得到的是正确解。

(2)比较方案 2、5、7 和方案 3、6、9，解的精度 ρ_{fea} 越高，价格越高，满足定价原则 3。其中方案 2、3 为最优解，价格最高。

(3)比较方案 7、8，当响应时间在时间阈值 $t_E=1$s 内时，价格不变。比较方案 2、4、10 可以发现，当响应时间超过时间阈值 t_E 时，由于惩罚定价机制的存在，响应时间越长，价格越低，满足定价原则 5。当响应时间过长时，服务解对用户来说已经没有意义，甚至可以不用对此次服务付费。

(4)比较方案 2、3 和方案 8、9，在高峰期用户被收取较高的价格，在低谷期被收取较低的价格，满足定价原则 6。通过差异化高峰期和低谷期的价格，使得用户可以根据自己的实际需要选择获得服务的时间和相应的服务质量，满足不同用户的不同需求。在实际定价时，可以根据请求计算服务的用户数量和系统负载的供求关系来实现更复杂的动态调整价格的机制。

表 1 不同方案的定价

	解(周游路线)	b_{fea}	b_{peak}	t	ρ_{fea}	P
方案 1	1-4-3-8-2-5-9-10-7-1	0	-	-	-	0
方案 2	1-2-7-5-8-9-3-10-4-6-1	1	1	0.4	1	3.46592
方案 3	1-2-7-5-8-9-3-10-4-6-1	1	2	0.5	1	6.93184
方案 4	1-2-7-5-8-9-3-10-4-6-1	1	1	2	1	2.46592
方案 5	1-3-2-5-6-4-8-9-7-10-1	1	1	0.3	0.3488	3.23331
方案 6	1-3-2-5-6-4-8-9-7-10-1	1	2	0.3	0.3488	6.46662
方案 7	1-4-9-6-2-3-8-7-10-5-1	1	1	0.2	0.1923	3.18642
方案 8	1-4-9-6-2-3-8-7-10-5-1	1	1	0.4	0.1923	3.18642
方案 9	1-4-9-6-2-3-8-7-10-5-1	1	2	0.6	0.1923	6.37284
方案 10	1-2-7-5-8-9-3-10-4-6-1	1	1	3	1	1.96592

通过比较这些方案，可以看出当采用我们的定价机制时，问题本身的复杂度、用户对服务质量的要求以及云服务提供商的系统负载对价格都能产生一定影响。至于具体哪部分的影响更大，可以根据实际情况调节参数 λ_1，λ_2，λ_3，λ_4。云服务提供商要获得相应的利润，必须提供 SLA 以保证服务质量。

结束语 本文从 TSP 问题出发，讨论了云计算环境下怎样用问题规模和服务质量制定合理的定价机制的方法。在分析了 TSP 问题的复杂性和解的特征后，提出了验证解的正确性的方法，并对服务解进行了功能验证。最后，我们给出了综合考虑问题规模和服务质量等因素的定价公式，并通过实例分析，证明公式满足之前提出的各项定价原则，在刻画问题本质的基础上从一个新的角度讨论了服务质量对价格的影响。本文从第三方的角度制定公平、灵活、动态、可适应的定价机制，与其它文献相比，能在保证云服务提供商的利润的同时使服务满足服务等级协定(SLA)。进一步的工作将深入分析影响服务质量的其它因素和已有因素如何更合理的使用，如高峰期的价格可以根据此时实际使用服务的用户的数量和系统负载进行更复杂的动态定价，使定价机制能够更好地满足现实生活中云计算的定价要求。下一步还将把本定价机制在实际的云计算平台(如 Hadoop)中实现，并检验机制的实际效果。本文通过 TSP 问题提出了定价机制，在其他更加一般的问题上应用本机制也是本文进一步的工作。

参 考 文 献

[1] 陈康,郑纬民. 云计算:系统实例与研究现状[J]. 软件学报,2009,20(5):1337-1348

[2] Buyya R, Yeo C S. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility[J]. Future Generation Computer Systems, 2009(25): 599-616

[3] Lee Y, Wang C. Profit-driven service request scheduling in clouds[C]// IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. 2010: 15-24

[4] Stober J, Neumann D. Market-based pricing in grids: on strategic manipulation and computational cost[J]. European Journal of Operational Research, 2010, 203: 464-475

[5] Mihailescu M, Teo Y M. On economic and computational-efficient resource pricing in large distributed systems[C]// IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. 2010: 838-843

[6] Saure D, Sheopuri A. Time-of-use pricing policies for offering cloud computing as a service[C]// IEEE International Conference on Service Operations and Logistics and Informatics. 2010: 300-305

[7] Mihailescu M, Teo Y M. Dynamic Resource Pricing on Federated Clouds[C]// IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. 2010: 513-517

[8] Paleologo G A. Price-at-Risk: A methodology for pricing utility computing services[J]. IBM Systems Journal, 2004, 43(1): 20-31

[9] Chen Y, Das A. Pricing-based strategies for autonomic control of web servers for time-varying request arrivals[J]. Engineering Applications of Artificial Intelligence, 2004(17): 841-854

[10] Ouyang J, Sahai A. A mechanism of specifying and determining pricing in utility computing environments[C]// IEEE/IFIP International Workshop on Business-Driven IT Management. 2007: 39-44

[11] Buyya R, Yeo C S. Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities [C]// 10th IEEE International Conference on High Performance Computing and Communications. 2008: 5-13

[12] Henzinger T A, Singh A V. FlexPRICE: Flexible provisioning of resources in a cloud environment[C]// IEEE International Conference on Cloud Computing. 2010: 83-90

[13] Yeo C S, Venugopal S. Autonomic metered pricing for a utility computing service[J]. Future generation computer systems, 2010(26): 1368-1380

[14] Woitaszek M, Tufo H M. Developing a cloud computing charging model for high-performance computing resources[C]// IEEE International Conference on Computer and Information Technology. 2010: 210-217

[15] Jiang G, Cybenko G. Functional validation in grid computing [J]. Autonomous Agents and Multi-agent Systems, 2004(8): 119-130

[16] Chandra A, Gong W, Shenoy P. Dynamic resource allocation for shared data centers using online measurements[C]// ACM/IEEE International Workshop on Quality of Service. 2003: 381-400

[17] Furht B, Escalante A. Handbook of cloud computing[Z]. New York: Springer, 2010

(上接第 186 页)

利用该算法对潜在别名进行校验和修正,进而抽取出块内所有有效别名。在 Enron 邮件数据集上的实验结果表明,本文提出的基于统计和规则过滤的块定位算法可较为准确、完整地提取出邮件正文中的称呼块和签名块文本片段;同时,别名边界词汇模板的引入,在很大程度上提高了仅依据命名实体识别工具提取出的别名的有效性和完整性。

参 考 文 献

[1] Sarawagi S, Bhamidipaty A. Interactive deduplication using active learning[C]// ACM Special Interest Group on Knowledge Discovery and Data Mining. 2002

[2] Bhattacharya I, Getoor L. A latent dirichlet model for unsupervised entity resolution[C]// The SIAM International Conference on Data Mining (SIAM-SDM). Bethesda, MD, USA, 2006

[3] Bollegala D, Matsuo Y, Ishizuka M. Disambiguating personal names on the web using automatically extracted key phrases[C]// Proc. of the 17th European Conference on Artificial Intelligence. 2006: 553-557

[4] Bollegara D, Matsuo Y, Ishizuka M. Extracting key phrases to disambiguate personal names on the web[C]// Proc. CICLing 2006. 2006

[5] Bollegala D, Honma T. Identification of Personal Name Aliases on the Web[C]// Proceedings of WWW 2008 Workshop on Social Web Search and Mining (SWSM 2008). Beijing, China, 2008

[6] Bollegala D, Honma T, Matsuo Y, et al. Mining for personal name aliases on the web[C]// Proceeding of the 17th international conference on World Wide Web. Beijing, China, April 2008

[7] Bird C, Gourley A, Swaminathan A. Mining Email Social Networks[C]// Proceedings of the 2006 international workshop on Mining software repositories. Shanghai, China, 2006: 137-143

[8] Diehl C, Getoor L, Namata G. Name reference resolution in organizational email archives[C]// Proceddings of SIAM International Conference on Data Mining. Bethesda, MD, USA, April 2006

[9] Elsayed T, Oard D W. Modeling Identity in Archival Collections of Email[C]// Proceedings of the Third Conference on Email and Anti-Spam. Mountain View, California, USA, 2006

[10] Elsayed T, Oard D W, Namata G. Resolving personal names in email using context expansion[Z]. Association for Computational Linguistics(ACL), 2008

[11] Elsayed T, Namata G, Getoor L, et al. Oard. Personal name resolution in email: A heuristic approach[R]. UMIACS LAMP-TR-150. University of Maryland, March 2008

[12] Chen H, Hu J, Sproat R. Integrating geometrical and linguistic analysis for e-mail signature block parsing [J]. ACM Transactions on Information Systems, 1999, 17(4): 343-366

[13] Carvalho V, Cohen W. Learning to extract signature and reply lines from email[C]// Proceedings of the 2004 Conference on Email and Anti-Spam (CEAS 04). August 2004

[14] Stanford University. Named Entity Recognition System [EB/OL]. http://nlp.stanford.edu/software/stanford-ner-2009-01-16.tgz, 2009

[15] The email collection of Enron Corporation [DB/OL]. http://www.cs.cmu.edu/~enron/, 2003