Internet 中基于启发式遗传算法的受限镜像放置问题求解

郭常杰, 钟玉琢, 林 闯
(清华大学计算机系, 北京 100084)

摘 要: 求解受限镜像放置问题对于 Internet 内容提供商具有重要的应用价值, 但其在数学上归结为 k-supplier 问题, 是一个 NPC 问题. 本文提出了一种求解受限镜像放置问题的启发式遗传算法, 该算法采用启发式交叉和变异算子, 本质上是对标准两点交叉和单点变异产生的非法染色体进行启发式修正, 以提高算法的局部搜索能力, 加速收敛. 仿真实验详细考察了启发式算子对收敛速度与全局优化性能的影响, 并与两种常用启发式算法进行了性能比较. 实验结果表明本文算法具有快速收敛, 高性能, 全局收敛等优点, 证明其可行性和有效性.

关键词: 放置; 镜像; 启发式; 遗传算法
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2002) 05-0689-05

Solving Constrained Mirror Placement Problem Based on Heuristic Genetic Algorithm in Internet

GUO Chang-jie, ZHONG Yu-zhuo, LIN Chuang
(Computer Science and Technology Department, Tsinghua University, Beijing 100084, China)

Abstract: Solving constrained mirror placement problem is very valuable to content providers in Internet. Constrained mirror placement problem can be modeled as a k-supplier problem, which belongs to NP-Complete problem and has only best possible 3-approximate algorithm. A heuristic genetic algorithm is presented in this paper, which uses the heuristic algorithm to revise the new illegal chromosome generated by standard two-points crossover and one-point mutation operations. The simulation results suggest that the algorithm is feasible and effective.

Key words: placement; mirror; heuristic; genetic algorithm

1 引言

在 Internet 中, 客户对信息的访问具有很大的不一致性, 流行的内容往往造成"热门"网站的过载, 从而严重的增加了访问这些内容和网站的等待延迟时间[1]. 解决问题的一个有效方法就是通过在靠近客户的地方放置镜像(或代理)服务器来均衡负载, 提高响应速度. 1998 年资料表明[2], Netscape 公司在全球放置了 101 个镜像, Yahoo 则在美国放置了 15 个镜像. 但显然, 建立和维护镜像需要付出代价, 且随着 Internet 的急剧扩大和客户群的分散, 如何合理放置有限的镜像资源已成为具有重要研究价值的课题. 经典的镜像放置问题在数学上归结为最小 k-中心(Minimum k-center)问题[3], 即给定加权连通图, 如何选择 k 个节点作为中心, 以最小化任意节点到达最近中心的最大距离. 最小 k-中心问题是一个典型的 NP 完全问题[4].

在经典的镜像放置问题中, 网络中的任意节点均可被选择放置镜像; 但实际网络并非如此, 只有部分的非客户节点, 如 ISP、网络中心等可供选择放置镜像, 为此 Sugih 等提出了受限镜像放置问题[5], 简称 CMP(Constrained Mirror Placement). 受限镜像放置问题具有重要的应用价值, 如它能帮助内容提供商(Content Provider)从分布在各地的多个可放置镜像节点选择可产生最大效益的子集; 但当前对其研究还很少, 仅有少数几个启发式算法[5,6]. 事实上, 受限镜像放置问题也归结为 k-供应商问题(k-supplier, 最小 k-中心问题的变种), 已经证明[7], 该问题是一个 NP 完全问题, 且只有最好为 3 的近似算法. 遗传算法[8]是一种新型的优化工具, 具有并行搜索的特性, 广泛应用于解决各类具有 NP 难度的问题. 本文通过对受限镜像放置问题的分析, 提出了一种基于启发式遗传算法的求解方法. 该算法在交叉和变异算子中引入启发式搜索信息, 以提高搜索效率, 使得算法具有快速收敛、高性能、全局收敛等优点. 我们通过仿真证明了算法的有效性, 并深入的分析了原因.

收稿日期: 2001-08-10; 修回日期: 2001-11-25
基金项目: 国家自然科学基金(No. 69973025); 973 项目(No. G1999032700)

2 受限镜像放置问题的数学模型

将受限镜像放置问题描述为一个连通无向加权图 $G(V, E)$[5], 其中 V 表示网络节点集合, $E \subseteq V \times V$ 为连接网络节点的通信链路. 对于任一链路 $e \in E$, 定义非负函数 $c(e): E \rightarrow R^+$, 称为链路代价函数; 本模型中, 链路代价表示实际应用中的某种可加度量, 如距离、延迟、费用等. 对于任意两个节点 $u, v \in V$, 定义非负函数 $dist(u, v): V \times V \rightarrow R^+$, 称为节点间的距离, 其计算方法如下: 设边序列 $(e_1, e_2, \cdots, e_k)$ 为 u, v 之间的最短路径, 则有 $dist(u, v) = \sum_{i=1}^{k} c(e_i)$.

定义 $H \subseteq V$ 表示网络中可放置镜像的候选节点集合, $B \subseteq V$ 代表客户集合, 在本模型中满足: $H \cup B \subseteq V, H \cap B \subseteq \Phi$. 给定任意正整数 $k \in [1, |H|]$ 表示待放置的镜像个数, 希望找到节点集合 $M \subseteq H$, 且 $|M| \leq k$, 使得任意 B 中节点到达 M 中最近节点的最大距离最小化. 受限镜像放置问题的形式化描述为:

$$\text{Minimize } Cost(M) = \max_{b \in B} \min_{m \in M} dist(b, m) \quad (1)$$

$$\text{S.T.} \quad M \subseteq H \quad (2)$$

$$|H| \leq k \quad (3)$$

目前求解受限镜像放置问题只有启发式算法, 其中性能最好的由文[7]提出, 简称为 SUP 算法. 该算法的基本思想是: 从任一客户开始, 寻找距离其最近的镜像放置候选节点, 并在该节点放置镜像; 然后再寻找距离当前已放置的所有镜像最远的客户, 并在距离该客户最近的候选节点放置镜像; 如此重复直到放置的镜像个数达到要求为止.

在实际应用中, 由于网络规模的不断扩大以及客户群的分散, 往往需要从大量分布的候选网络节点中选择并放置固定数量的镜像, 因此该问题是一个复杂的全局搜索问题. 已提出的启发式算法可在多项式时间内找到近似优化解, 但并不能保证解的可行性和最优性, 在某些情况下, 甚至无法阐述所得解与最优解的近似程度(如文献[5]提出的贪婪算法). 即使是上述文献[7]提出的最优启发式算法, 在最坏情况下仍可能产生 3 倍于最优解的近似解, 显然是无法忍受的.

3 算法

J. Holland 于 1975 年提出遗传算法, 其本质是模拟生物进化过程的一种并行优化算法, 适用于在复杂而庞大的搜索空间中寻找最优解或近似解. 遗传算法的以上特点, 使其成为解决 NP 困难问题的有效工具. 但基本的遗传算法往往求解效率不高, 需要较长的进化时间才可搜索到较好的近似解, 为此本文提出一种基于启发式遗传算法的受限镜像放置问题求解. 该算法在进化过程中引入启发式信息, 指导搜索过程, 提高搜索效率, 具有以下特点: (1) 混合的初始种群选取; (2) 动态的适应度函数; (3) 特别的是, 引入启发式交叉和变异算子, 极大的增强了算法的局部搜索能力.

3.1 编码

使用长度为 $|H|$ 的一维二进制编码作为遗传算法的编码机制. 任意染色体表示为 $S = \{s_1, s_2, \cdots, s_{|H|} \mid s_k \in \{0, 1\}\}$, 其中 s_k 表示候选节点集合 H 中第 k 个节点是否被选中放置镜像($s_k = 1$ 表示被选中, $s_k = 0$ 表示未被选中).

定义 **1** 任意染色体 S, 如果 $\sum_{j=1}^{|H|} s_j = |M|$, 称其为合法染色体; 否则称为非法染色体.

本算法中, 通过选择合法的初始种群, 并利用启发式的交叉和变异算子, 防止了非法染色体的产生. 显然, 这种编码方式既可以简化编解码操作, 又能保证编码的完备性、健全性和非冗余性[8].

3.2 初始种群

算法在执行前需要预先选择 Ng 条染色体作为初始种群. 本算法的初始种群构造如下:

(1) 以文献[5]提出的贪婪算法的近似解产生一条染色体;

(2) 以文献[7]提出的 SUP 算法的近似解产生一条染色体;

(3) 随机生成另外 $Ng-2$ 条染色体;

将两种启发式算法的结果引入初始种群具备两个优点: (1) 加速收敛; (2) 获得不差于启发式算法的近似解. 另外, 鉴于初始种群的群体规模 Ng 选取的较大(实验中为 80), 在仿真中未发现因引入启发式算法而导致早熟的现象.

3.3 适应度函数的建立

适应度函数评估是选择操作的依据, 其设计是否合理直接影响算法的性能. 受限镜像放置问题的目标是最小化客户的到达最近镜像的最大距离, 因此首先为染色体定义代价函数 $g: \{0, 1\}^{|H|} \rightarrow R^+$ 为 $g(S) = \max_{b \in B} \min_{h_i \in H \cap s_i = 1} dist(b, h_i)$. 显然, $g(S)$ 取值越小, 染色体越优. 为将此最小化问题转换为最大化问题, 通常构造适应度函数如下[8]:

$$Fit(S) = \begin{cases} Cmax - g(S), & g(S) < Cmax \\ 0, & \text{else} \end{cases} \quad (4)$$

其中 $Cmax$ 称为选择系数, 通常取值为一个预先制订的大数或群体中曾经出现的最大值. 但此种取值机制有两个缺点: 第 1, $Cmax$ 的取值对算法的性能影响很大. 若其取值过大, 导致不同染色体的适应度趋近, 优良染色体竞争力下降, 优化过程趋于盲目的漫游, 从而降低收敛速度, 增加了进化代数; 若其取值过小, 部分超常个体将在群体中占据过大比例, 将导致早熟现象. 问题的关键是, 我们事先未必能够准确的估计最优解的范围, 因此很难给出适合的值. 第 2, 在进化一定代数之后, 虽然群体中个体多样性尚存在, 但往往会出现群体的平均适应度已经接近最佳个体适应度, 此时应该动态调整选择系数, 以便增加优良个体的竞争力. 但预先制订 $Cmax$ 值的策略无法满足以上需求, 为此, 本文采用一种动态的 $Cmax$ 值计算方法如下:

$$Cmax = \alpha \sum_{i=1}^{Ng} g(S(i))/Ng \quad (5)$$

其中因子 α 用于调整不同进化阶段 $Cmax$ 的值, 在实验中我们对其取值方法如下: 进化初期, 取一个较大值($\alpha = 1.5$), 可以防止超常个体的出现导致算法陷入早熟; 随着进化代数的增加, α 取值不断减小(最终为 1.05), 以增强优良个体竞争力, 加快收敛速度. 显然, 比起预先设定 $Cmax$ 值, 该种策略能够较好克服前文所述的两个缺点.

3.4 选择方法

选择的目的是把优化的个体直接遗传到下一代或通过交叉变异产生新个体再遗传到下一代. 本文采用最佳个体保留法, 即在群体进行其他操作之前, 预先选出最佳个体直接遗传到子代群体. 其余个体采用轮盘赌方式选择: 即第 k 个染色体 $S(k)$ 被选中的概率为:

$$P(S(k)) = Fit(S(k)) / \sum_{j=1}^{Ng} Fit(S(j)) \qquad (6)$$

3.5 启发式遗传算子

3.5.1 交叉算子

交叉操作指两个染色体按照某种方式相互交换部分基因, 从而形成两个新的子个体, 该操作是解得以进化的核心算子. 针对受限镜像放置问题, 本文定义了一种启发式交叉算子(HCX), 该算子利用启发信息修正交叉后产生的子代, 以保证其合法性. HCX 算子描述如下:

(1) 对两个父代染色体进行两点交叉

对于父代染色体 X 和 Y, 随机选择两个交叉点 p_1 和 p_2 ($p_1 \le p_2$), 交叉时两个染色体介于两个交叉点之间的基因互换, 形成子代染色体 X' 和 Y'. 对于本算法的染色体, 最多可能存在 $(|H|-2)\cdot(|H|-3)$ 种交叉点的设置[8].

两点交叉完成后, 子代染色体满足下式:

$$\sum_{j=1}^{|H|} x'_j = |M| - \sum_{i=p_1}^{p_2} (x_i - y_i)$$

$$\sum_{j=1}^{|H|} y'_j = |M| + \sum_{i=p_1}^{p_2} (x_i - y_i)$$

令 $\sigma = \sum_{i=p_1}^{p_2} (x_i - y_i)$, 显然当 $\sigma \neq 0$ 时, 根据定义 1 可知, 子代染色体 X' 和 Y' 均为非法染色体, 需要将其修正为合法染色体. 最简单的修正方法是随机修正: 即为 X' 和 Y' 随机增加或减少 $|\sigma|$ 个值为 1 的基因. 随机修正虽然简单, 但由于其盲目性, 极易破坏染色体中的优良基因, 导致其退化. 4.1 节的仿真结果证明了这一点. 为此, 提出了一种启发式的修正办法, 在第 2 步做出描述.

(2) 对两个子代染色体进行启发式修正

不失一般性, 令 $\sigma > 0$, 此时 X' 染色体缺少 σ 个值为 1 的基因, 对其修正如下:

步骤 1.1: 对任意处于 $[1, p_1) \cup (p_2, |H|]$ 且值为 0 的基因 x'_i, 计算修正权值如下:

$$V(x'_i) = g(X') - g(X' \oplus x'_i) \qquad (7)$$

其中 $X' \oplus x'_i$ 表示将染色体 X' 的第 i 个基因值置 1 后得到的新染色体.

步骤 1.2: 选择具有最大修正权值的基因 x'_{best}, 令 $X' = X' \oplus x'_{best}$.

步骤 1.3: 重复执行步骤 1.1 和步骤 1.2 共 σ 次, X' 即为修正后的合法染色体.

相对应的, Y' 染色体多余 σ 个值为 1 的基因, 对其修正如下:

步骤 2.1: 对任意处于 $[1, p_1) \cup (p_2, |H|]$ 且值为 1 的基因 y'_i, 计算修正权值如下:

$$V(y'_i) = g(Y' \oplus y'_i) - g(y'_i) \qquad (8)$$

其中 $Y' \oplus y'_i$ 表示将染色体 Y' 的第 i 个基因值置 0 后得到的新染色体.

步骤 2.2: 选择具有最小修正权值的基因 y'_{best}, 令 $Y' = Y' \oplus y'_{best}$.

步骤 2.3: 重复执行步骤 2.1 和步骤 2.2 共 σ 次, Y' 即为修正后的合法染色体.

以上的交叉策略, 采用了基于贪婪的启发式算法修正非法的子代染色体, 尽量保存具有高适应度的优良基因, 使子代能够具备更好的性能, 加快算法的收敛速度. 启发式交叉算子以概率 Pc 执行.

3.5.2 变异算子

本文提出的变异方法为: 在染色体 S 上任取一个基因 s_i, 以小概率 Pm 对其值进行取反操作, 得到新染色体 S'. 显然, 变异后的染色体为非法染色体, 对此, 我们采用类似于交叉算子的启发式策略修正, 规则如下:

(1) 如果 $s_i = 0$, 采用 3.5.1 节中对染色体 Y' 的修正方法 (步骤 2.1~2.3);

(2) 如果 $s_i = 1$, 采用 3.5.2 节中对染色体 X' 的修正方法 (步骤 1.1~1.3);

可以看出, 上述的启发式变异算子, 在翻转某个基因值后, 又采用基于贪婪的搜索方法寻找最优的替代基因, 从而增强了遗传算法的局部搜索能力.

3.5.3 逆转算子

本文采用的逆转算子操作方法如下: 在染色体内, 随机选择两个逆转点, 将两点之间的基因串反序后插入原位置. 经过逆转操作的染色体不改变其合法性, 因此无需修正. 同变异操作类似, 逆转操作也以小概率 Pi 进行. 逆转算子不进行启发式修正, 因此其局部搜索能力低于启发式变异算子, 引进该算子的主要目的是增强群体的多样性.

4 仿真试验和结果分析

仿真程序采用标准 C++ 编写, 运行平台为一台 PIII650/64M 的 PC 机. 网络模型由拓扑生成工具 BRITE[9] 按 WAXMAN 模型构造, 具体方式为: ①在大小为 $L \times W$ 的矩形面板内随机放置 N 个节点; ②任意两个节点以概率 $P(u, v) = \alpha e^{-d(u,v)/\beta L}$ 连线, 其中参数 $0 < \alpha, \beta \le 1$ 分别用于调节图中长短边比例和节点的度数, d 是节点 u, v 之间的欧几里的距离, L 代表面板中任意两点之间的最大距离. 不失一般性, 将延迟作为边的链路代价; ③随机选择 $|H|$ 个节点作为镜像放置的候选节点, 并从剩余节点中选择 $|B|$ 个作为客户节点.

仿真试验首先验证了启发式交叉和变异算子对算法的收敛速度、全局优化性能等因素的影响; 然后同两种启发式算法进行性能比较.

4.1 启发式遗传算子对收敛速度与全局优化性能的影响

实验中的遗传算法参数为: $Pc = 0.4$, $Pm = 0.001$, $Pi = 0.001$, $Ng = 80$. 已经证明[8], 如果变异概率 $Pm \in (0, 1)$, 交叉概率 $Pc \in [0, 1]$, 同时采用比例选择法(即轮盘赌), 且在选择前保留当前最优解的遗传算法可以收敛到全局最优解. 显然, 本文提出的算法完全符合上述条件, 可以收敛到全局最优解.

为提高搜索效率, 本文采用启发式交叉和变异算子, 它们本质上是采用启发信息修正操作后产生的非法子代染色体,

以增强局部搜索能力. 为验证其效果, 将其与 3. 5 节中提到的随机修正策略进行仿真比较. 组合启发式算子和随机算子得到四种进化策略: (1)随机交叉+ 变异算子; (2)随机交叉+ 启发式变异算子; (3)启发式交叉+ 随机变异算子; (4)启发式交叉+ 变异算子. 实验中发现策略(1)近乎盲目搜索, 很难收敛到较好的近似解, 故只比较后三种策略.

首先在 1000× 100 的面板内生成多组不同规模($N \times |B| \times |H|$)的随机网络拓扑. 在选取不同镜像数($|M|$)的情况下, 分别采用三种策略进行了大量的实验. 限于篇幅, 只列出其中的 9 组数据; 对每种规模的问题, 运行 20 次获得表 1 中结果. 其中进化代数为实施对应策略进化到其最优值的平均代数.

表 1 三种算子组合策略的性能比较

		随机交叉+ 启发式变异算子			启发式交叉+ 随机变异算子			启发式交叉+ 启发式变异算子		
网络规模	镜像数	最优值	平均值	进化代数	最优值	平均值	进化代数	最优值	平均值	进化代数
300* 200* 50	5	377	377. 8	62	377	377	18	377	377	11
	7	360	362. 0	91	360	360. 2	35	360	360	19
	9	342	353. 2	118	336	339. 3	47	332	334. 2	40
600* 400* 100	7	397	398. 7	97	397	397	33	397	397	24
	9	378	387. 2	111	374	376. 1	49	374	374. 3	41
	11	373	384. 9	155	368	369. 2	63	363	363. 8	51
900* 700* 150	9	431	436. 1	146	417	419. 4	68	417	417	53
	11	413	417. 7	193	412	412. 5	92	409	410. 8	71
	13	412	413. 5	266	401	40	113	396	396. 7	85

另外通过一个实例, 分析启发式算子在算法收敛过程中的作用. 实例的网络规模为 600* 400* 100, 镜像数为 11. 图 1 为分别采用同表 1 的三种策略, 实例的收敛过程.

从表 1 和图 1 的结果可以看出, 采用启发式的修正策略能够以更小的进化代数获得较好的性能. 并且随着网络规模的扩大, 算法的收敛速度仍然很快, 说明其效率很高. 导致以上实验结果的原因是, 启发式的交叉和变异算子在修正非法染色体时, 能够防止优良基因的破坏, 使算法可以更快的收敛到全局最优解. 相反, 随机修正虽然简单, 但是因为经常破坏优良基因而使算法陷入盲目随机搜索, 迟迟无法收敛到全局最优解.

4. 2 算法性能

选择文[5]中的贪婪算法和文[7]中的SUP 算法进行性能比较. 使用同 4. 1 节的网络拓扑, 改变镜像个数, 在每种情况下, 分别运行三种算法, 计算最小延迟. 其中本文启发式遗传算法的值是进行 20 次实验的平均值. 鉴于文[7]算法性能和选择的初始点相关, 故实验中的结果为随机选择 10 个初始点, 取其中的最优值. 图 2 和图 3 分别是在网络规模为 600* 400* 100 和 900* 700* 150 时, 性能比较结果.

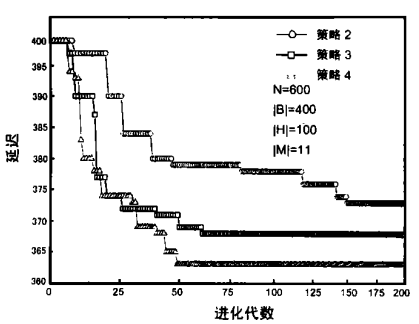

图 1 三种策略下实例的收敛过程

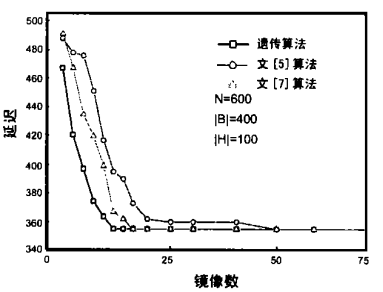

图 2 网络规模 600* 400* 100 下性能比较

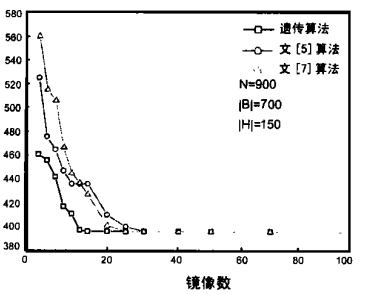

图 3 网络规模 900* 700* 150 下性能比较

从图 2 和图 3 可知, 随着放置镜像数的增加, 三种算法都将收敛到同一个优化值, 该值即放置最大数目的镜像($|H|$)可获得的最优值, 但它们收敛的速度显然不同. 实验结果表明, 不同网络规模下, 本文算法求出的结果均优于两种启发式算法.

5 结论

本文利用遗传算法来求解受限镜像放置问题, 该问题能够帮助 Internet 内容提供商从大量分布的候选节点中选择合适子集以放置有限的镜像资源. 算法的本质是利用启发式算法修正标准两点交叉和单点变异算子产生的非法染色体, 以提高局部搜索能力. 算法还采用了混合的初始群体选择以及动态调整的适应度函数. 不同网络规模下的仿真实验结果都证明了算法的有效性和可行性. 特别的是, 我们通过比较两类启发式算子和随机算子, 证明了前者具备较强的局部搜索能力, 能够有效的加速算法收敛, 提高算法性能.

参考文献:

[1] Tewari R, Dahlin M, Vin H M, Kay J. Beyond hierarchies: design considerations for distributed caching on the Internet [A]. Proc of the 19th Inter Conf On Distributed Computing Systems [C]. Austin, USA: 1999.

[2] M Sayal, Y Breitbart, P Scheuermann, R Vingralek. Selection algorithms

for replicated web servers [A]. Internet Server Performance Workshop in conjunction with ACM Sigmetrics' 98/ Performance' 98 [C]. Madison, USA: 1998.

[3] Michael R Garey, David S Johnson. Computer and Intractability [M]. NY: W H Freeman and Co, 1979.

[4] O Kariv, S L Hakimi. An algorithmic approach to network location problems, part I: the p-centers [J]. SIAMJ. Appl Math, 1979, 37: 513 - 538.

[5] Sugil Jamin, Cheng Jin, Anthony R Kurc, Dan Raz, Yuval Shavitt. Constrained mirror placement on the Internet [A]. Proc of IEEE INFOCOM2001 [C]. Anchorage, AK: 2001.

[6] Sugil Jamin, Cheng Jin, Yixin Jin, Dan Raz, Yuval Shavitt, Lixia Zhang. On the placement of Internet instrumentation [A]. Proc of IEEE INFOCOM2000 [C]. Tel Aviv, Israel: 2000.

[7] Dorit S Hochbaum. Approximation Algorithms for NP-Hard Problems [M]. International Thomson Publishing, 1999.

[8] 陈国良, 等. 遗传算法及其应用 [M]. 北京: 人民邮电出版社, 1996.

[9] A Medina, A Lakhina, I Matta, J Byers. BRITE: Boston university representative internet topology generator [DB/OL]. March 2001. http://cs-pub.bu.edu/brite/index.htm.

作者简介:

郭常杰　男, 1976 年生于江苏省连云港市, 清华大学计算机系人机交互与媒体集成研究所博士研究生, 主要研究领域为视频流化技术, 多媒体代理缓存.

钟玉琢　男, 1938 年生于辽宁省沈阳市, 清华大学计算机系教授, 博士生导师, 主要研究领域为多媒体数据压缩编码和分布式视频服务器.