一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
SITE2023-98,IA2023-104(2024-03)

低信頼なエッジキャッシュに対する最適キャッシュ制御法

吉田　　開†　　上山　憲昭†

† 立命館大学 情報理工学部　〒 525-0058 滋賀県草津市野路東 1-1-1
E-mail: †is0588pi@ed.ritsumei.ac.jp, ††kamiaki@fc.ritsumei.ac.jp

あらまし　Mobile Edge Cache（MEC）は, 無線の基地局にキャッシュサーバ (ES: Edge Server) を設置し, 人気のあるコンテンツをユーザの近くの ES にキャッシュをすることで, 検索待ち時間, ネットワークの混雑, リモートコンテンツプロバイダへのリクエスト数を削減することが期待されている. しかし無線の基地局の設置数は膨大であるため, 一般的に MEC ではコストの低い低信頼な ES が用いられており, 障害による不稼働率の増加が課題となっている. 低信頼な ES を想定した研究では, erasure coding を用いてキャッシュコンテンツの可用性を高めているが, コンテンツを事前に ES へ配置することを想定している. しかし実際の ES では LRU （Least Recently Used）等を用いた置換による動的制御が一般的である. そこで本稿では, 低信頼な ES において LRU を用いた置換制御を想定し，ES の不稼働を考慮した上で ES からのコンテンツ取得可能性を向上させる erasure coding を用いた ES へのコンテンツ挿入法を提案する. また目標ヒット率 $\hat{h}_m$ と実際のヒット率 h_m との差異が小さくなるよう遺伝的アルゴリズムを用いてキャッシュ挿入確率 f_m を適切に設定する. また全コンテンツ m の平均取得成功率（ES からのチャンク取得のみでコンテンツを復元できた割合）を提案方式を用いてキャッシュ挿入確率 f_m を設定した場合と, キャッシュミス時に常にコンテンツをキャッシュした場合について数値評価により比較し, 提案方式は平均取得成功率を大きく改善することを確認する.
キーワード　モバイルエッジキャッシュ, 消去符号化，遺伝的アルゴリズム

Optimum Cache Method for Unreliable Edge Caches

Kai YOSHIDA† and Noriaki KAMIYAMA†

† College of Information Science and Engineering, Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu, Shiga 525-0058
E-mail: †is0588pi@ed.ritsumei.ac.jp, ††kamiaki@fc.ritsumei.ac.jp

Abstract　Mobile Edge Cache (MEC) is expected to reduce latency, network congestion, and the number of requests to remote content providers by installing a cache server (ES: Edge Server) at the wireless base station and caching popular content at an ES near the user. However, the installation of a wireless base station is not sufficient to reduce the number of requests to the remote content providers. However, since the number of wireless base stations is huge, low-cost and low-reliability ESs are generally used in MEC, and the increase of unavailability rate due to failures is an issue. In the studies assuming low-reliability ESs, erasure coding is used to increase the availability of cached contents, but it is assumed that the contents are placed in the ESs in advance. In actual ESs, however, dynamic control by replacement using LRU (Least Recently Used) and so on is common. Therefore, in this paper, we propose an ES insertion method using error coding that improves the possibility of retrieving contents from ESs, with considering the non-availability of ESs that are assumed to be controlled by replacement using LRU in low-reliability ESs. To decrease the difference between the target hit rate $\hat{h}_m$ and the actual hit rate h_m, we optimize the cache insertion probability f_m by using a genetic algorithm. We numerically compare the average success rate of retrieval of all contents m, i.e., the percentage of contents recovered only by retrieving chunks from the ES, between the case where the cache insertion probability f_m is set using the proposed method and the case where contents are always cached in case of cache misses. As a result, we confirm that the proposed method significantly improves the average success rate of retrieval compared to the case where the content is simply cached in the ES.
Key words　Mobile Edge Cache （MEC）, erasure coding, genetic algorithm

1. はじめに

スマートフォン等の移動端末で動画を視聴する機会が増加しており, モバイルネットワークの負荷が急増し, 通信品質の劣化やネットワークコストの増大が課題となっている. そこで 5G では, 無線の基地局にキャッシュサーバ（ES: edge Server）

を設置し, ユーザに近い ES からコンテンツを配信する Mobile Edge Cache（MEC）が用いられている． MEC は近接して配置された ES 間で協調的なエッジキャッシュネットワークを形成し [2], 人気のあるコンテンツをユーザの近くの ES にキャッシュをすることで, ユーザが要求したコンテンツを取得する際の遅延を大幅に削減することが可能である．また MEC はバックホールリンクを介したデータ伝送を回避するため, バックホールトラフィックを削減することが可能である．MEC については, 様々な目的や用途で数多くの研究が行われているが, これらの研究のほとんどはキャッシュの障害を考慮せず高信頼な ES を前提としている．しかし高信頼なエッジキャッシュ資源は限られており [3], 非常に高価である [4]．実際の MEC では, ES の設置数は膨大であるためコストの低い低信頼な ES が用いられ，障害による不稼働率の増加が課題となっている．また従来のキャッシュ制御では全ての ES が常時稼働し, 想定したキャッシュ容量が常に利用できることを想定しているが, 低信頼な ES を想定したキャッシュ制御が必要である．

また近年, 高いストレージ効率と信頼性を提供するために, 複数のサーバに冗長性を持たせた分散データストレージとして erasure coding がストレージシステムで広く利用されている [5] [6] [7]．Maximum distance separable (MDS) code 等の Reed-Solomon (RS) 符号 [9] は, ストレージ容量効率の点で最適であり, ストレージシステムで広く使用されている [5] [6] [10] [11]．低信頼な ES を想定した既存研究 [12] では, エッジ資源の信頼性の低さに対処するために erasure coding を用いて ES の可用性を高める方式を提案しているが, コンテンツを ES 上に事前配置することを想定している．しかし実際の ES では LRU （Least Recently Used）等を用いた置換による動的制御が一般的である．そこで本稿では, 低信頼な ES において LRU を用いた置換制御を想定した ES の不稼働を考慮した上で，ES からのコンテンツ取得可能性を向上させる erasure coding を用いた ES 挿入法を提案する．本稿の貢献を以下にまとめる．

- 各コンテンツ m の ES 上のキャッシュ数の期待値 W_m が目標値 T_m となる目標ヒット率 $\hat{h}_m$ を算出する．
- 目標ヒット率 $\hat{h}_m$ と, 実際のヒット率 h_m の差異が小さくなるようキャッシュ挿入確率 f_m を最適設計する遺伝的アルゴリズムを提案する．
- 数値評価により，キャッシュ挿入確率 f_m を提案方式で最適化することで, 実際のヒット率 h_m と目標ヒット率 $\hat{h}_m$ の差異は小さく, 適切に f_m を設定できていること確認する．
- さらに平均取得成功率（ES から取得したチャンクのみでコンテンツを復元できた割合）を提案方式を用いてキャッシュ挿入確率 f_m を設定した場合と，キャッシュミス時に常にコンテンツをキャッシュした場合で比較し, 提案方式の有効性を明らかにする．

以後，2 節で関連研究をまとめ，3 節で既存の信頼性の低いリソースを持つ分散協調エッジキャッシングシステムのためのキャッシュ配置アルゴリズムの概要を述べる．4 節で, 本稿で新たに提案する ES へのチャンク挿入アルゴリズムについて述べ, 5 節で数値評価結果を示し，最後に 6 節で全体をまとめる．

2. 関 連 研 究

MEC では, 近接して配置された ES 間で協調的なエッジキャッシュネットワークを形成し, 人気のあるコンテンツをユーザの近くの ES にキャッシュすることで, 検索待ち時間やネットワークトラフィックを削減することが可能である．Ramaswamy らは効果的で効率的な協調エッジキャッシュネットワークを生成するために，与えられたエッジキャッシュネットワークのサーバを指定された数の協調キャッシュグループに分割する方法を提案している [14]．さらに Ramaswamy らは, 動的コンテンツをキャッシュするための協調的なエッジネットワークを設計する際の課題に取り組み，大規模なエッジキャッシュネットワークにおける協調のための枠組みとして新たにキャッシュクラウドを提案した [2]．しかし MEC に関する上記の全ての研究では, キャッシュ障害を考慮せず, 高信頼な ES を前提としている．

一方, 信頼性を向上させ, 冗長性を備えた分散データストレージシステムを使用することが注目されている [5]．レプリケーションは, 冗長性を備えた分散データストレージの最も単純な形式である [8]．レプリケーションでは, オリジナルデータの同一のコピーをレプリカとして別々のストレージにコピーをし, 冗長性を提供する．あるノードで障害が発生し, データが失われた場合に, 別のノードにて複製されたコピーを使用して情報を復元することが可能である．しかしレプリケーションでは元のデータの完全なコピーを各ノードに保存する必要があるため, より多くのストレージ容量が必要となる．そこで信頼性とストレージ効率を向上させる erasure coding がストレージシステムで広く利用されている．erasure coding では, データを断片に分解し, 消去符号と呼ばれる追加の断片を生成する．生成した断片は複数のノードに分散され, 一部の断片が破損, 消失したとしても元のデータを復元することを可能にする．erasure coding をストレージシステムで用いることで, 障害により一定数のストレージノードが利用できない場合や, データの損失が発生した場合に元のデータを再構築できるためシステムの耐障害性やデータの信頼性, 耐久性を向上させることが可能となる．また, データを完全に複製することなく冗長性を実現するため, ストレージ効率を向上することが可能となる．

そこで低信頼な ES に対処するために, erasure coding を活用してキャッシュするファイルの信頼性を高める方法が考えられるが, 信頼性の低い ES から取得できるデータサイズを最大化することは以下の理由により困難である．

- ES のキャッシュ容量には限りがあるため, キャッシュされるファイルの信頼性と ES がキャッシュできるファイル数の間にはトレードオフがある．
- 異なるファイルには異なる人気があるため, より人気のあるファイルの存続は, 人気のないファイルよりもユーザに利益をもたらす可能性がある．
- ES の故障確率は異なり, 各ファイルの信頼性はチャンクの場所によって変わる．

このような課題を解決するために, 容量制約の下で各ファイルに追加される冗長性, 各ファイルのチャンク数やチャンクの場

所を適切に設定する必要がある．そこで, 信頼性の低いリソースを持つ分散協調エッジキャッシングシステムのためのキャッシュ配置アルゴリズムが提案されている [12].

3. DEC

3.1 低信頼なキャッシュ資源を持つ分散協調キャッシングシステム

[12] で Li らは, 低信頼なキャッシュ資源を持つ分散協調キャッシングシステム（DEC）を提案した．DEC システムで利用する記号と定義を以下の表 1 に示す．DEC では各ファイルの長さは全て M ビットと仮定し [1]，各ファイルをサイズが M/k の k 個のチャンクに分割し，RS 符号などの MDS 符号 [2] によって k 個のチャンクから x 個の冗長チャンクを生成し，これらを N 個の ES に分散配置する．そして ES から任意の k 個以上のチャンクが取得できれば，元のファイルを復元可能である．

ユーザの要求したコンテンツのチャンクが利用可能な ES 上に k 個以上キャッシュされていない場合は, コアネットワークを介し, コンテンツプロバイダから取得する必要がある．各ファイルのチャンクに冗長性を追加し異なる ES にキャッシュすることで，ES が故障しやすい環境においても k 個以上のチャンクを ES から取得できる可能性が高まるため, ファイル取得の待ち時間, コアネットワークの輻輳, リモートサーバの負荷を軽減できる. [12] では ES から取得できるデータの総量が最大化するよう，各コンテンツの静的な要求頻度に対し各 ES に配置するチャンクを最適設計している．

表 1　DEC システムで用いる記号と定義

記号	定義
$[T]=\{1,\cdots,T\}$	タイムスロットの集合
$\mathcal{I}=\{1,\cdots,I\}$	人気ファイルの集合
h_i	ファイル i の要求レート
k	ファイル回復に必要なチャンクの下限値
$\mathcal{N}=\{1,\cdots,N\}$	ES の集合
p_n	ES n の信頼性パラメタ
$f_{n.t}$	サーバ n がタイムスロット t で使用できる確率
$a_{i,t}$	タイムスロット t のファイル i の可用性
a_t	ファイル i の時間累積可用性
$x_{i,n}$	サーバ n にキャッシュされたファイル i のチャンク数
$\mathbf{x}_i$	$n\in N$ に対する $x_{i,n}$ の集合
$\mathbf{x}$	$i\in I, n\in N$ に対する $x_{i,n}$ の集合
$y_i=\sum_{n\in N}x_{i,n}$	ファイル i のチャンク数

3.2 DEC システムにおけるキャッシュ配置問題 (CP)

[12] では, DEC システムにおけるキャッシュ配置問題 (CP) を離散最適化問題として定式化し, *HomoCP* (Homogeneous Version of CP) と呼ばれる CP のアルゴリズムを設計している．$x_{i,n}\in\{0,1\}$ を, ファイル i のチャンクが ES n に配置されているかどうかを表す変数とする．*HomoCP* では, ES は同じ容量と障害確率を持つため $s=s_n$，$p=p_n$ である．ES にキャッシュされたファイル i のチャンクの総数を y_i で表す．すなわち, $y_i=\sum_{n\in\mathcal{N}}x_{i,n}$ で表される．*HomoCP* では, ファイル i の可用性（ファイル i の使用可能なチャンクによってファイル i が回復できる確率）は y_i によって決定されるため, 最適な $y_i, i\in[N]$ を見つけることに焦点を当てている．ファイル i は, ファイル i の任意の k 個以上のチャンクによって回復できるため, ファイル i の可用性はファイル i の k 個以上のチャンクが利用可能な確率に等しい．よって与えられた y_i に対して, タイムスロット t におけるファイル i の期待可用性は次式で得られる．

$$A_t(y_i)=\begin{cases}\sum_{j=k}^{y_i}\binom{y_i}{j}(f_{n,t})^j(1-f_{n,t})^{y_i-j} & \text{if } y_i\geqq k,\\ 0 & \text{if otherwise.}\end{cases}\tag{1}$$

ここで, $\binom{y_i}{j}$ は, y_i 個から j 個選ぶ組み合わせ数である．

DEC システムの下では, ES 上に k 個以上チャンクがない場合, ユーザは混雑したコアネットワークを介してデータをダウンロードしなければならないため遅延が大きい．そこで DEC システムの目的は ES からダウンロードされるファイルの期待サイズを最大化することである．*HomoCP* の下で, ES からダウンロードされるデータサイズを以下に示す．

$$g(\mathbf{y})=\sum_{i\in\mathcal{I}}h_ia_i=\sum_{i\in\mathcal{I}}h_i\sum_{t\in[T]}A_t(y_i)=\sum_{i\in\mathcal{I}}h_iA(y_i)\tag{2}$$

ここで $A(y_i)$ は, ファイル i の時間累積可用性である.*HomoCP* の最適化問題は, 以下のように記述される．

$$\begin{aligned}\max_{y_i,i\in\mathcal{I}}\sum_{i\in\mathcal{I}}h_ia_i&=\sum_{i\in\mathcal{I}}h_iA(y_i)\\ s.t.\sum_{i\in\mathcal{I}}y_i&\leqq s\cdot N \qquad (HomoCP)\\ y_i\in\{0,1,\cdots,N\},&\forall i\in\mathcal{I}\end{aligned}$$

[12] では, 動的計画法（DP）に基づき Algorithm for *HomoCP*（*AHM*）と呼ばれるアルゴリズムを提案している.$w(z,j)$ を P1 の最適な目的値とする．

$$\begin{aligned}\max_{y_i,i\in[j]}\sum_{i\in[j]}h_iA_{y_i}&\\ s.t.\sum_{i\in[j]}y_i&\leqq z \qquad (P1)\\ y_i\in\{0,1,\cdots,N\},&\forall i\in[j]\end{aligned}$$

$w(z,j)$ は j 個のファイルが存在し, ES の合計容量が z チャンクである場合の *HomoCP* の最適な目的値である．また $r(z,j)$ は j 個のファイルが存在し ES の合計容量が z チャンクである場合の最適な選択肢を記録する．$w(z,j)$ の定義に基づくと, *HomoCP* の最適な目的値は $w(sN,I)$ に等しくなる. よって動的計画法を用いて，*HomoCP* の最適解が得られる．$w(z,j)$ と $r(z,j)$ の初期化 ($j=1$ の場合) を以下に示す．

$$w(z,1)=\begin{cases}h_1A(z) & \text{if } k\leqq z\leqq N\\ 0 & \text{if otherwise.}\end{cases}\tag{3}$$

$$r(z,1)=\begin{cases}z & \text{if } k\leqq z\leqq N\\ N.A. & \text{if otherwise.}\end{cases}\tag{4}$$

ファイルを回復できるのは, チャンクが k 個以上存在しキャッシュできるチャンクの最大数が N の場合であるため $r(z,1)$ は $z<K$，$N<z$ の場合，解が得られない．各 $j\in\{2,3,\cdots,I\}$

, $z = \{0, 1, \cdots, sN\}$ に対して, $w(z, j)$ と $r(z, j)$ は以下の漸化式で得られる.

$$w(z,j) = \max_{y_i} \{w(z - y_i, j-1) + h_j A(y_i)\} \quad (5)$$

$$r(z,j) = \operatorname*{argmax}_{y_i} \{w(z - y_i, j-1) + h_j A(y_i)\} \quad (6)$$

これらを用いた動的計画法による AHM のアルゴリズムを Algorithm 1 に示す. 最適な目的値である $w(sN, I)$ を取得した後, $r(z, j), 0 \leqq z \leqq sN, j \in [I]$ の記録をバックトラックすることで, 最適な解を得ることができる. *HomoCP* の DP ベースのアルゴリズムは, Algorithm1 で記述されている. 本稿では AHM アルゴリズムの解を, 目標チャンク数 T_m の設定値とする.

Algorithm 1 *AHM* for *HomoCP*

```
Set w(z,1) and r(z,1) based on (5) and (6);
for j = {2,3,···,I} do
    for z = {0,1,···,sN} do
        Set w(z,j) based on (7);
        Set r(z,j) based on (8);
    end for
end for
Set z = sN;
for i = {I, I−1,···,1} do
    Set y_i = r(z,j);
    Set z = z − r(z,i);
end for
Return y_i, i ∈ I as the optimal solution of HomoCP;
```

4. 提 案 方 式

既存研究では, コンテンツの人気度は一定と想定し ES へコンテンツの事前配置を行っていた. しかし実際の ES では, コンテンツの人気度は動的に変化するため LRU 等を用いた置換による動的制御が一般的に行われている. そこで本節では, ES の不稼働を考慮した上で, 低信頼な ES において LRU を用いた置換制御を想定した, ES からのコンテンツ取得可能性を向上させる erasure coding を用いた ES 挿入法を提案する.

4.1 目標ヒット率 $\hat{h}_m$ の導出

あるエリアに存在する N 個の ES へのチャンクキャッシュを想定し, 置換方式は LRU を使用する. 各 ES は, 同じ容量 s と同じ稼働率 r を想定する. 各コンテンツ m のチャンクは, 各 ES に 1 つまでキャッシュすることが可能であり, N 個の ES 全体で最大 sN 個のチャンクをキャッシュすることが可能である. 各コンテンツ m の N 個の ES 上にキャッシュされるチャンク数の期待値 T_m が, [12] の設計値に一致するよう, コンテンツ m のキャッシュへの挿入確率 f_m を事前に設定する. 各コンテンツ m の ES 上のキャッシュ数の期待値が T_m となる ES のコンテンツ m の目標ヒット率 $\hat{h}_m$ は次式で与えられる.

$$T_m = Nr\hat{h}_m \quad (7)$$

4.2 キャッシュ挿入確率 f_m の最適設計問題

各コンテンツ m の目標ヒット率 $\hat{h}_m$ と, 実際のヒット率 h_m との差が最小となるよう, キャッシュ挿入確率 f_m を最適設計する. 本設計問題で用いるパラメタを以下の表 2 にまとめる.

表 2 提案方式で用いる記号と定義

記号	定義
N	ES の個数
s	ES のストレージ容量
r	ES の稼働率
m	コンテンツ数
q_m	コンテンツ m の要求レート
T_{m}	コンテンツ m のキャッシュチャンク数の期待値
h_m	コンテンツ m のヒット率
$\hat{h}_m$	コンテンツ m の目標ヒット率
f_m	コンテンツ m のキャッシュ挿入確率

全体のコンテンツ取得成功率を向上させるには, 要求が多い高人気コンテンツほど, 目標ヒット率 $\hat{h}_m$ と実際のヒット率 h_m との差異を小さくすることが望ましい. そのため以下の最適化目標関数を定義する.

$$min \sum q_m (h_m - \hat{h}_m)^2 \quad (8)$$

実際のヒット率 h_m は, 文献 [13] で提案されている, 以下の近似式を用いて与える.

$$h_m = 1 - e^{-f_m q_m t_C} \quad (9)$$

ここでの t_C は, 任意のコンテンツが, キャッシュに挿入されてから, まったく要求されない状態で LRU キャッシュ内に存在し続けることができる時間のことであり, s をキャッシュ容量として次式を解くことで得られる.

$$\sum_{m \in M} 1 - e^{-f_m q_m t_C} = s \quad (10)$$

4.3 遺伝的アルゴリズムを用いた近似解の取得

前節で定義したキャッシュ挿入確率の最適化問題の厳密解を得るために要する時間は, コンテンツ数や ES 数, ES のストレージ容量の増加に伴い急激に増加する. そこで多くの最適化問題に対し良好な近似解が得られることで知られる遺伝的アルゴリズム (*GA*) を用いて, 本最適化問題の近似解を得る. 以下に GA を用いた近似解導出のアルゴリズムを示す.

(1) キャッシュ挿入確率 f_m として離散的な区間を考え, 全コンテンツに対して, 各コンテンツ m に対して設定する f_m の区間の値を要素としてもつベクトルを各遺伝子として考え, 初期の複数の遺伝子をランダムに生成

(2) 各遺伝子の適応度を (2) 式の逆数で算出し, 適応度の高いものをトーナメント選択により選択

(3) 選択した遺伝子をもとに次世代の遺伝子を生成

(4) 一定の確率で, 遺伝子内の f_m に対して交叉または突然変異を適用

(5) step2〜step4 を G 世代まで反復

(6) step5 で最後の世代まで反復し, 生成遺伝子の中で最も適応度の高い遺伝子をキャッシュ挿入確率 f_m として決定

5. 性 能 評 価

本節では, 各コンテンツ m の目標ヒット率 $\hat{h}_m$ と実際のヒット率 h_m を比較し, 提案方式を用いることで適切にキャッシュ

挿入確率 f_m が設定できていることを確認する．また N の二つの場合について，平均取得成功率（ES からのチャンク取得のみでコンテンツを復元できた割合）を提案方式を用いて f_m を設定した場合 ($f_m(GA)$) と，キャッシュミス時に常にコンテンツをキャッシュした場合 ($f_m(1.0)$) について，各 ES の稼働率 r を変化させた場合と，Zipf のパラメタ θ を変化させて異なる要求レートを与えた場合で比較する．

5.1 評価条件

コンテンツ数 m を 50，コンテンツの復元に必要なチャンク数の下限値 k を 5，各 ES の稼働率 r を 0.7〜0.9 までの 0.05 刻みで設定する．各 ES の容量 s を 6 に設定し，ES の個数 N を 20 と 30 に設定する．N が 20 の場合，各 ES の容量 s が 6 であることから，合計で 120 個のチャンクをキャッシュ可能である．パラメタ θ の Zipf 分布に従いコンテンツを要求させ LRU によってキャッシュの置換を行った．また遺伝的アルゴリズムのパラメタとして，個体数 200，世代数 200，交叉率 0.7，突然変異率 0.1，トーナメントサイズを 5 に設定した．

5.2 目標ヒット率 $\hat{h}_m$ と実際のヒット率 h_m の比較

図 1(a) に，GA で算出された各コンテンツ m の最適キャッシュ挿入確率 f_m を各コンテンツ m に対して人気の降順にプロットする．また図**??**に GA で算出された f_m を用いたときのコンテンツ m のヒット率 h_m と，目標ヒット率 $\hat{h}_m$ を，各々 m に対して同様にプロットする．m は，人気順を示しており，値が小さいほど人気度が高くなる．キャッシュ挿入確率 f_m を提案方式で最適化することで，実際のヒット率 h_m と目標ヒット率 $\hat{h}_m$ の差異は小さく，適切にキャッシュ挿入確率 f_m を設定できることが確認できる．また高人気コンテンツは要求頻度が高いため，f_m が小さくても高いヒット率を達成することができ，そのため f_m の設計値は m が小さい領域では m の増加に伴い増加する．

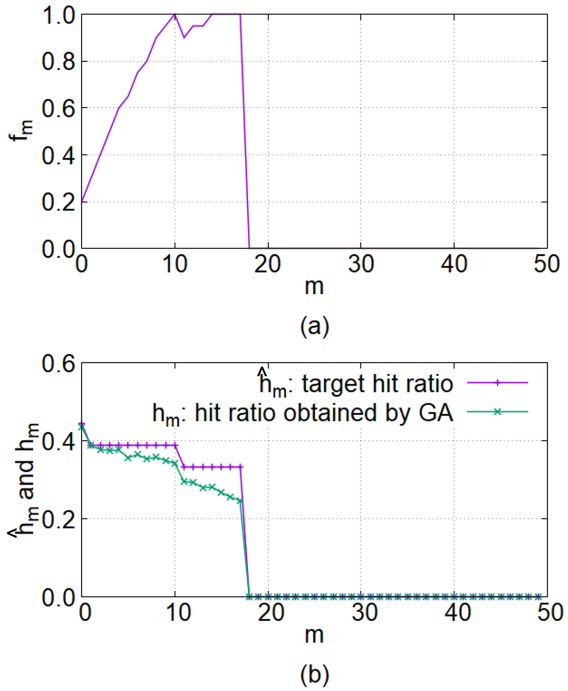

図 1　各コンテンツ m の (a) キャッシュ挿入確率 f_m と (b) 目標ヒット率 $\hat{h}_m$ と実際のヒット率 h_m

5.3 平均取得成功率

図 2 に $N = 20$，$N = 30$ の各々の場合について，全コンテンツの平均取得成功率 (ES からのチャンク取得のみでコンテンツを復元できた割合) を，提案方式を用いて f_m を設定した場合 ($f_m(GA)$) と，キャッシュミス時に常にコンテンツをキャッシュした場合 ($f_m(1.0)$) について，各々，各 ES の稼働率 r に対してプロットする．各 ES の稼働率 r が増加するにつれて，ES の故障確率は減少し，平均取得成功率はどちらの手法も増加する．提案方式を用いることで，単にコンテンツをすべて ES にキャッシュする場合と比較して，平均取得成功率を大きく改善できることが確認できる．

また図 3 に $N = 20$，$N = 30$ の各々の場合について，全コンテンツの平均取得成功率を同様にコンテンツ要求比率の分布を決める Zipf パラメタ θ に対してプロットする．どちらの方式も Zipf 分布の θ の値の増加に伴い，高人気コンテンツの要求比率が高くなる結果，平均取得成功率が増加する，これは，LRU を用いた置換制御を行ってため，人気の低いコンテンツのチャンクを優先的にキャッシュから削除するためである．また提案方式は既存研究をもとに目標チャンク数 T_m を設定し，設計値に近づけるようにキャッシュ挿入確率 f_m を最適設計しているため，平均取得成功率を大きく改善できることが確認できる．

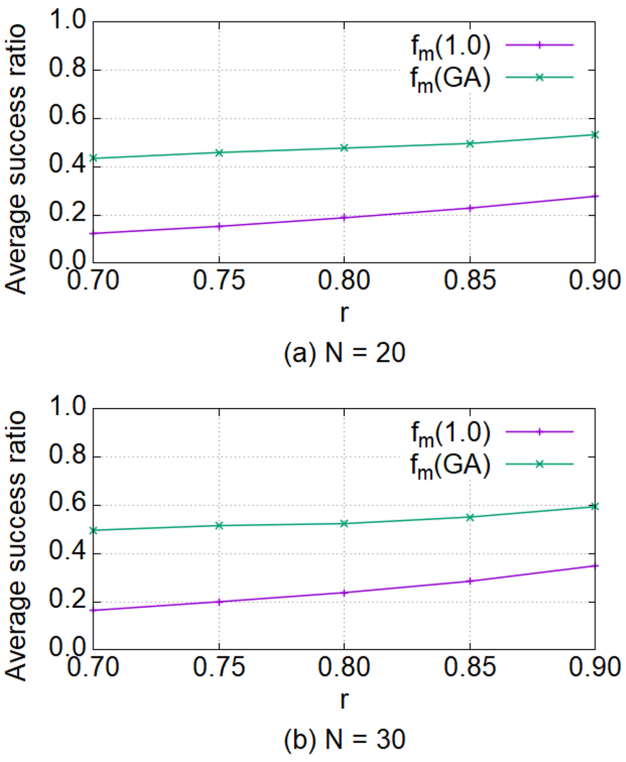

図 2　各 ES の稼働率 r に対する平均取得成功率

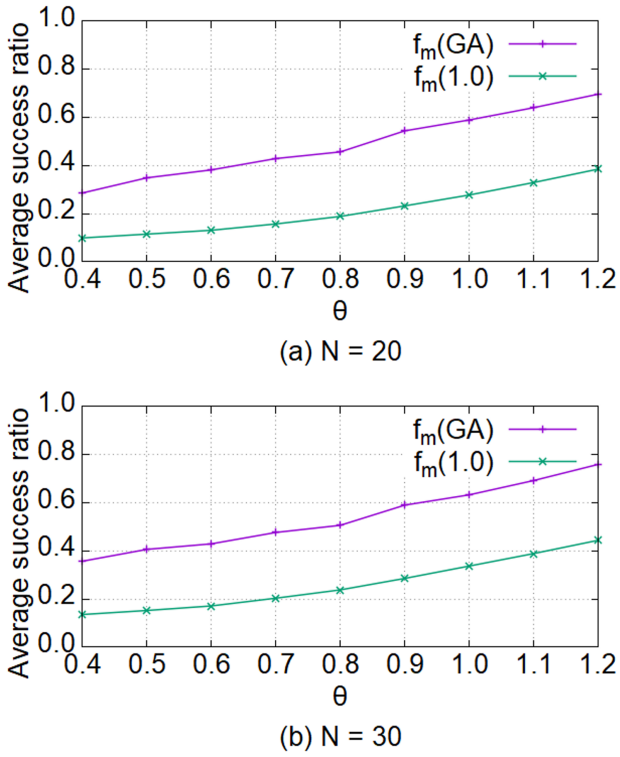

図 3　コンテンツの需要分布の Skew パラメタ θ に対する平均取得成功率

6. ま と め

5G では，トラフィックを削減するために無線の基地局にキャッシュサーバを設置しユーザに近い ES からコンテンツを配信する MEC が用いられている．しかし ES の設置数が膨大であることからコストの低い低信頼な ES が用いられており，故障発生による不稼働率の増加が課題である．低信頼な ES を想定した研究 [12] では，ES の信頼性の低さに対処するために，erasure coding を用いて ES の可用性を高める方式を提案しているが，コンテンツを ES 上に事前配置している．しかし実際の ES では LRU 等を用いた動的制御が一般的である．そこで，本稿では ES の不稼働率を考慮した上で，低信頼な ES において LRU を用いた置換制御を想定した ES へのチャンク挿入確率の，GA を用いた最適設計法を提案した．また計算機シミュレーションによる性能評価から以下のことを確認した．

- キャッシュ挿入確率 f_m を提案方式で最適化することで，実際のヒット率 h_m と目標ヒット率 $\hat{h}_m$ の差異を抑えた適切な f_m 設定が実現できることを確認した．
- キャッシュ挿入確率 f_m を提案方式で設定した場合 ($f_m(GA)$) と，キャッシュミス時に常にコンテンツをキャッシュした場合 ($f_m(1.0)$) で，平均取得成功率を比較し，提案方式を用いることで平均取得成功率を大きく改善することを確認した．

謝辞　本研究成果は，JSPS 科研費 21H03436 と 21H03437 の助成を受けたものである．ここに記して謝意を表す．

文　献

[1] Y. Zeng, Y. Huang, J. Liu, and Y. Yang, "Privacy-preserving distributed edge caching for mobile data offloading in 5g networks," IEEE ICDCS 2020

[2] L. Ramaswamy, L. Liu, and A. Iyengar, "Cache clouds: Cooperative caching of dynamic documents in edge networks," IEEE-ICDCS 2005

[3] Y. Liu, X. Shang, and Y. Yang, "Joint sfc deployment and resource management in heterogeneous edge for latency minimization," IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 8, pp. 2131–2143, 2021

[4] X. Shang, Z. Liu, and Y. Yang, "Online service function chain placement for cost-effectiveness and network congestion control," IEEE Transactions on Computers, pp. 1–1, 2020

[5] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," IEEE Transactions on Information Theory, vol. 56, no. 9, pp. 4539–4551, 2010

[6] V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh, "Asymptotic interference alignment for optimal repair of mds codes in distributed storage," IEEE Transactions on Information Theory, vol. 59, no. 5, pp. 2974–2987, 2013

[7] D. Bindel, et al., "Oceanstore: An extremely wide-area storage system," in Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems. Citeseer, 2000, pp. 190–201

[8] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, "Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster," IEEE International Conference on Cluster Computing 2010

[9] A. Soro and J. Lacan, "Fnt-based reed-solomon erasure codes," IEEE Consumer Communications and Networking Conference 2010

[10] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," ACM symposium on Operating systems principles 2003

[11] D. Ford, F. Labelle, F. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," USENIX 2010

[12] Y. Liu, et al., "Distributed Cooperative Caching in Unreliable Edge Environments," IEEE INFOCOM 2022

[13] H. Che, et al., "Hierarchical Web Caching Systems: Modeling, Design and Experimental Results," IEEE J. Selected Areas of Commun., vol.20, no.7, Sep. 2002

[14] L. Ramaswamy, L. Liu, and J. Zhang, "Efficient formation of edge cache groups for dynamic content delivery," IEEE ICDCS 2006