University of Naples Federico II

Dep. of Mathematics and Applications "R. Caccioppoli"

Optimization for Networks and Object Recognition

Author: Giuseppe Vettigli

Supervisor: Paola Festa
Co-Supervisor: Francesco Isgrò
Antonia M. Tulino

A thesis submitted for the degree of

Doctor of Philosophy

Naples, February 2018

Alla mia famiglia
e a tutte le persone
che credono e
hanno fiducia in me.

La vita non è una dura lotta contro gli altri
ma contro se stessi. Non si ha il bisogno di
dimostrare qualcosa a qualcuno eccetto che
a se stessi. Solo quando capirete questo
diventerete persone migliori in grado
di aiutare gli altri ma soprattutto voi stessi.

Acknowledgements

La sezione relativa ai ringraziamenti penso sia molto delicata e difficile da scrivere, questo per il semplice fatto che si prova a spiegare a determinate persone quanto siano importanti per la tua vita e, quanto tu sia loro grato per tutto quello che hanno fatto durante questo lungo cammino. Fatto questo breve preambolo, possiamo passare ai veri e propri ringraziamenti, voglio innanzitutto ringraziare mamma e papà per essermi stati sempre vicini, per aver creduto in me e per avermi dato il coraggio di affrontare la scalata di una montagna che ho sempre ritenuto insormontabile, grazie a mio fratello Luigi perchè in silenzio mi ha ammirato (forse anche troppo) e appoggiato, ignaro di quanto io ammiri lui per la forza, il coraggio e la tenacia; grazie ai miei zii Anna Vettigli, Franco Ipomeo, Pasquale Vitale, Rosaria Candurro, Valentino Vettigli e, ai miei cugini Anna Franzese, Anna Ipomeo, Antonio Ranieri, Giuseppe Ambrosio, Ivano Vettigli, Lucia Maffettone, Luigi Ipomeo, Rachele Franzese, Raffaele Storto Marinaccio, i quali mi sono stati sempre vicino, mi hanno sostenuto, consigliato e aiutato nei momenti difficili. Un grazie al Dottor Stefano Pappalardo e a sua moglie Tiziana Bassetti, i quali mi hanno fatto crescere non solo professionalmente ma anche mentalmente, grazie di esservi sempre preoccupati per me, di essermi stati sempre vicini, in particolare, voglio ringraziare Stefano per avermi fatto superare momenti della mia vita particolarmente difficili e per l'immensa pazienza. Grazie al mio amico Guido Bassetti, una persona che si è sempre interessata al mio lavoro e alla mia vita, mi ha dato consigli e ha ascoltato i miei sfoghi e le mie ire e, grazie a Ros, persona che mi elogia per il mio modo di fare il caffè. Un grazie ai miei cari amici Claudio Pipicelli, persona gentilissima e premurosa, Giovanni Ferrarese, un amico che mi è stato vicino

dalla magistrale e, con il quale ho avuto molte avventure non solo universitarie, grazie a Massimiliano Di Mella, una persona capace di implementare di tutto, con un qualsiasi linguaggio di programmazione e che mi ha sempre osannato e sopravvalutato, sempre disponibile e pronto a venire in mio soccorso in qualsiasi momento e dovunque, grazie a Sabatino Daniele Iovino, una persona sempre presente, in pratica non mi lascia mai da solo. Grazie a Marialuigia Malgeri Manzo e a Valentina Malgeri Manzo per le loro premure e per l'immensa forza che trasmettono. Grazie ai miei colleghi, nonché amici e, compagni di viaggio Andrea Apicella, una persona limpida, trasparente, sincera e leale, Andrea credimi, è stato davvero un onore lavorare con te, grazie ad Antonio Napoletano e Marco Viola, i quali hanno sempre creduto in me e, per tale ragione, mi hanno fatto capire che ero all'altezza di sostenere un dottorato di ricerca, grazie a Fulvio Maddaloni, un artista incompreso che ama da morire la matematica, in particolare, l'algebra teorica, lui è stato il mio "professore di matematica" e vive la musica allo stesso mio modo, grazie a Simone Celestino, il quale, durante la preparazione degli esami che lungo il nostro cammino abbiamo sostenuto, ha dato direttive fondamentali e di notevele importanza, grazie a Tommaso Pastore, una persona che ama in modo particolare la ricerca e che con semplici parole mi ha fatto sempre capire cosa fosse giusto e cosa fosse sbagliato, grazie ad Antonio Di Stasio, persona un pò particolare, ma con la quale ho condiviso pensieri profondi non solo di natura universitaria. Un grazie particolare va ai miei advisor persone con un cuore grande, che hanno scommesso su di me, non hanno mai esitato un solo istante a darmi fiducia, coraggio, a supportarmi e a sopportarmi; grazie di cuore alla Professoressa Paola Festa, la quale è una sincera amica, una linea guida, un punto di riferimento, grazie di cuore al Professor Francesco Isgrò, il quale è un amico, un fratello, mi ha tirato molte volte fuori dal baratro, gli bastava un semplice sguardo per capire cosa io pensassi o cosa stessi per dire, grazie di cuore alla Professoressa Anna Corazza, una persona molto precisa, puntuale, che non ama giochi di parole ma che allo stesso tempo ti sorride e ti guida,

grazie di cuore alla Professoressa Antonia Maria Tulino, una persona esplosiva, che ama da morire il suo lavoro, dona l'anima, non solo a se stessa, ma soprattutto alle altre persone, segue gli studenti, li valorizza, certo a volte assume un atteggiamento duro ma ha un cuore profondo e gentile. Grazie ad Antonia ho avuto una grossa e importantissima opportunità, quella di lavorare ai *NOKIA Bell Labs* (Holmdel, NJ, USA), un'esperienza indimenticabile che mi ha molto segnato. Durante il mio soggiorno negli USA ho conosciuto persone davvero stupende che meritano di essere citate in questa sezione, grazie al Dottor Jaime Llorca, il quale è una persona molto semplice e gentile, si è preso cura di me, mi ha aiutato sia con l'inglese, sia con un progetto su cui abbiamo collaborato insieme, mi ha fatto vivere dei momenti bellissimi, è un vero amico, Jaime è stato davvero un onore e un piacere lavorare con te; grazie al Dottor Cristian Czegledi, il mio "professore di inglese", con lui siamo stati in diversi posti, abbiamo ammirato svariati paesaggi, abbiamo parlato, mi ha aiutato ad uscire dalla mia "timidezza" e a non aver paura di esprimere il mio pensiero; grazie al Dottor Hao Feng, un genio incompreso, una persona profonda dal nobile cuore, grazie al Dottor Chang-Heng David Wang, persona molto riservata ma che con me si è lasciata andare; grazie di cuore a Susan Feingold, una vera mamma, mi ha ospitato in casa sua e mi ha trattato come un figlio.
Grazie di cuore a tutti voi per l'immensa stima e la grande fiducia che avete riposto in me e, soprattutto, grazie per esserci stati e per esserci tuttora, senza di voi, non solo non avrei raggiunto questo importantissimo traguardo ma, non sarei nemmeno diventato una persona migliore.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

The present thesis explores two different application areas of combinatorial optimization, the work presented here, indeed, is two fold, since it deals with two distinct problems, one related to data transfer in networks and the other to object recognition. This Chapter is organized as follow: the Section 1.1 provides an overview of the optimization methods, Section 1.2 and Section 1.3 provide an overview of the coding and caching in networks and object recognition problem, Section 1.4 provides a brief description of the thesis work contributions, finally, Section 1.5 provides a dissertation outline.

1.1 Optimization Methods

The algorithms used to solve optimization problems are divided in 3 categories:

- *Exact Algorithms* that provide an optimal solution to a combinatorial optimization problem, but take an exponential number of iterations when the problem is intractable. They include *cutting-planes*, *branch-and-bound*, and *dynamic programming*;

- *Approximation Algorithms* that provide a solution, provably close to optimal, to a combinatorial optimization problem. The approximation ratio of an algorithm is the ratio between the result obtained by the approximation algorithm and the optimal cost or profit. An algorithm with approximation ratio n is called a n-approximation algorithm. Approximation algorithms

are typically used either, when finding an optimal solution is intractable or a exact solution is not needed;

- *Heuristic Algorithms* that provide a suboptimal solution, but without a guarantee on its quality. A heuristic algorithm is any *fast* (polynomial) algorithm that finds a feasible solution. Heuristic algorithms can be classified in the following way:
  - Constructive: start from an empty solution and iteratively add new elements to the current partial solution until a complete solution is found;
  - Local Search: start from an initial feasible solution and iteratively try to improve it by slightly modifying it, stop when local optimum is found;
  - Meta-heuristics: an evolution of local search algorithms which avoid to find local optima using specific techniques.

In our case we have used heuristic and exact methods to address the problems studied in this thesis.

1.2 Index Coding and Caching

Caching is an essential technique to improve throughput and latency in a vast variety of applications. The core idea is to duplicate content in memories distributed across the network, which can then be exploited to deliver requested content with less congestion and delay. In particular, it has been shown that the use of caching together with smart offloading strategies in a RAN composed of evolved NodeBs (eNBs), AP (e.g., WiFi), and UEs, can significantly reduce the backhaul traffic and service latency. The traditional role of cache memories is to deliver the maximal amount of requested content locally rather than from a remote server. While this approach is optimal for single-cache systems, it has recently been shown to be, in general, significantly suboptimal for systems with multiple caches (i.e., cache networks) since it allows only additive caching gain, while instead, cache memories should be used to enable a multiplicative caching gain. Recent studies have shown that storing different portions of the content across the

wireless network caches and capitalizing on the spatial reuse of device-to-device (D2D) communications [53, 49], or exploiting globally cached information in order to multicast coded messages simultaneously useful to a large number of users [58, 56, 29, 67, 64, 77, 83], enables a global caching gain. In particular, the recently proposed schemes in [53, 49, 54, 58, 56, 29, 67, 64, 77, 83] exhibit overall network throughputs that are proportional to the aggregate cache capacity, such that, when compared with today's local caching and unicast transmission policies, gains on the order of the number of users per cell (e.g., more than 100x in urban areas) are possible. While these initial results are promising, the existing literature on wireless caching and delivery (i) lacks understanding of fundamental limits beyond two basic network structures, namely the D2D network [53, 49] and the shared link network [53, 49, 54, 58, 56, 29, 67, 64, 77, 83], (ii) does not exploit properties of video signals such as scalability and correlation for improved performance and robustness, (iii) does not take into account the losses and variations in the wireless channel, (iv) lacks practical schemes in the finite file size and finite delay regime. We focus on the case of a single server (e.g., a base station) and multiple users, each of which caches segments of files in a finite library. Each user requests one (whole) file in the library and the server sends a common coded multicast message to satisfy all users at once. The problem consists of finding the smallest possible codeword length to satisfy such requests. To solve this problem we present two achievable caching and coded delivery scheme, and one correlation-aware caching scheme, each of them is based on a heuristic polynomial-time coloring algorithm.

1.3 Object Recognition

At the core of this ability there is the solution of an object recognition task.

Automatic object recognition has become, over the last decades, a central toping the in the artificial intelligence research, with a a significant burt over the last new year with the advent of the deep learning paradigm.

In this context, the objective of the work discussed in the last two chapter of this thesis is an attempt at improving the performance of a natural images classifier introducing in the loop knowledge coming from the real world, expressed in terms of probability of a set of spatial relations between the objects in the images. In

different words, the framework presented in this work aims at integrating the output of standard classifiers on different image parts with some domain knowledge, encoded in a probabilistic ontology.

Standard ontologies are very much used for encoding a-priori information on the application domain, but they do not perform very well when dealing with real world uncertainty.

Probabilistic ontologies aim at filling this gap by associating probabilities to the coded information, and provide then an adequate solution to the issue of coding the context information necessary to correctly understand the content of an image. Such information is then combined with the classifier output in order to correct possible classification errors on the basis of surrounding objects.

In the framework proposed a probabilistic ontology is exploited for trying to improve the performance of a classifier, by integrating the ontology with a probabilistic model. This work presents two main aspects of novelty. The former is the use of a probabilistic ontology for the solution of a computer vision problem. The latter element of novelty lies in the integration of a probabilistic ontology with a probabilistic model.

1.4 Synopsis

In **Part I**, we consider a shared link caching network, which is formed by a single source node (a server or base station) with m files, connected via a shared noiseless link to n user nodes, each with cache of size M files. For the shared link structure, the performance metric is the number of time slots necessary to satisfy all the demands. In Chapter 4, we first extend the analysis of the achievable scheme in [51] to the case of heterogeneous cache sizes and popularity distributions, providing an upper bound on the limiting average performance where the number of packets goes to infinity while the remaining system parameters are kept constant. We then focus on finite regimes and show how the scheme achieving this upper bound can very quickly loose its multiplicative caching gain for finite content packetization. To overcome this limitation, we design a novel polynomial-time algorithm based on greedy graph-coloring that, while keeping the same finite content packetization, recovers a significant part of the multiplicative caching gain.

In Chapter 5, we first extend the analysis of the achievable scheme in [51] to the case of heterogeneous cache sizes and demand distributions, providing an upper bound on the limiting average performance where the number of packets goes to infinity while the remaining system parameters are kept constant. We then focus on finite regimes and show how the scheme achieving this upper bound can very quickly loose its multiplicative caching gain for finite content packetization. To overcome this limitation, we design a novel polynomial-time algorithm based on greedy graph-coloring that, while keeping the same finite content packetization, recovers a significant part of the multiplicative caching gain. In Chapter 6, we first provide an overview of the emerging caching-aided coded multicast technique, including state of art schemes and their theoretical performance. We then focus on the most competitive scheme proposed to date and describe a fully working prototype implementation in CorteXlab [28], one of the few experimental facilities where wireless multiuser communication scenarios can be evaluated in a reproducible environment. We use our prototype implementation to evaluate the experimental performance of state-of-the-art caching-aided coded multicast schemes compared to state-of-the-art uncoded schemes, with special focus on the impact of coding computation and communication overhead on the overall bandwidth efficiency performance. Our experimental results show that coding overhead does not significantly affect the promising performance gains of coded multicasting in small-scale real-world scenarios, practically validating its potential to become a key next generation 5G technology. In last Chapter (7) of the first Part, we propose the use of video data present in persistent receiver-side memories as distributed side information for video coding. We utilize index coding to efficiently transmit coded messages that satisfy the new video demanded by multiple receivers.

In **Part II**, we present a framework which aims at integrating the output of standard classifiers on different image parts with some domain knowledge, encoded in a probabilistic ontology. In fact, while standard ontologies are quite widespread as a means to manage a-priori information, they fail in the important task of dealing with real world uncertainty. Probabilistic ontologies aim at filling this gap by associating probabilities to the coded information, and provide then an adequate

solution to the issue of coding the context information necessary to correctly understand the content of an image. Such information is then combined with the classifier output in order to correct possible classification errors on the basis of surrounding objects. The framework presented aims at determining a set of keywords describing the content of an image and the relations existing among them. The framework presents two main aspects of novelty. First, the use of a probabilistic ontology for a computer vision problem has, at the best of our knowledge, never been proposed before. A second element of novelty is the integration of a probabilistic model with a probabilistic ontology.

1.5 Dissertation Outline

This dissertation is organized in two Parts. **Part I** is related to data transfer optimization in network and it is organized as follows: Chapter 3 provides an overview of the caching-aided coded multicast technique, including state of the art schemes and the network model used in our works. Chapter 4 and Chapter 5 provide a description of two achievable caching and coded delivery scheme, along with the general upper bound of the average achievable rate. Each scheme is based on a proposed polynomial-time coloring algorithm. In Chapter 6 we provide a fully working prototype implementation of the most competitive caching and coded delivery scheme. Then, in Chapter 7, we propose an achievable caching and coded delivery scheme based on library correlation-aware. **Part II** is related to object recognition and it is organized as follows: Chapter 9 provides a description of the different modules of the our system, with a few details about the probabilistic ontology, and to the model adopted to combine classification and ontology probabilities.

Part I

Networks Problems

Chapter 2

Introduction

In this chapter, we consider a shared link caching network, which is formed by a single source node (a server or base station) with m files, connected via a shared noiseless link to n user nodes, each with cache of size M files. For the shared link structure, the performance metric is the number of time slots necessary to satisfy all the demands. In the case of symmetric links, the number of time slots can be normalized by the number of times lots necessary to send a single file across a point to point link. The performance metric is *rate* defined as in the index coding setting [8, 7, 22, 63, 11, 17, 48, 46, 2, 82], i.e., the number of equivalent file transmissions.

2.1 An Efficient Coded Multicasting Scheme Preserving the Multiplicative Caching Gain

Consider a network with one source (base station), having access to m files, and n users (caches), each with a storage capacity of M files. In [50], the authors showed that if the users can communicate between each other via Device-to-Device (D2D) communications, a simple distributed random caching placement scheme and TDMA-based unicast D2D delivery achieves the order-optimal[1] throughput $\Theta\left(\max\{\frac{M}{m}, \frac{1}{m}, \frac{1}{n}\}\right)$,[2] whose linear scaling with M when $Mn \geq m$ shows a re-

[1] Order-optimal means that the gap between the information theoretic converse and the achievable throughput can be bounded by a constant number when $m, n \to \infty$.

[2] Given two functions f and g, we say that: 1) $f(n) = O\left(g(n)\right)$ if there exists a constant c and integer N such that $f(n) \leq cg(n)$ for $n > N$ 2) $f(n) = \Theta\left(g(n)\right)$ if $f(n) = O\left(g(n)\right)$ and $g(n) = O\left(f(n)\right)$.

markable caching potential. Moreover, in this scheme each user caches entire files without the need of partitioning files into packets,and missed files are delivered via unicast transmissions from neighbor nodes, making it practically implementable in real scenarios.

In the case that users cannot communicate between each other, but share a multicast link from the content source, the authors in [68] presented a deterministic caching and coded multicasting scheme achieving the same order-optimal throughput as in the D2D caching network for the worst-case demand setting. However, the scheme in [68] requires a centralized caching policy and each file to be partitioned into a number of packets that grows exponentially with the number of users. In [65], the authors presented an alternative scheme for the same network that uses a simpler decentralized random caching policy while a more complex coded multicasting scheme requiring a number of computations that grows exponentially with the number of users. Nonetheless, to guarantee the same throughput, the file size (or equivalently the number of packets per file) is required to go to infinity.

In [51], the authors considered the same shared link network under random demands characterized by a popularity distribution, and proposed a scheme consisting of a distributed random popularity-based (RAP) caching policy and a chromatic-number index coding (CIC) based multicasting scheme, referred to as RAP-CIC, proved to be order-optimal in terms of average throughput. In order to analytically quantify the performance of RAP-CIC, the authors in [51] resorted to a polynomial-time approximation of CIC, referred as greedy constrained coloring (GCC) that guarantees the order-optimal throughput given an infinite number of packetizations. RAP-GCC is also shown to achieve the same performance as the algorithm in[65] for the worst-case demand setting.

It is then of key importance to understand if using any of above mentioned schemes, the promising linear throughput scaling with cache size (multiplicative caching gain) can be preserved in practical settings with finite file packetization. In this Chapter, we first extend the analysis of the achievable scheme in [51] to the case of heterogeneous cache sizes and popularity distributions, providing an upper bound on the limiting average performance where the number of packets goes to infinity while the remaining system parameters are kept constant. We then focus on finite regimes and show how the scheme achieving this upper bound can very

quickly loose its multiplicative caching gain for finite content packetization. To overcome this limitation, we design a novel polynomial-time algorithm based on greedy graph-coloring that, while keeping the same finite content packetization, recovers a significant part of the multiplicative caching gain.

2.2 An Efficient Multiple-Groupcast Coded Multicasting Scheme for Finite Fractional Caching

In a shared link network, where all the users, each of which only makes one request, share a multicast link from the content source, the authors in [68] presented a deterministic caching and coded multicasting scheme achieving the order-optimal throughput $\Theta\left(\max\{\frac{M}{m}, \frac{1}{m}, \frac{1}{n}\}\right)$ [3] for the worst-case demand setting. However, the scheme in [68] requires a centralized caching policy and each file to be partitioned into a number of packets that grows exponentially with the number of users. In [65], the authors presented an alternative scheme for the same network that uses a simpler decentralized random caching policy while a more complex coded multicasting scheme requiring a number of computations that grows exponentially with the number of users. Nonetheless, to guarantee the same throughput, the file size (or equivalently the number of packets per file) is required to go to infinity. In [52], the authors extended these results to the case when each user make $L = \{1, \cdots, m\}$ requests and showed that the coloring based scheme used in [68, 65] is not sufficient to capture the order-optimality. Instead, the local coloring scheme [76] can achieve the order-optimality for the multiple requests scenario.

In [51], the authors considered the same shared link network, where each user requests only one file, under random demands characterized by a demand distribution, and proposed a scheme consisting of a distributed random popularity-based (RAP) caching policy and a chromatic-number index coding (CIC) based multicasting scheme, referred to as RAP-CIC, proved to be order-optimal in terms of average throughput. In order to analytically quantify the performance of RAP-CIC, the authors in [51] resorted to a polynomial-time approximation of CIC,

[3] Given two functions f and g, we say that: 1) $f(n) = O(g(n))$ if there exists a constant c and integer N such that $f(n) \leq cg(n)$ for $n > N$ 2) $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $g(n) = O(f(n))$.

referred as greedy constrained coloring (GCC) that guarantees the order-optimal throughput given an infinite number of packetizations. RAP-GCC is also shown to achieve the same performance as the algorithm in[65] for the worst-case demand setting.

It is then of key importance to understand if using any of above mentioned schemes, the promising linear throughput scaling with cache size (multiplicative caching gain) can be preserved in practical settings with finite file packetization. In this Chapter, we first extend the analysis of the achievable scheme in [51] to the case of heterogeneous cache sizes and demand distributions, providing an upper bound on the limiting average performance where the number of packets goes to infinity while the remaining system parameters are kept constant. We then focus on finite regimes and show how the scheme achieving this upper bound can very quickly loose its multiplicative caching gain for finite content packetization. To overcome this limitation, we design a novel polynomial-time algorithm based on greedy graph-coloring that, while keeping the same finite content packetization, recovers a significant part of the multiplicative caching gain.

2.3 Coding for Caching in 5G Networks

Along with the Internet revolution, IP traffic is growing at a tremendous pace and it is expected to reach two zettabytes per year by 2019. Mobile data networks are envisioned to support up to 14% of this global data traffic coming from a plethora of different market segments. Among these segments, multimedia streaming is the service with the highest penetration rate, having the major impact on the overall traffic increase. On the other hand, current mobile network generations cannot cope with this explosive traffic growth due to the capacity limitations of radio access, backhaul, and core mobile networks, and the increasingly unicast and on-demand nature of users' content demands. In order to support this traffic expansion, the fifth generation (5G) of mobile networks is under preparation. Among the key performance challenges that 5G needs to address are: throughput, latency, and energy efficiency. That is, 5G is expected to provide 1000x higher throughput, sub-millisecond service latencies, and up to 90 percent overall

energy savings [24]. Despite the myriad of technological advances at the physical (PHY) and medium access control (MAC) layers (e.g., inter-cell interference coordination (ICIC), massive multiple-input-multiple-output (MIMO), carrier aggregation), targeted data rates are still significantly out of reach. To this end, 5G envisions novel architectural components for the next generation radio access network (RAN), including small cell densification, efficient wireless backhauling, and network self-organization [24]. In this context, the use of inexpensive storage resources within the RAN is emerging as a promising approach to reduce network load, and effectively increase network capacity.

2.3.1 Prominence of Wireless Caching in 5G

Wireless caching, i.e., caching content within the wireless access network is gaining interest, specially in ultra-dense networks where many connected devices try to access various network services under latency, energy efficiency, and/or bandwidth limitation constraints [24]. Proactively caching content items at the network edge (e.g., at the RAN) helps in relieving backhaul congestion and meeting peak traffic demands with lower service latency as Fig. 6.1 illustrates. For maximum benefits, network operators can intelligently exploit users' context information, classify content by popularity, and improve predictability of future demands to proactively cache the most popular content before being requested by end users. Such a strategy is able to fulfill the quality of service (QoS) requirements while significantly reducing the use of bandwidth resources and its associated energy consumption. Content items can be cached at different locations of the mobile network. Within the RAN, base stations (or small base stations), user equipment (UE) devices, and access points (AP) can be enhanced with additional memory for content caching. While caching can also happen within the evolved packet core (EPC), the main benefit of caching at the EPC is to reduce peering traffic between internet service providers (ISP). It is the additional deployment of cache memories within the RAN that can crucially help minimizing intra-ISP traffic, relieving backhaul load, and reducing service latencies [32].

2.3.2 From Uncoded to Coded Content Distribution

A substantial amount of recent studies have analyzed the use of wireless caching as a promising solution for 5G. Among these studies, [31] introduced the idea of femtocaching and addressed the question of which files should be assigned to which helper nodes (femtocell-like base stations), while [25, and reference therein] considered the improvement in caching efficiency that can be obtained by dynamically learning content popularity and updating cache contents at the network edge. Despite considerable interest, such studies focus on the data placement problem in isolation, assuming the use of unicasting or naive (uncoded)[4] multicasting during transmission, and hence ignoring the potential benefits of joint placement and transmission code design.

In [26], the data placement problem is generalized to the coded content distribution problem where the goal is to jointly determine the placement and routing of (possibly coded) information over the network, showing that joint code design significantly increases multicast efficiency, leading to substantial improvements in reducing network load and access latencies. A number of information-theoretic studies have then characterized the order-optimal performance of a caching network of special practical interest, the shared link caching network, formed by a single source node (e.g., base station) with access to a library of content files connected via a shared multicast link to multiple user nodes (e.g., end devices or access points), each with caching capabilities. In this context, the work in [66] showed that under worst-case demands, caching portions of each file uniformly at random and using index coding (IC) [12] during transmission yields an overall load reduction that is proportional to the aggregate cache size. In [30], the authors analyzed the case in which user demands follow a Zipf popularity distribution, designing order-optimal achievable schemes[5] that adjust the caching distribution as

[4]The term uncoded is used to refer to a scheme in which, at each use of the channel, the transmission is composed of packets that belong to the same file, while the term coded refers to a scheme in which transmissions can be composed of a mixture of packets from different files.

[5]An achievable scheme is said to be order-optimal if, as the file size goes to infinity, the number of transmissions needed to satisfy the user demands scales as the information theoretic optimal number of transmissions needed to satisfy the user demands; i.e the ratio between the achievable and optimal number of transmissions is upper bounded by a constant independent of all the system parameters.

a function of the system parameters to balance the gains from local cache hits and coded multicasting. Shortly after, [27] showed that the gains achieved by these schemes require a number of packets per requested item that grows exponentially with the number of caches in the system, leading to codes of exponential complexity that compromise their theoretical gains. Efficient polynomial-time schemes (e.g., [84]) have then been proposed to recover a significant part of the promising multiplicative caching gain.

In terms of practical implementations, the work in [25] provided a big data platform where learning algorithms can be used to predict content popularity and drive caching decisions, but the benefit of the learning techniques on improving caching efficiency is evaluated via numerical simulations. In addition, only conventional uncoded schemes are considered, and aspects related to advanced coding techniques such as caching-aided coded multicasting that can potentially provide much larger gains are largely overlooked. It is also important to note that, so far, only information-theoretic studies have shown the potential gains of such schemes, and the emulation work in [71] only considers $2-4$ users and 3 files, a very limited scenario that does not allow showing the real impact of computational complexity and coding overhead. Moreover, it is unclear whether existing schemes meet the requirements of current technologies, thus leaving plenty of open questions regarding practical performance benefits. In this Chapter, we first provide an overview of the emerging caching-aided coded multicast technique, including state of art schemes and their theoretical performance. We then focus on the most competitive scheme proposed to date and describe a fully working prototype implementation in CorteXlab [28], one of the few experimental facilities where wireless multiuser communication scenarios can be evaluated in a reproducible environment. We use our prototype implementation to evaluate the experimental performance of state-of-the-art caching-aided coded multicast schemes compared to state-of-the-art uncoded schemes, with special focus on the impact of coding computation and communication overhead on the overall bandwidth efficiency performance. Our experimental results show that coding overhead does not significantly affect the promising performance gains of coded multicasting in small-scale real-world scenarios, practically validating its potential to become a key next generation 5G technology.

2.4 Video Coding Using Receiver-Side Memory and Index Coding

Conventional video coding uses motion compensated block based interframe prediction and transform coding with the Discrete Cosine Transform (DCT) or similar transforms. This assumes the presence of a decoded picture buffer (DPB) at the encoder and decoder that contains a set of frames used to search for the best matching reference block(s) to the block in the current frame. These blocks are used to produce a predictor for the current block, and the motion vector and frame index in the DPB are encoded to identify the reference block. The prediction residual is transformed, quantized and entropy coded. DPBs store only a few whole frames(usually 32 or less) that are retained in memory for a relatively short period of time. A frame at the start of a group of pictures or one which cannot be efficiently encoded by interframe prediction (for example, due to a scene change) is encoded by intra-prediction i.e. a block is encoded using a prediction signal from neighboring blocks that are likely to be similar. An intracoded frame requires many more bits to encode than an inter-coded frame.

Digital Video Recorders (DVRs) are commonly used to record and store video downloaded from cable and satellite sources. These represent a large receiver-side memory cache that contains video data which is similar to videos that are likely to be downloaded by the consumer in the future. For example, a viewer may watch weekly episodes of a TV serial while previous episodes are still in the DVR, or recurring sports events at a given venue during a series. This presents an opportunity to exploit textures, images etc. that occur frequently in a particular video sequence, such as a TV serial, as reference signals when similar video is downloaded in the future. These signals may occur too far apart to be retained in a conventional DPB. For example, a scene may occur in an episode and may recur in a later episode. Receiver-side memories commonly occur in PCs, smartphones, digital TVs and other consumers of video, and can use the same principle.

Using such long range reference predictors should be especially useful for frames that are conventionally intra-coded: these may be encoded similar to conventional inter-coded frames using as reference the frames from previous episodes stored in the DVR. This will result in bit rate comparable to an inter-coded frame. However

such predictors should also be useful to improve the coding efficiency of inter-coded frames due to better matches between the predictor signal and the pixel blocks being predicted. In this Chapter, we propose the use of video data present in persistent receiver-side memories as distributed side information for video coding. We utilize index coding to efficiently transmit coded messages that satisfy the new video demanded by multiple receivers.

2.5 Part I Outline

The first part of this dissertation is organized as follows.

In Chapter 3 we provide an overview of the caching-aided coded multicast technique, including state of the art schemes and the network model used in our works. In Chapter 4 and in Chapter 5 we propose two achievable caching and coded delivery scheme, along with the general upper bound of the average achievable rate. Each scheme is based on a proposed polynomial-time coloring algorithm. In Chapter 6 we provide a fully working prototype implementation of the most competitive caching and coded delivery scheme. Then, in Chapter 7, we propose an achievable caching and coded delivery scheme based on library correlation-aware.

Chapter 3

Caching-Aided Coded Multicast

The use of caching together with smart offloading strategies in a RAN composed of evolved NodeBs (eNBs), AP (e.g., WiFi), and UEs, can significantly reduce the backhaul traffic and service latency. In this context, a shared link caching network (SLCN) topology can be identified at different levels of the mobile network. Indeed, a radio cell constitutes a SLCN where the eNB acts as the source node connected to the UEs via a shared multicast link. In addition, a SLCN can also be formed by a core network (CN) server (source node) connected to a set of eNBS via a shared wireless backhaul. In both cases, user nodes are equipped with storage resources for content caching. Accordingly, we focus on the analysis and implementation of a SLCN composed of a source node, with access to a library $\mathcal{F}$ of m binary files, connected to n user nodes via a shared multicast link. Each user node is equipped with a cache of storage capacity equivalent to M files, and can make up to L file requests according to a Zipf demand distribution. A multicast link is a shared channel in which any transmission can be overheard by all receivers.

A caching-aided coded multicast scheme is performed over two phases: i) the caching phase, where the source node populates the user caches with appropriate functions of the content library, and ii) the delivery phase, where the source forms a multicast codeword to be transmitted over the shared link in order to meet the users' content demands. These phases are generic for both coded and uncoded schemes, but naively performed in the uncoded case. In fact, when relying on uncoded or naive multicasting during the delivery phase, it is well known that the optimal caching strategy is to cache the top M most popular files at each user

cache. This is however, in general, far from optimal when coding can be used in the delivery phase [30]. In the following, we discuss the potential of caching-aided code design and illustrate its major advantages compared to the optimal caching policy under uncoded (naive) multicasting.

3.1 Network Model and Problem Formulation

We consider a network consisting of a source node with access to a content library $\mathcal{F} = \{1, \cdots, m\}$ of files with size F bits, and n user nodes $\mathcal{U} = \{1, \cdots, n\}$. We assume the source node communicates to the user nodes through a shared multicast link of finite capacity C. Without loss of generality, we can assume $C = F$ bits/unit time and measure the transmission rate of the scheme in units of time necessary to deliver the requested messages to the users. User $u \in \mathcal{U}$ has a storage capacity of size $M_u F$ bits (i.e., M_u files). The channel between the source and all the users follows a shared error-free deterministic model. User u requests L_u files, each of which follows the probability distribution $q_{f,u}$, where $q_{f,u} \in [0, 1]$ and $\sum_{f=1}^{m} q_{f,u} = 1$ (e.g., file f is requested with probability $q_{f,u}$ by one request made by user u.). All the requests (by one user or across users) are assumed to be independently made. We denote $\mathbf{Q} = [q_{f,u}], u = 1, \cdots, n, f = 1, \cdots, m$, as the demand distribution. Let the requested files by user u be $\mathbf{f}_u = \{f_{1,u}, f_{2,u}, \cdots, f_{L_u,u}\}$. One example of the network model is shown in Fig. 3.1. The goal is to design a content distribution scheme (i.e., determine the information stored in the user caches and the multicasted codeword to be sent to all users through the shared link) such that all demands are satisfied with probability 1 and the expected rate $\bar{R}(\mathbf{Q})$ is minimized.[1] The expectation is over the demand distribution $\mathbf{Q}$. We denote the minimum achievable expected rate by $\bar{R}^*(\mathbf{Q})$.

3.2 RAndom Popularity-based caching

Each binary file $f \in \mathcal{F}$ is divided into B_f equal-size packets or chunks. Given the *caching distribution* $\{p_f\}$, with $\sum_{f=1}^{m} p_f = 1$, each user caches chunks of file f with

[1]The expected rate is defined as the average minimum number of file transmissions, which is inversely proportional to the average throughput

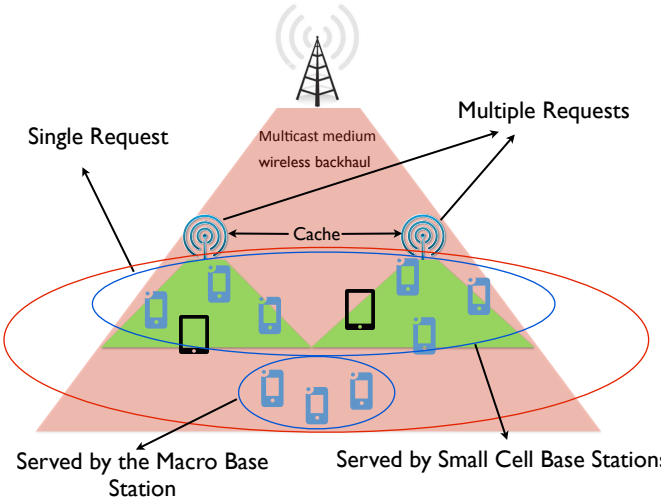


Figure 3.1: An example of the network model, which consists of a source node (base station in this figure) having access to the content library and connected to the users via a shared bottleneck (multicast) link. Each user (small cell base station) may have different cache size, request files according to its own demand distribution and different number of requests.

probability p_f. That is, each user caches a number of chunks $p_f MB_f$ ($p_f \leq 1/M$) of file f chosen uniformly at random. It is important to note that the randomized nature of the selection process allows users to cache different sets of chunks of the same file, shown to be key in creating coded multicast opportunities during the delivery phase. In [30], the authors showed that the optimal caching distribution can be approximated by a truncated uniform distribution $p_f = 1/\tilde{m}, \forall f \leq \tilde{m}$ and $p_f = 0, \forall f > \tilde{m}$, without affecting order-optimality[2], and referred to this caching policy as random least frequently used (RLFU). Compared to the least frequently used (LFU) caching policy (best option under naive multicasting) where the same most popular files are entirely cached at each user, RLFU maximizes the amount of distinct packets collectively cached by the network. The caching placement is shown in Algorithm 1.

3.3 Coded multicasting

A simple example in Fig. 3.2 illustrates the key benefits of coded multicasting during the delivery phase. The network is composed of a source and 3 user nodes re-

[2]Details about the selection of the optimal $\tilde{m}$ are given in [30] (see section IV).

Algorithm 1 Distributed Random Caching Algorithm

1: **for all** $f \in \mathcal{F}$ **do**
2: Each user u caches a subset ($\mathbf{C}_{u,f}$) of $p_{f,u}MB$ distinct packets of file f uniformly at random.
3: **end for**
4: **return** $\mathbf{C} = \{\mathbf{C}_{u,f}, u = 1, \cdots, n, f = 1, \cdots, m\}$.

questing files from a library of $m = 4$ binary files $\mathcal{F} = \{A, B, C, D\}$. Each file (e.g., video segment) is divided into 2 chunks, yielding a library of chunks $\mathcal{C} = \{A_1, A_2, B_1, B_2, C_1, C_2, D_1, D_2\}$. During the caching phase, users 1, 2, and 3 randomly fill their caches with chunks $\{A_1, B_2, D_2\}$, $\{A_2, B_1, D_2\}$, and $\{A_2, B_2, D_1\}$, respectively. During the delivery phase, at a given request round, users 1, 2, and 3 make requests for video segments A, B, and D, respectively. Under an uncoded naive multicasting transmission scheme, the source needs to transmit the missing chunks A_2, B_2, and D_2 over the shared multicast link using 3 time slots. In contrast, by employing coded multicasting, the source can mix the three chunks A_2, B_2 and D_2 via a XOR operation (binary addition) and multicast the coded chunk $A_2 \oplus B_2 \oplus D_2$ using only one time slot. Clearly, in this case, coded multicasting reduces the number of transmissions (and hence the number of delivery time slots) by a factor of three.

As illustrated in the above example, a given user is able to decode its requested chunk from a mixture of combined chunks if and only if it has knowledge of all other combined chunks. Such a problem can be seen as an IC problem [12], and can be described by what is referred to as the *conflict graph* [30]. The conflict graph is constructed such that each graph vertex corresponds to one requested chunk, and an edge between two vertices is created if: i) they correspond to different requested chunks and ii) for each vertex, the associated chunk is not included in the cache of the user requesting the chunk associated with the other vertex. Notice that an edge between two vertices indicates that their associated chunks must be separately transmitted, while non-connected vertices can be modulo summed via XOR operation [12]. The goal is to find the best chunk combinations such that the total number of transmissions is minimized. A common approach, referred to as chromatic index coding (CIC) [30], is to compute a minimum *graph coloring* of the IC conflict graph, where the goal is to find an assignment of colors to the

vertices of the graph such that no two connected vertices have the same color, and the total number of colors is minimized. The multicast codeword is constructed by generating sub-codewords obtained XORing the chunks with the same color, and then concatenating the resulting sub-codewords. The conflict graph of the example given in Fig. 3.2 is illustrated in the top left corner of the figure. The graph consists of 3 vertices corresponding to the three requested packets A_2, B_2, and D_2. There are no edges between the vertices of the graph since, for each vertex, the associated chunk is included in the cache of the users associated with the other vertices. Therefore, all vertices can be assigned the same color and binary added into a single coded transmission, as shown in 3.2.

The work in [30] showed that the combined use of RLFU caching and CIC coded multicasting is order-optimal[3] under any Zipf demand distribution, and that RLFU-CIC provides multiplicative caching gains, that is, the per-user throughput scales linearly or super-linearly with the cache size. In order to prove this result, the authors resort to a polynomial-time approximation of CIC, referred to as greedy constrained coloring (GCC). While GCC exhibits polynomial complexity in the number of users and packets, both CIC and GCC can only guarantee the promising multiplicative caching gain when the number of packets per file grows exponentially with the number of users, significantly limiting their practical performance [27]. Subsequently, the works in [84] and [55] extended the RLFU-CIC and RLFU-GCC schemes to the non-homogeneous SLCN and proposed two improved coded multicasting algorithms: i) the greedy randomized algorithm search procedure (GRASP) based on a greedy randomized approach, and ii) the hierarchical greedy coloring (HGC). These algorithms have been shown to recover a significant part of the multiplicative caching gain, while incurring a complexity at most quadratic in the number of requested packets.

3.4 Decoding phase

From the observation of the received multicast codeword and its cached content, each user has to decode its intended chunks via its own decoding function. In order

[3]Order-optimal in the sense that the number of transmissions needed to satisfy the user demands scales (in number of users, number of files, and memory size) as the optimal scheme.

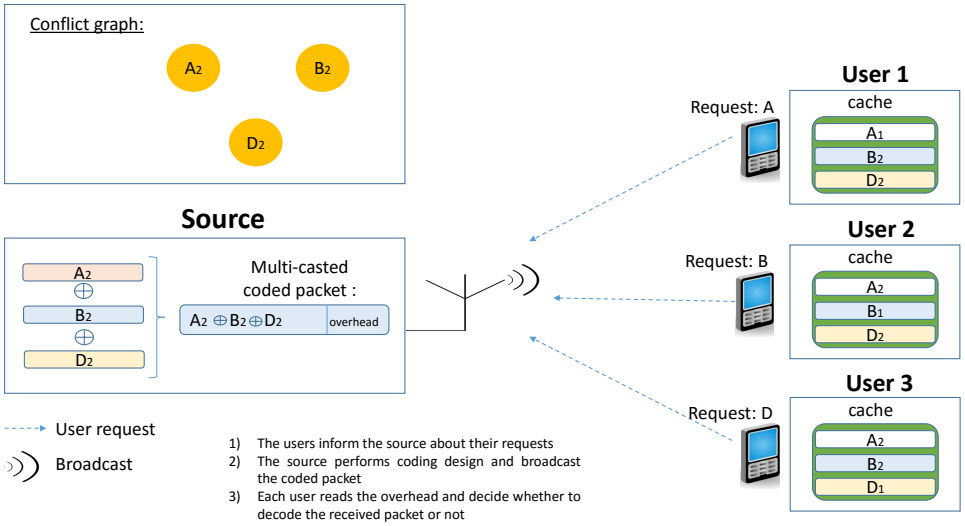


Figure 3.2: Joint design of caching-aided coded multicasting in a 3-user SLN where each user is equipped with a storage capacity of 1.5 file and requests one file from a library with 4 binary files.

to guarantee decoding, the receiver needs to be informed (e.g., via a packet header that carries all necessary information, as shown in Fig. 3.3.a) of the sub-codewords in the concatenated multicast codeword that contain any of its intended chunks. For each of the identified sub-codewords, the receiver obtains its intended chunks by performing the simple binary addition.

In the next section, we describe a fully working prototype implementation that includes the design of the required packet header to ensure full decodability.

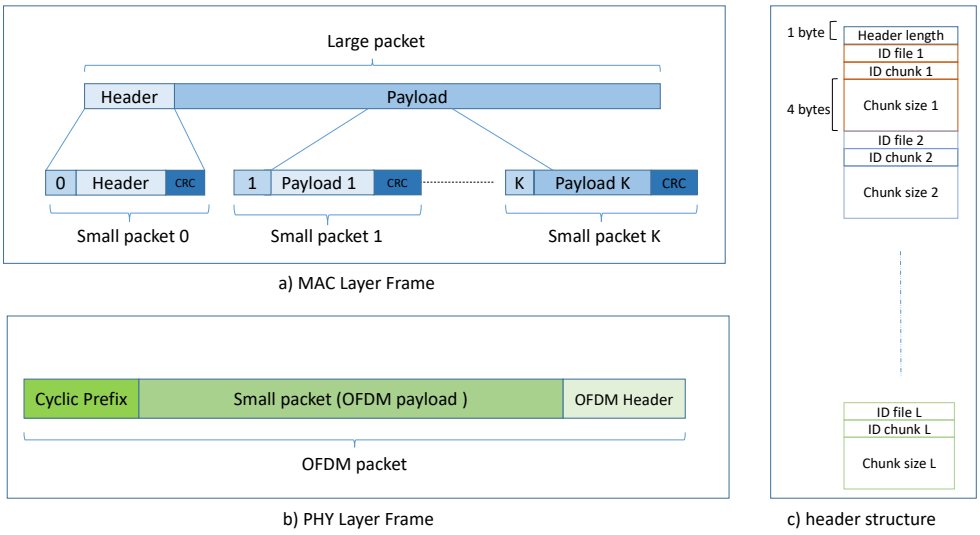


Figure 3.3: An example of the proposed frame structure.

Chapter 4

An Efficient Coded Multicasting Scheme Preserving the Multiplicative Caching Gain

In this Chapter, we address a non-homogenous caching network with a shared multicast link, where users make requests according to possibly different demand distributions and have possibly different cache sizes (see Fig. 4.1 as an example). The considered network is a generalization of the one considered in [51], where all the users are assumed to have the same demand distribution and equal storage capacity. The contributions of this Chapter are as follows. First, we extend RAP-CIC and RAP-GCC to the non-homogenous shared link network and quantify their average performance. Next, we focus on the regime of finite file packetization and numerically show that neither GCC nor the coded delivery scheme proposed in [65], can guarantee the promising performance. Consequently, we introduce a novel algorithm based on a greedy randomized approach referred to as Greedy Randomized Algorithm Search Procedure (GRASP), which is shown to recover a significant part of the multiplicative caching gain, while incurring a complexity at most quadratic in the number of packets.

The Chapter is organized as follows. The achievable caching and coded delivery scheme, along with the general upper bound of the average achievable rate are presented in Section 4.1. Section 4.2 describes the proposed polynomial-time coloring algorithm. Finally, Section 4.3 presents the simulation results and related discussions, and we conclude the Chapter in Section 4.4.

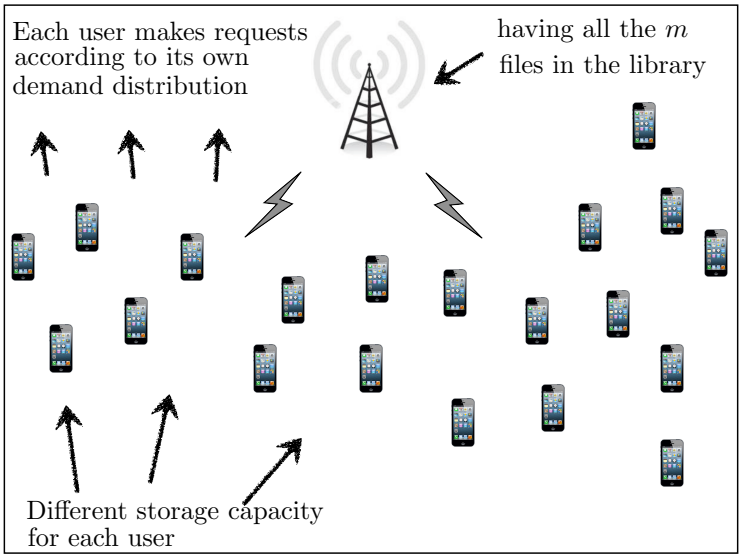


Figure 4.1: An example of the network model, which consists of a source node (base station in this figure) having access to the content library and connected to the users via a shared bottleneck (multicast) link. Each user may have different cache size and request files according to its own demand distribution.

4.1 Achievable Scheme

In this section, we present an achievable scheme based on random popularity-based caching and index coding based delivery.

4.1.1 Coded Multicast Delivery

Our coded delivery scheme is based on chromatic number index coding [11, 51]. The (undirected) conflict graph $\mathcal{H}_{\mathbf{M},\mathbf{W}} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of vertices and edges of $\mathcal{H}_{\mathbf{M},\mathbf{W}}$, respectively, is constructed as follows:

- Consider each packet requested by each user as a distinct vertex, i.e., if the same packet is requested by $N > 1$ users, it results in N distinct vertices.
- Create an edge between vertices $v_1, v_2 \in \mathcal{V}$ if: 1) they do not represent the same packet, and 2) v_1 is not available in the cache of the user requesting v_2, or v_2 is not available in the cache of the user requesting v_1.

Example 1 *We consider a network with* $n = 3$ *users denoted as* $\mathcal{U} = \{1, 2, 3\}$ *and* $m = 3$ *files denoted as* $\mathcal{F} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$. *We assume* $M = 1$ *and sub-packetize each file into three packets. For example,* $\mathrm{A} = \{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3\}$. *Let* $p_{\mathrm{A},u} = \frac{2}{3}$, $p_{\mathrm{B},u} = \frac{1}{3}$ *and* $p_{\mathrm{C},u} = 0$ *for* $u \in \{1, 2, 3\}$, *which means that two packets of* A, *one packet of* B *and none of* C *will be stored in each user's cache. We assume a caching realization* $\mathbf{M}$ *is given by: user* 1 *caches* $\{\mathrm{A}_1, \mathrm{A}_2, \mathrm{B}_1\}$ *(*$\mathbf{M}_{1,\mathrm{A}} = \{\mathrm{A}_1, \mathrm{A}_2\}, \mathbf{M}_{1,\mathrm{B}} = \{\mathrm{B}_1\}, \mathbf{M}_{1,\mathrm{C}} = \emptyset$*); user* 2 *caches* $\{\mathrm{A}_1, \mathrm{A}_3, \mathrm{B}_2\}$*; user* 3 *caches* $\{\mathrm{A}_1, \mathrm{A}_2, \mathrm{B}_3\}$. *We let user* 1 *request* A, *user* 2 *request* B *and user* 3 *request* C *(*$\mathbf{f} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$*) such that* $\mathbf{W} = \{\mathrm{A}_3, \mathrm{B}_1, \mathrm{B}_3, \mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3\}$. *The conflict graph is shown in Fig. 4.2.*

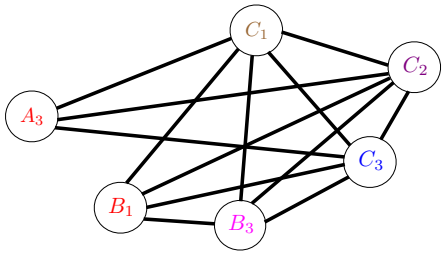


Figure 4.2: An illustration of the conflict graph, where $n = 3, \mathcal{U} = \{1, 2, 3\}$, $m = 3$, $\mathcal{F} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$ and $M = 1$. Each file is partitioned into 3 packets. For example, $\mathrm{A} = \{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3\}$. The caching realization $\mathbf{M}$ is that user 1 caches $\{\mathrm{A}_1, \mathrm{A}_2, \mathrm{B}_1\}$; user 2 caches $\{\mathrm{A}_1, \mathrm{A}_3, \mathrm{B}_2\}$; user 3 caches $\{\mathrm{A}_1, \mathrm{A}_2, \mathrm{B}_3\}$. The requests vector is $\mathbf{f} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$. Hence, $\mathbf{W} = \{\mathrm{A}_3, \mathrm{B}_1, \mathrm{B}_3, \mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3\}$. The color for each vertex in this graph represents a vertex coloring scheme achieved by the algorithm given by Fig. 2. Note that in this case, this vertex coloring is the minimum vertex coloring and the resultant number of packet transmissions is 5. ◊

Next, given a minimum vertex coloring of the conflict graph $\mathcal{H}_{\mathbf{M},\mathbf{W}}$ (see Fig. 4.2), the corresponding index coding scheme transmits the modulo sum of the packets (vertices in $\mathcal{H}_{\mathbf{M},\mathbf{W}}$) with the same color. Therefore, given $\mathbf{M}$ and $\mathbf{W}$, the total number of transmissions in terms of packets is given by the chromatic number $\chi(\mathcal{H}_{\mathbf{M},\mathbf{W}})$. This achieves the transmission rate $\chi(\mathcal{H}_{\mathbf{M},\mathbf{W}})/B$. In the following we refer to this coding scheme scheme as CIC (chromatic index coding).

4.1.2 Achievable Expected Rate

Given n, m, M and the popularity distribution $\mathbf{Q}$, our goal is to find the caching distribution $\mathbf{P}$ that minimizes the expected rate $\bar{R}(\mathbf{P},\mathbf{Q}) \triangleq \mathbb{E}[\chi(\mathsf{H}_{\mathsf{M},\mathsf{W}})/B]$.[1] The upper bound of $\bar{R}(\mathbf{P},\mathbf{Q})$ is given by the following theorem:

Theorem 1 *For any given m, n, M, and $\mathbf{Q}$, when $B \to \infty$, the expected rate $\bar{R}(\mathbf{P},\mathbf{Q})$ achieved by a content distribution scheme that uses caching policy in Fig. 1 with caching distribution $\{\mathbf{P} = [p_{f,u}] : \sum_{f=1}^{m} p_{f,u} = 1, \forall u; p_{f,u} \leq 1/M_u, \forall f, u\}$, and CIC transmission, satisfies*

$$\bar{R}(\mathbf{P},\mathbf{Q}) \leq \bar{R}^{\text{ub}}(\mathbf{P},\mathbf{Q}) \triangleq \min\{\psi(\mathbf{P},\mathbf{Q}), \bar{m}\}, \tag{4.1}$$

with high probability.[2] *In (4.1),*

$$\bar{m} = \sum_{f=\lfloor \min_u\{M_u\}\rfloor+1}^{m} \left(1 - \prod_{u=1}^{n}(1-q_{f,u})\right), \tag{4.2}$$

and

$$\begin{aligned}\psi(\mathbf{P},\mathbf{Q}) \;=\; & \sum_{\ell=1}^{n}\sum_{\mathcal{U}^\ell\in\mathcal{U}}\sum_{f=1}^{m}\sum_{u\in\mathcal{U}^\ell} \\ & \rho_{f,u,\mathcal{U}^\ell}(1-p_{f,u}M_u)^{n-\ell+1}(p_{f,u}M_u)^{\ell-1},\end{aligned} \tag{4.3}$$

where

$$\begin{aligned}&\rho_{f,u,\mathcal{U}^\ell} \triangleq \\ &\mathbb{P}(f = arg\max_{f_u\in\mathbf{f}(\mathcal{U}^\ell)} (p_{f,u}M_u)^{\ell-1}(1-p_{f,u}M_u)^{n-\ell+1}),\end{aligned} \tag{4.4}$$

denotes the probability that f is the file whose $p_{f,u}$ maximizes the term $(p_{f,u}M_u)^{\ell-1}(1-p_{f,u}M_u)^{n-\ell+1})$ among $\mathbf{f}(\mathcal{U}^\ell)$ (the set of files requested by $\mathcal{U}^\ell$). □

[1] $\mathsf{H}_{\mathsf{M},\mathsf{W}}$ denotes the random conflict graph, which is a function of the random caching and demand configurations, M and W, respectively.

[2] The term "with high probability" means that $\lim_{F\to\infty}\mathbb{P}(\bar{R}(\mathbf{P},\mathbf{Q}) \leq \bar{R}^{\text{ub}}(\mathbf{P},\mathbf{Q})) = 1$.

Theorem 1 is proved in Appendix B.

Observe that under homogeneous popularity and cache size, $q_{f,u} = q_f, M_u = M, \forall u \in \mathcal{U}$, then $p_{f,u} = p_f, \forall u \in \mathcal{U}$, and (4.2) and (4.3) become

$$\bar{m} = \sum_{f=\lfloor M \rfloor + 1}^{m} \left(1 - (1 - q_f)^n\right), \tag{4.5}$$

and

$$\psi(\mathbf{P}, \mathbf{Q}) = \sum_{\ell=1}^{n} \binom{n}{\ell} \sum_{f=1}^{m} \rho_{f,\ell} (1 - p_f M)^{n-\ell+1} (p_f M)^{\ell-1}, \tag{4.6}$$

where $\rho_{f,\ell} \triangleq \mathbb{P}(f = \underset{j \in \mathcal{F}^\ell}{\operatorname{argmax}} (p_j M)^{\ell-1}(1 - p_j M)^{n-\ell+1})$ denotes the probability that file f is the file whose p_f maximizes the term $\left((p_j M)^{\ell-1}(1 - p_j M)^{n-\ell+1}\right)$ among $\mathcal{F}^\ell$ (the set of files requested by an arbitrary subset of users of size ℓ). Eq. (4.6) is indeed the upper bound used in [51] to obtain the order-optimal caching distribution for the homogenous network model.

Using the generalized upper bound $\bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q})$ in Theorem 1, we can obtain the desired caching distribution for a wide class of heterogeneous network models. We use $\mathbf{P}^*$ to denote the caching distribution that minimizes $\bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q})$.

4.2 Polynomial-time Algorithms

In this section, we first recapitulate the polynomial-time coded multicasting used in [51] to quantify the order-optimal performance of homogeneous shared link networks by letting the number of packets $B \to \infty$. This scheme is based on a greedy constrained coloring (GCC) approach. In Appendix B, we prove that when $B \to \infty$, GCC achieves the upper bound of the average rate for heterogeneous shared link networks given by (4.1). It is also easy to verify that GCC achieves the same performance as the algorithm given in [65] for the worst-case demand setting.

We then present a novel coded multicasting algorithm based on a greedy random coloring approach that exhibits lower polynomial-time complexity than GCC and refer to it as GRASP (Greedy Randomized Algorithm Search Procedure). In

Section 4.3, we show that for finite file packetization, while GCC loses the multiplicative caching gain, GRASP is still able to approach the limiting performance and recover a significant part of the multiplicative caching gain.

4.2.1 GCC (Greedy Constrained Coloring)

The GCC algorithm works by computing two valid colorings of the conflict graph $\mathcal{H}_{\mathbf{M},\mathbf{W}}$, referred to as GCC_1 and GCC_2. GCC compares the rate achieved by the two coloring solutions and constructs the transmission code based on the coloring with minimum rate.[3]

Algorithm 2 GCC_1

Let $\hat{\mathcal{V}} = \mathcal{V}$;
Let $\mathcal{C} = \emptyset$;
$\mathbf{c}_1 = \emptyset$;
while $\hat{\mathcal{V}} \neq \emptyset$ **do**
  Pick an arbitrary vertex v in $\hat{\mathcal{V}}$. Let $\mathcal{I} = \{v\}$;
  for all all $v' \in \hat{\mathcal{V}}/\{v\} \rightarrow$ **do**
    if (There is no edge between v' and $\mathcal{I}$ $\cap$ $\mathcal{K}_{v'} = \mathcal{K}_{\tilde{v}} : \forall \tilde{v} \in \mathcal{I}$) **then**
      $\mathcal{I} = \mathcal{I} \cup v'$;
    end if
  end for
  Color all the vertices in $\mathcal{I}$ by $c \notin \mathcal{C}$;
  Let $\mathbf{c}_1[\mathcal{I}] = c$;
  $\hat{\mathcal{V}} = \hat{\mathcal{V}} \setminus \mathcal{I}$;
end while
return ($\mathbf{c}_1$);

The greedy constrained coloring algorithm GCC_1 (alg. 2) achieving (eq. 4.3) for large enough F. $\mathcal{K}_v$ denotes the set of users that are either caching or requesting packet v.

GCC_1 computes a coloring of the conflict graph $\mathcal{H}_{\mathbf{M},\mathbf{W}}$ as described in Alg. 2. Note that both the outer while-loop starting at line 4 and the inner for-loop starting at line 6 iterate at most $|\mathcal{V}|$ times, respectively. The operation in line 7 has complexity $O(n)$. Therefore, the complexity of GCC_1 is $O(n|\mathcal{V}|^2)$ or equivalently $O(n^3B^2)$ since $|\mathcal{V}| \leq nB$, which is polynomial in $n, |\mathcal{V}|$ (or n, B).

[3]Recall that the transmission code (index code) is constructed by the modulo sum of all the vertices (packets) in $\mathcal{H}_{\mathbf{M},\mathbf{W}}$ with the same color.

On the other hand, GCC_2 computes the minimum coloring of $\mathcal{H}_{\mathbf{M},\mathbf{W}}$ subject to the constraint that only the vertices representing the same packet are allowed to have the same color. In this case, the total number of colors is equal to the number of distinct requested packets, and the coloring can be found in $O(|\mathcal{V}|^2)$. It is immediate to see that this scheme corresponds to the naive (uncoded) multicasting transmission of all requested packets.

In Appendix B, we prove that GCC achieves the upper bound of the average rate for heterogeneous shared link networks given by (4.1) (when $F \to \infty$). However, as will be shown in Section 4.3, for finite F, GCC loses the promising multiplicative caching gain.

4.2.2 GRASP (Greedy Randomized Algorithm Search Procedure)

In this section, we design an efficient metaheuristic to find suboptimal good solutions of the coded multicasting scheme (coloring problem) given in Section 4.1.1 in reasonable running times (lower than GCC), and that allow preserving the multiplicative caching gain.

The graph coloring problem is a well known NP-complete problem [43]; indeed, given an undirected graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, Garey and Johnson have shown that obtaining colorings using $s \cdot \chi(\mathcal{H})$ colors, where $s < 2$, is NP-hard [42]. So far, Column Generation-based and Branch-and-Price are reputed the most efficient exact methods to solve the problem. An implementation of a Branch-and-Price-and-Cut approach was proposed by Hansen et al. [44], who in 2009 found a family of valid inequalities that do not break the structure of the pricing subproblem. Despite this interesting idea, the practical impact of this approach is limited. Regarding polynomial-time approximation frameworks, one of the most recent results is an algorithm to color any k-colorable graph (for k constant) with $O(n^{\frac{3}{8}} \operatorname{polylog}(n))$ colors [15]. It is based on examining second-order neighborhoods of vertices, rather than just immediate neighborhoods of vertices as in previously proposed approaches. Nevertheless, both exact and approximation algorithms proposed for the coloring problem are able to solve consistently only small instances,

with up to 80 vertices. Thus, the complexity of the problem requires the development of heuristics for large problem instances. Among the hundreds of algorithms that have been proposed in literature, the most recent, effective, and original contributions for the heuristic solution of the problem include a Tabu Search-based (TS) algorithm and a Variable Neighborhood Search (VNS). The TS-based algorithm is called MIPSCLR (MInimal-state Processing Search algorithm for the graph CoLoRing problem) and was proposed in 2000 by Funabiki and Higashino [41]. It combines a Tabu Search technique that considers a number k of available colors, and applies solution recombination technique in an attempt to expand a feasible partial coloring to a complete coloring. The Variable Neighborhood Search (VNS) was proposed by Avanthay et al. [4]. It is a technique based on the exploration of a dynamic neighborhood model. When the algorithm is trapped in a local optimum, the search continues in a new and increasingly distant neighborhood of the current best found solution, which is usually not locally optimal with respect to the new neighborhood. In 2001, Laguna and Martì [62] proposed a GRASP to find suboptimal solution on instances characterized by sparse graph. GRASP [36, 37, 38, 39, 40] is acronym of Greedy Randomized Algorithm Search Procedure and its general framework will be detailed later in this Chapter.

To solve in reasonable running times problem instances characterized by any graph topologies and by using a variable number of colors (and not a constant number k, as done in previous works), we propose a GRASP, whose general framework is described in the following:

1. A GRASP performs a certain number of iterations, until a stopping criterion is met (such as, for example, the total run of `MaxIterations` iterations or the a fixed running time);

2. At each GRASP iteration,

   (a) a greedy-randomized adaptive solution $\mathbf{c}$ is built;

   (b) starting from $\mathbf{c}$ as initial solution, a local search phase is performed returning a locally optimal solution $\mathbf{c}^*$;

3. At the end of all GRASP iterations, the best locally optimal solution $\mathbf{c}_{\text{best}}$ (i.e., the solution corresponding to the best function objective value $f(\mathbf{c}_{\text{best}})$ is returned as final solution and the algorithm stops.

Our GRASP differs from the GRASP proposed by Laguna and Martì [62] in two main aspects: 1) it is able to tackle problem instances characterized by any graph topology, density/sparsity, and any size; 2) the local search strategy checks for redundant color focusing on each vertex, one at the time; while Laguna and Martì's GRASP iteratively joins a pair of independent sets and focuses only on *illegal vertices* (i.e., those vertices that after the union result colored with the same color as one of their adjacent vertex).

Algorithm 3 GRASP_GraphColoring(`MaxIterations`, $\mathcal{V}$, $\mathcal{E}$, d, $Adj(\cdot)$, $f(\cdot)$)

$\mathbf{c}_{\text{best}} := \emptyset$; $f(\mathbf{c}_{\text{best}}) := +\infty$;
$\hat{\mathcal{V}} :=$ `sort`$(\mathcal{V})$;
for $k = 1$ **to** `MaxIterations`$\rightarrow$ **do**
  $\mathcal{C} := \emptyset$;
  $\beta :=$ `random`$[0, 1]$;
  $\mathbf{c} :=$ `BuildGreedyRandAdaptive`$(\beta, \hat{\mathcal{V}}, \mathcal{E}, d, Adj(\cdot), f(\cdot), \mathcal{C})$
  $\mathbf{c}^* :=$ `LocalSearch`$(\hat{\mathcal{V}}, \mathcal{E}, \mathbf{c}, f(\mathbf{c}), \mathcal{C})$;
  if $(f(\mathbf{c}^*) < f(\mathbf{c}_{\text{best}}))$ **then**
    $\mathbf{c}_{\text{best}} := \mathbf{c}^*$;
    $f(\mathbf{c}_{\text{best}}) := f(\mathbf{c}^*)$;
  end if
end for
return $(\mathbf{c}_{\text{best}})$;

Alg. 3 depicts the pseudo-code of our GRASP heuristic for the Graph Coloring Problem. In $\mathcal{H}_{\mathbf{M},\mathbf{W}}$, for each vertex $i \in \mathcal{V}$, $Adj(i) = \{j \in \mathcal{V} \mid [i, j] \in \mathcal{E}\}$. In the following subsections, we will describe the construction solution method and the local search method performed by our algorithm.

4.2.2.1 Building A Solution

Let $\mathcal{Q}$ be a set of $|\mathcal{V}|$ candidate colors. Then, the construction phase assigns to each vertex $i \in \mathcal{V}$ a color $c \in \mathcal{Q}$ in such a way that c is not assigned to any vertex adjacent to vertex i and that the total number of colors utilized is attempted to be as much smaller as possible.

Algorithm 4 BuildGreedyRandAdaptive(β, $\hat{\mathcal{V}}$, $\mathcal{E}$, d, $Adj(\cdot)$, $f(\cdot)$, $\mathcal{C}$)

$\mathbf{c} := \emptyset$;
for $j = 1$ **to** $|\mathcal{V}|\rightarrow$ **do**
  $RCL :=$ `MakeRCL` (β, $\mathcal{V}$, $\mathcal{E}$, d, $\mathbf{c}$);
  $i :=$ `SelectIndex` (RCL);
  $c :=$ `GetColor` ($\mathcal{V}$, $\mathcal{E}$, i, $\mathcal{C}$, $Adj(\cdot)$, $\mathbf{c}$);
  $\mathbf{c}[i] := c$;
  if ($c \notin \mathcal{C}$) **then**
    $\mathcal{C} := \mathcal{C} \cup \{c\}$;
    $f(\mathbf{c}) := |\mathcal{C}|$;
  end if
end for
return ($\mathbf{c}$);

Algorithm 5 MakeRCL(β, $\mathcal{V}$, $\mathcal{E}$, d, $\mathbf{c}$)

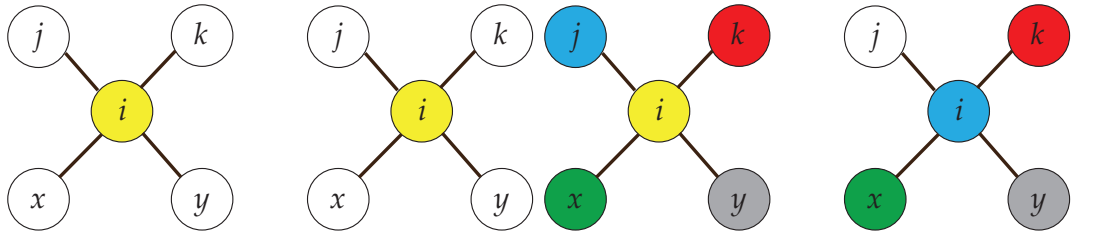
$g_{\min} := \min_{i \in \mathcal{V} \setminus \mathbf{c}} d(i)$;
$g_{\max} := \max_{i \in \mathcal{V} \setminus \mathbf{c}} d(i)$;
$\tau := g_{\min} + [\beta \cdot (g_{\max} - g_{\min})]$;
$RCL := \{i \in \mathcal{V} \setminus \mathbf{c} \mid d(i) \geq \tau\}$;
return (RCL);

Alg. 4 shows the pseudo-code of the construction procedure and the pseudo-code of the functions in Alg. 4 are shown in Alg. 5 and 6. To build a feasible solution, as shown in Alg. 4, starting from an empty solution (line 1) the GRASP construction phase performs in for-loop in lines 2–11 $|\mathcal{V}|$ iterations assigning at each iteration a color to a not yet colored vertex and proceeding in a greedy, randomized, and adaptive manner. In more detail, at each iteration, the choice of the next vertex to be colored is determined by ordering all candidate vertices (i.e., those that are still uncolored) in a candidate list $\mathcal{W} = \mathcal{V} \setminus \mathbf{c}$ (see Alg. 5) with respect to a greedy function $g : \mathcal{W} \mapsto \mathbb{R}$ that measures the myopic benefit of selecting each vertex and that in our case is related to the degree of a candidate vertex. The construction is adaptive because the benefits associated with every candidate vertex are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous vertex. The probabilistic component is characterized by randomly choosing one of the best candidates in the list $\mathcal{W}$, but not necessarily the top candidate. The list of best candidates is called

Algorithm 6 GetColor($\mathcal{V}$, $\mathcal{E}$, i, $\mathcal{C}$, $Adj(\cdot)$, $\mathbf{c}$)

$\mathcal{L} := \emptyset$;
for all each $j \in Adj(i)$ $\mathcal{L} := \mathcal{L} \cup \{\mathbf{c}[i]\}$ **do**
  if $(\mathcal{C} \setminus \mathcal{L} \neq \emptyset)$ **then**
    $c' :=$ `SelectColor` $(\mathcal{C} \setminus \mathcal{L})$;
  else
    $c' :=$ `NewColor` $(\mathcal{C})$;
  end if
end for
return (c');

the *Restricted Candidates List* (RCL). We will see later how to build the RCL (see Alg. 5) selecting the best candidates to be inserted into the list (line 3). Once randomly selected one of the best candidate from the RCL (line 4), that vertex will be assigned a color according to the colors assigned to its adjacent vertices (line 5).



(a) Scenario 6.i.: $Adj(i) = \{j, k, x, y\}$, $\mathcal{C} = \emptyset$. Then, c =yellow and $\mathcal{C}$ = {yellow}.

(b) Scenario 6.ii.: $Adj(i) = \{j, k, x, y\}$, $\mathcal{C}$ = {yellow,blue}. Then, c =yellow.

(c) Scenario 6.iii.: $Adj(i) = \{j, k, x, y\}$, $\mathcal{C}$ = {blue,green,red,gray}. Then, c =yellow and $\mathcal{C} = \mathcal{C} \cup$ {yellow}.

(d) Scenario 6.iv.: $Adj(i) = \{j, k, x, y\}$, $\mathcal{C}$ = {blue,green,red,gray}. Then, c =blue.

Figure 4.3: The four possible scenarios that may occur once selected a vertex i to be colored during the construction phase.

The construction phase performs the following steps (shown by Alg. 4, 5 and 6).

Let $d(i) = |Adj(i)|$, for all $i \in \mathcal{V}$, be the degree of vertex i. Let $\mathbf{c} = \emptyset$ be the solution under construction (initially empty), i.e., the set of vertices already

assigned to a color, and let $\mathcal{C} = \emptyset$ (initially empty) be the set of colors that are associated with at least a vertex in $\mathbf{c}$. At each iteration, the following quantities are computed and operations performed:

1. $g_{\min}$, the minimum greedy value:

$$g_{\min} = \min_{i \in \mathcal{V} \setminus \mathbf{c}} d(i);$$

2. $g_{\max}$, the maximum greedy value:

$$g_{\max} = \max_{i \in \mathcal{V} \setminus \mathbf{c}} d(i);$$

3. a threshold value τ:

$$\tau = g_{\min} + [\beta \cdot (g_{\max} - g_{\min})], \text{ where } \beta \in [0, 1];$$

4. the RCL as the subset of candidate uncolored vertices whose degree is at least τ:

$$RCL = \{i \in \mathcal{V} \setminus \mathbf{c} \mid d(i) \geq \tau\};$$

5. a vertex i is randomly selected from the RCL ($i =$ `SelectIndex`(RCL) in Alg. 4).

   Note that, from the value of $\beta \in [0, 1]$ depends the percentage of greediness versus randomness in the choice of the vertices to be inserted in the RCL at each iteration. In fact, for $\beta = 1$, the choice is totally greedy and only vertices with degree $g_{\max}$ are inserted. On the contrary, for $\beta = 0$, the choice is totally random and all candidate vertices are inserted (i.e., $RCL = \mathcal{W}$);

6. once selected vertex i, its adjacent vertices are analyzed and the four possible scenarios that may occur are the following:

   i. all adjacent vertices are still uncolored and the set $\mathcal{C} = \emptyset$: in this case, a new color c is assigned to vertex i and $\mathcal{C} = \mathcal{C} \cup \{c\}$ (Fig. 4.3(a));

ii. all adjacent vertices are still uncolored and the set $\mathcal{C} \neq \emptyset$: in this case, vertex i is colored with the first color $c \in \mathcal{C}$ available (Fig. 4.3(b));

iii. at least one adjacent vertex is colored with a color $c \in \mathcal{C}$ and all currently used colors $c \in \mathcal{C}$ are already assigned to at least an adjacent vertex: in this case, vertex i is colored with a new color c' and $\mathcal{C} = \mathcal{C} \cup \{c'\}$ (Fig. 4.3(c));

iv. at least one adjacent vertex is colored with a color $c \in \mathcal{C}$ and there is a color $c' \in \mathcal{C}$ that has not been assigned to any adjacent vertex: in this case, vertex i is colored with color c' (Fig. 4.3(d)).

7. vertex i is inserted into the solution under construction ($\mathbf{c}[i] = c'$ or $\mathbf{c}[i] = c$, according to scenarios 6.i.–6.iv.) and the objective function value coherently updated (i.e., $f(\mathbf{c}) = |\mathcal{C}|$).

4.2.2.2 Local Search

Solutions generated by the GRASP construction are not guaranteed to be locally optimal with respect to simple neighborhood definitions. Given a solution $\mathbf{c}$, the neighborhood structure $\mathcal{N}(\mathbf{c})$ relates $\mathbf{c}$ to a subset of solutions $\mathcal{N}(\mathbf{c})$ "close" to $\mathbf{c}$ and $\mathbf{c}$ is said to be locally optimal if there is no better solution in $\mathcal{N}(\mathbf{c})$. It is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the current neighborhood. In our GRASP, the local search algorithm, whose pseudo-code is reported in Alg. 7, has the purpose of checking redundancy of each color $c \in \mathcal{C}$, in order to eventually decrease the current objective function value $|\mathcal{C}|$.

In more detail, in loop for in lines 1–19 in Alg. 7, the local search iteratively for each color $c \in \mathcal{C}$ computes the set $\mathcal{G}_c$ of all vertices colored with color c (line 2) and performs the following steps:

1. for each vertex $i \in \mathcal{G}_c$, it spans $Adj(i)$, i.e., the vertices adjacent to vertex i: if there is a color $c' \in \mathcal{C}$, $c' \neq c$, not assigned to any adjacent vertex $j \in Adj(i)$, then vertex i becomes colored with color c';

Algorithm 7 LocalSearch($\hat{\mathcal{V}}$, $\mathcal{E}$, $\mathbf{c}$, $f(\mathbf{c})$, $\mathcal{C}$)

```
for all each c ∈ C→ do
  G_c := {i ∈ V | H[i] = c};
  B:=∅;
  ĉ := c;
  for all each i ∈ G_c→ do
    A := ∅;
    for all each j ∈ Adj(i) A := A ∪ {c[j]} do
      if (C \ A ≠ ∅) then
        c' := SelectColor (C \ A);
        ĉ[i] := c';
        B := B ∪ {i};
      end if
    end for
    if (|B| = |G_c|) then
      c := ĉ;
      C := C \ {c};
      f(c) := |C|;
    end if
  end for
end for
return (c);
```

2. color c is removed from the set $\mathcal{C}$ if and only if in Step 1 it has been possible to replace c associated with each vertex $i \in \mathcal{G}_c$ with some color c', $c' \neq c$.

4.2.2.3 Computational complexity of the proposed GRASP

Given the input undirected graph $\mathcal{H}_{\mathbf{M},\mathbf{W}} = (\mathcal{V}, \mathcal{E})$, the complexity analysis of the designed GRASP algorithm to solve the Graph Coloring Problem of is as follows:

1. Sorting the set $\mathcal{V}$ according to a non-ascending order of the degree of the vertices has a computational time equal to $O(|\mathcal{V}| \log |\mathcal{V}|)$;

2. The construction phase runs $|\mathcal{V}|$ iterations and at each iteration it performs the following steps:

   - construction of the RCL: since $g_{\min}$, $g_{\max}$, and τ can be computed in $O(1)$ (the vertices are sorted according to their degree), to identify the

candidate vertices to be inserted in the RCL (according to the rule described in Section 4.2.2.1) requires $O(|\mathcal{V}|)$;

- color assignment: to this end, it must be scanned the adjacent vertices of vertex i randomly extracted from the RCL in order to choose the color that will be assigned to it in an appropriate way. This task requires $O(|Adj(i)|)$. Therefore, summing all over vertices $i \in \mathcal{V}$ to be colored, we obtain $\sum_{i=1}^{|\mathcal{V}|} |Adj(i)| = 2 \cdot |\mathcal{E}|$. It is clear that the computational complexity of the construction procedure is equal to $O(|\mathcal{E}|)$.

3. For each used color $c \in \mathcal{C}$ and for each $i \in \mathcal{G}_c$, the local search procedure analyzes the adjacent vertices of vertex i in order to attempt to remove color c from the set $\mathcal{C}$. Since $|\mathcal{C}| + |\mathcal{G}_c| = O(|\mathcal{V}|)$, this procedure has a computational complexity equal to $O(|\mathcal{E}|)$.

Step 1) is performed only once, at the beginning of the algorithm. Since step 2) and step 3) are performed a fixed number of iterations (`MaxIterations`), it results that the overall computational complexity of the designed algorithm to solve the Graph Coloring problem has a computational complexity equal to:

$$\begin{aligned} &O(|\mathcal{V}| \log |\mathcal{V}| + \texttt{MaxIterations} \cdot \max\{|\mathcal{V}|, |\mathcal{E}|\}) \\ &= O(|\mathcal{V}| \log |\mathcal{V}| + \texttt{MaxIterations} \cdot |\mathcal{E}|). \end{aligned} \tag{4.7}$$

Remark: we can see that complexity of GRASP is $O(|\mathcal{V}|^2)$, which is a factor of n lower than the complexity ($O(n|\mathcal{V}|^2)$) of GCC achieving $\bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q})$ for large enough B.

4.3 Simulations and Discussion

In this section, we numerically analyze the performance of the two achievable schemes illustrated in Section 4.2 for finite file packetization. Specifically, assuming the distributed random popularity-based caching policy in Alg. 1, we compare the average performance of GCC and GRASP when files are partitioned into a finite number (B) of packets. For comparison, we also plot 1) the performance of LFU

(Least Frequently Used),[4] shown to be optimal in uncoded networksand 2) the performance of GCC for infinite packetization ($B \to \infty$), as given in Theorem 1.

For simplicity and to illustrate the effectiveness of GRASP, we consider a homogeneous network scenario, in which users request files according to a Zipf popularity distribution with parameter $\gamma \in \{0.2, 0.6\}$ and all caches have size M files. For all considered schemes, the caching distribution $\mathbf{P}^*$ is obtained by minimizing $\bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q})$ in (4.1) among all $\mathbf{P}$ described by a m-dimensional vector taking value in $\{\frac{1}{\tilde{m}}, 0\}$.[5],

Further, we assume that when using GCC or GRASP, the source node pre-evaluates the performance of LFU and chooses the minimum accordingly. Hence, denoting by R_{LFU}, R_{GCC} and R_{GRASP} the average rate achieved by LFU, GCC and GRASP, respectively, Fig. 4.4 plots the performance of GCC and GRASP as $\min\{R_{LFU}, R_{GCC}\}$, and $\min\{R_{LFU}, R_{GRASP}\}$, respectively.

Fig. 4.4(a), (b) and (c) plot the average rate for a network with $n = 10$ users, $m = 250$ files and Zipf parameter $\gamma = 0.2$. Observe how the significant caching gains (with respect to LFU) quantified by the order-optimal upper bound are completely lost when using GCC with finite packetization $B = 20$, and only slightly recovered as the packetization increases to $B = 100$ and $B = 200$. On the other hand, observe how GRASP remarkably preserves most of the promising multiplicative caching gains for the same values of file packetization. For example, in Fig. **??**, if M doubles from $M = 50$ to $M = 100$, then the rate achieved by GRASP essentially halves from 4.2 to 2.2. For the same regime, it is straightforward to verify that neither GCC nor LFU exhibits this property.[6] Fig. 4.4(d) illustrates a scenario with higher popularity skewness, e.g., $\gamma = 0.6$. Observe how, also in this scenario, a finite number of packets ($B = 50$) completely limits the gains of GCC.

[4]LFU discards the least frequently requested file upon the arrival of a new file to a full cache of size M_u files. In the long run, this is equivalent to caching the M_u most popular files.

[5]This constraint on the caching distribution introduced in [51], originates a scheme referred to as Random Least-Frequently-Used (Random LFU), which approximates RAP and generalizes the well known LFU caching scheme. In Random LFU, each user just caches packets from the (carefully designed) $\tilde{m}$ most popular files in a distributed and random manner.

[6]While LFU can only provide an additive caching gain, additive and multiplicative gains may show indistinguishable when M is comparable to the library size m. Hence, one needs to pick a reasonably small M ($\frac{m}{n} < M \ll m$) to observe the multiplicative caching gain of GRASP.

On the other hand, GRASP is still able to preserve significant gains. For example, when M doubles from $M = 70$ to $M = 140$, the achievable rate by GRASP goes from 8.8 to 5.5, approaching a half rate reduction even with only 50 packets per file. Finally note from Fig. 4.4(b), that in oder to guarantee a rate $R = 4$, GCC requires a cache size of $M = 120$, while GRASP can reduce the cache size requirement to $M = 50$, a 2.4× cache size reduction.

4.4 Conclusions

In this Chapter, we show that the promising multiplicative caching gain analytically quantified for the shared link caching network can be completely lost in finite regimes of the system parameters. We first extend the analysis of this caching network to the case of heterogeneous cache sizes and popularity distributions, providing an upper bound on the limiting average performance when the number of packets per file goes to infinity. We then focus on finite regimes of all system parameters and show that the greedy constrained coloring (GCC) scheme used to quantify this upper bound quickly loses the multiplicative caching gain for finite file packetization. We then design a novel polynomial-time coded multicasting scheme based on a greedy randomized algorithm search procedure (GRASP), which is able to recover a significant part of the multiplicative caching gain with the same finite file packetization. Our results, while initially negative, shed light on the possibilities to preserve the multiplicative caching gain via careful design of coded multicasting schemes for finite values of the system parameters.

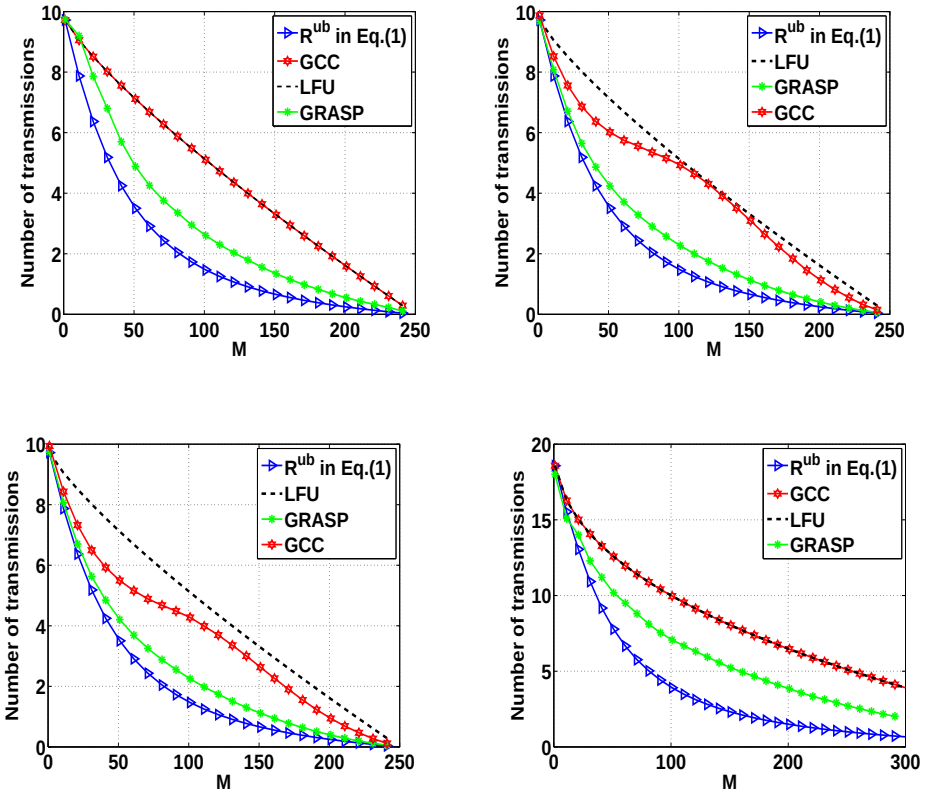


Figure 4.4: Average number of transmission for a shared multicast link. a) $n = 10$,$m = 250$, $B = 20$, and $\gamma = 0.2$; b) $n = 10$,$m = 250$, $B = 100$, and $\gamma = 0.2$; c) $n = 10$,$m = 250$, $B = 200$, and $\gamma = 0.2$; d) $n = 20$,$m = 500$, $B = 50$, and $\gamma = 0.6$.

Chapter 5

An Efficient Multiple-Groupcast Coded Multicasting Scheme for Finite Fractional Caching

In this Chapter, we address a non-homogenous caching network with a shared multicast link, where users make requests according to possibly different demand distributions and have possibly different cache sizes (see Fig. 4.1 as an example). The considered network is a generalization of the one considered in [51], where all the users are assumed to have the same demand distribution, equal storage capacity and one request per user. The contributions of this Chapter are as follows. First, we extend RAP-CIC and RAP-GCC to the non-homogenous shared link network, where users may have different storage capacities, different demand distributions, and different number of requests, and quantify their average performance for infinite packetization by introduce the delivery scheme referred to as greedy constraint local coloring (GCLC). It is worth to mention that the extension of the multiple requests is of importance. As shown in Fig. 4.1, the users can be model as small cell base stations, each of which serves a set of users. In this case, each small cell base station can be modeled as a user with multiple requests. Next, we focus on the regime of finite file packetization and numerically show that neither GCLC nor the coded delivery scheme proposed in [52], can guarantee the promising performance. Consequently, we introduce a novel algorithm referred to as hierarchical greedy local coloring (HgLC), which is shown to recover a significant part of the multiplicative caching gain, while incurring a complexity at most

quadratic in the number of packets.

The Chapter is organized as follows. The achievable caching and coded delivery scheme, along with the general upper bound of the average achievable rate are presented in Section 5.1. Section 5.2 describes the proposed polynomial-time coloring algorithm. Finally, Section 5.3 presents the simulation results and related discussions.

5.1 Achievable Scheme

In this section, we present an achievable scheme based on random popularity-based caching and index coding based delivery.

5.1.1 Coded Multicast Delivery

Our coded delivery scheme is based on local chromatic number index coding [76, 52]. The directed conflict graph $\mathcal{H}_{\mathbf{C},\mathbf{W}} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of vertices and edges of $\mathcal{H}_{\mathbf{C},\mathbf{W}}$, respectively, is constructed as follows:

- Consider each packet requested by each user as a distinct vertex, i.e., if the same packet is requested by $N > 1$ users, it results in N distinct vertices.
- For any pair of vertices v_1, v_2, we say that vertex (packet) v_1 interferes with vertex v_2 if v_1 is not in the cache of the user(s) who requests v_2, and v_1 and v_2 do not represent the same packet. Then, draw a directed edge from vertex v_2 to vertex v_1 if v_1 interferes with o_2.

Based on this caching scheme, we design a delivery scheme based on linear index coding. In particular, we focus on encoding functions of the following form: for the request vectors $\mathbf{f}_u, u \in \mathcal{U}$, the multicast codeword is given by

$$X_{\{\mathbf{f}_u, u\in\mathcal{U}\}} = \sum_{v\in\mathcal{V}} \omega_v \mathbf{g}_v = \mathbf{G}\boldsymbol{\omega}, \tag{5.1}$$

where ω_v is the binary vector corresponding to packet v, represented as a (scalar) symbol of the extension field $\mathbb{F}_{2^{F/B}}$, $\mathbf{g}_v \in \mathbb{F}_{2^{F/B}}^{\nu}$ is the coding vector of packet v and where we let $\mathbf{G} = [\mathbf{g}_1, \cdots \mathbf{g}_{|\mathcal{V}|}]$ and $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_{|\mathcal{V}|}]^{\mathsf{T}}$. The number of rows

ν of $\mathbf{G}$ yields the number of sub-packet transmissions. Hence, the coding rate is given by ν/B file units.

To find the desired ν, we introduce the definition of the local chromatic number:

Definition 1 *(***Local Chromatic Number***) The directed local chromatic number of a directed graph $\mathcal{H}^d$ is defined as:*

$$\chi_{\text{lc}}(\mathcal{H}^d) = \min_{\mathbf{c}\in\mathcal{C}} \max_{v\in\mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))| \tag{5.2}$$

where $\mathcal{C}$ denotes the set of all vertex-colorings of $\mathcal{H}$, the undirected version of $\mathcal{H}^d$, $\mathcal{V}$ denotes the vertices of $\mathcal{H}^d$, $\mathcal{N}^+(v)$ is the closed out-neighborhood of vertex v,[1] *and $\mathbf{c}(\mathcal{N}^+(v))$ is the total number of colors in $\mathcal{N}^+(v)$ for the given coloring c.* $\Diamond$

It can be shown that based on each local coloring number $\max_{v\in\mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|$, there exists an $\mathbf{G}$ such that a valid index code can be found. The design of $\mathbf{G}$ is given by [76, 52]. [2] An example of the delivery scheme is described in Example 2 in the following for illustration.

Hence, given $\mathbf{C}$ and $\mathbf{W}$, and a coloring scheme c of $\mathcal{H}_{\mathbf{C},\mathbf{W}}$, there exists an index code described by $\mathbf{G}$, and the total number of transmissions in terms of packets is given by the local coloring number $\max_{v\in\mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|$. This achieves the transmission rate $\max_{v\in\mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|/B$. Ideally, the minimum transmission rate given by local chromatic number is $\chi_{\text{lc}}(\mathcal{H}_{\mathbf{C},\mathbf{W}})/B$. In the following we refer to this coding scheme scheme as LCIC (local chromatic index coding). Due to the difficulty of characterizing the exact value of $\chi_{\text{lc}}(\mathcal{H}_{\mathbf{C},\mathbf{W}})$, instead, in the following, we will capture an upper bound of $\chi_{\text{lc}}(\mathcal{H}_{\mathbf{C},\mathbf{W}})$. In another word, we will propose an efficient coloring scheme called Hierarchical greedy Local Coloring (HgLC) such that the achievable local coloring number or transmission rate achieves a significant gain compared with the conventional caching scheme in literature.

Example 2 *We consider a network with $n = 3$ users denoted as $\mathcal{U} = \{1, 2, 3\}$ and $m = 3$ files denoted as $\mathcal{F} = \{\text{A}, \text{B}, \text{C}\}$. We assume $M = 1$ and sub-packetize each file into three packets. For example, $\text{A} = \{\text{A}_1, \text{A}_2, \text{A}_3\}$. Let $p_{\text{A},u} = \frac{1}{3}$, $p_{\text{B},u} = \frac{1}{3}$ and*

[1] Closed out-neighborhood of vertex o includes vertex v and all the connected vertices via out-going edges of v.

[2] Instead of using local chromatic number it is also straightforward to use fractional local chromatic number to design the coding vector $\mathcal{G}$ as illustrated in [76, 52].

$p_{\mathrm{C},u} = \frac{1}{3}$ *for* $u \in \{1,2,3\}$*, which means that one packet from each of* $\mathrm{A}, \mathrm{B}, \mathrm{C}$ *will be stored in each user's cache. We assume a caching realization* $\mathbf{C}$ *is given by: user* u *caches* $\{\mathrm{A}_u, \mathrm{B}_u, \mathrm{C}_u\}$ *(*$\mathbf{C}_{u,\mathrm{A}} = \{\mathrm{A}_u\}, \mathbf{C}_{u,\mathrm{B}} = \{\mathrm{B}_u\}, \mathbf{C}_{u,\mathrm{C}} = \mathrm{C}_u$*). We let each user make one request. Specifically, we let user* 1 *request* A*, user* 2 *request* B *and user* 3 *request* C *(*$\mathbf{f}_1 = \{\mathrm{A}\}, \mathbf{f}_2 = \{\mathrm{B}\}, \mathbf{f}_3 = \{\mathrm{C}\}$*) such that* $\mathbf{W}_{1,A} = \{\mathrm{A}_2, \mathrm{A}_3\}, \mathbf{W}_{2,A} = \{\mathrm{A}_1, \mathrm{A}_3\}, \mathbf{W}_{3,B} = \{\mathrm{B}_1, \mathrm{B}_2\}$*. The conflict graph and the corresponding coloring are shown in Fig. 5.1. We can see that the total number of colors needed, the chromatic number in this case, is* 5*, while the local coloring number, or the local chromatic number in this case, is* 4*. We construct* $\mathbf{V}$ *by using the parity-check matrix of a* $(5,4)$ *MDS code, which is given by:*

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{5.3}$$

Then, we allocate the same vector to the vertex (packet) with the same color as shown in Fig. Fig. 5.1. Hence, the transmitted codeword is given by $A_1 \oplus A_2$*,* $A_1 \oplus A_3$*,* $A_1 \oplus B_1$*,* $A_1 \oplus B_2$*, of length* $4/3$ *file units.* $\Diamond$

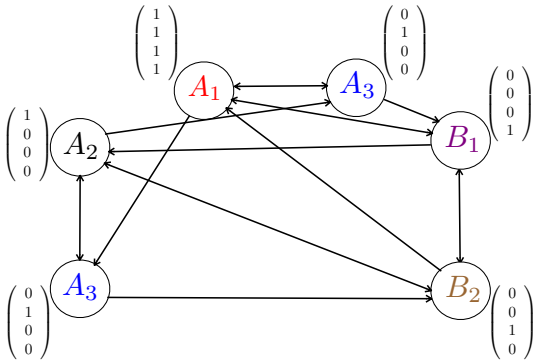


Figure 5.1: An illustration of the directed conflict graph and the corresponding index code. The coloring of the graph is given by the colors of the fonts. The total number of colors is 5, and the local coloring number is 4.

5.1.2 Achievable Expected Rate

Given n, m, M and the demand distribution $\mathbf{Q}$, our goal is to find the caching distribution $\mathbf{P}$ that minimizes the expected rate $\bar{R}(\mathbf{P}, \mathbf{Q}) \triangleq \mathbb{E}[\chi_{\text{lc}}(\mathsf{H}_{\mathsf{M},\mathsf{W}})/B]$.[3] Let $L = \max_u L_u$ and we order $L_u, u \in \mathcal{U}$ as a decreasing sequence denoted as $L_{[1]} \geq L_{[2]} \geq L_{[3]}, \cdots, L_{[n]}$, where $L_{[i]}$ is the ith largest L_u and $[i] = u$. It can be seen that $L_{[1]} = \max_u L_u$ and $L_{[n]} = \min_u L_u$. Let $n_j = \sum_{[i]} 1\{L_{[i]} - j \geq 0\}$, where $1\{\cdot\}$ is the indicator function. Let $\mathcal{U}_{n_j} = \{[i] \in \mathcal{U} : 1\{L_{[i]} - j \geq 0\}\}$. The upper bound of $\bar{R}(\mathbf{P}, \mathbf{Q})$ is given by the following theorem:

Theorem 2 *For any given m, n, M, and $\mathbf{Q}$, when $B \to \infty$, the expected rate $\bar{R}(\mathbf{P}, \mathbf{Q})$ achieved by a content distribution scheme that uses caching policy in Fig. 1 with caching distribution $\{\mathbf{P} = [p_{f,u}] : \sum_{f=1}^{m} p_{f,u} = 1, \forall u; p_{f,u} \leq 1/M_u, \forall f, u\}$, and CIC transmission, satisfies*

$$\bar{R}(\mathbf{P}, \mathbf{Q}) \leq \bar{R}^{\text{GCLC}}(\mathbf{P}, \mathbf{Q}) \triangleq \min\{\psi(\mathbf{P}, \mathbf{Q}), \bar{m} - \bar{M}\}, \tag{5.4}$$

with high probability.[4] *In (5.4),*

$$\bar{m} = \sum_{f=1}^{m} \left(1 - \prod_{u=1}^{n} (1 - q_{f,u})^{L_u}\right), \tag{5.5}$$

and

$$\bar{M} = \sum_{f=1}^{m} \min_u p_{u,f} \left(1 - \prod_{u=1}^{n} (1 - q_{f,u})^{L_u}\right), \tag{5.6}$$

and

$$\psi(\mathbf{P}, \mathbf{Q}) \;=\; \sum_{j=1}^{L} \sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \in \mathcal{U}_{n_j}} \sum_{f=1}^{m} \sum_{u \in \mathcal{U}^\ell} \rho_{f,u,\mathcal{U}^\ell} (1 - p_{f,u} M_u)^{n_j - \ell + 1} (p_{f,u} M_u)^{\ell - 1}, \tag{5.7}$$

where

$$\rho_{f,u,\mathcal{U}^\ell} \triangleq \mathbb{P}(f = arg \max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} \; (p_{f,u} M_u)^{\ell-1} (1 - p_{f,u} M_u)^{n_j - \ell + 1}), \tag{5.8}$$

[3] $\mathsf{H}_{\mathsf{M},\mathsf{W}}$ denotes the random conflict graph, which is a function of the random caching and demand configurations, M and W, respectively.

[4] The term "with high probability" means that $\lim_{F \to \infty} \mathbb{P}(\bar{R}(\mathbf{P}, \mathbf{Q}) \leq \bar{R}^{\text{GCLC}}(\mathbf{P}, \mathbf{Q})) = 1$.

denotes the probability that f is the file whose $p_{f,u}$ maximizes the term $(p_{f,u}M_u)^{\ell-1}(1-p_{f,u}M_u)^{n-\ell+1})$ among $\mathbf{f}(\mathcal{U}^\ell)$ (the set of files requested by $\mathcal{U}^\ell$). □

Under homogeneous demand distribution, cache size and number of request per user, we have the following corollary

Corollary 1 *Let $q_{f,u} = q_f, M_u = M, L_u = L, \forall u \in \mathcal{U}$ and $L = \{1, \cdots, m\}$, then $p_{f,u} = p_f, \forall u \in \mathcal{U}$ and when $B \to \infty$, $\bar{R}(\mathbf{P}, \mathbf{Q})$ is given by (5.4), where*

$$\bar{m} = \sum_{f=1}^{m} \left(1 - (1 - q_f)^{nL}\right), \tag{5.9}$$

and

$$\bar{M} = \sum_{f=1}^{m} p_f \left(1 - (1 - q_f)^{nL}\right), \tag{5.10}$$

and

$$\psi(\mathbf{P}, \mathbf{Q}) = L \sum_{\ell=1}^{n} \binom{n}{\ell} \sum_{f=1}^{m} \rho_{f,\ell} (1 - p_f M)^{n-\ell+1} (p_f M)^{\ell-1}, \tag{5.11}$$

where $\rho_{f,\ell} \triangleq \mathbb{P}(f = \underset{j \in \mathcal{D}}{\arg\max}\ (p_j M)^{\ell-1}(1 - p_j M)^{n-\ell+1})$ denotes the probability that file f is the file whose p_f maximizes the term $\left((p_j M)^{\ell-1}(1 - p_j M)^{n-\ell+1}\right)$ among $\mathcal{D}$, which is a set of ℓ i.i.d. demands distributed as $\mathbf{q}$. It can be seen that $\rho_{f,\ell}$ is easy to evaluate. □

Corollary 1 can be obtained directly from Theorem 2.

Using the generalized upper bound $\bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q})$ in Theorem 2, we can obtain the desired caching distribution for a wide class of heterogeneous network models. We use $\mathbf{P}^*$ to denote the caching distribution that minimizes $\bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q})$. It is worth to notice that for the homogeneous case described above, where $q_{f,u} = q_f, M_u = M, L_u = 1, \forall u \in \mathcal{U}$, $\bar{R}^{\text{ub}}(\mathbf{P}^*, \mathbf{Q})$ is indeed order optimal, which is proved in [51].

5.2 Polynomial-time Algorithms

In Section 5.1.1, we can see that only the local coloring number $\max_{v \in \mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|$ and the corresponding coloring $\mathbf{c}$ are needed for the coded multicasting delivery.

Hence, in this section, we only focus on the algorithms that achieve these goals. We first introduce a polynomial-time coloring method, which generalizes the coloring algorithm used in [51] to quantify the order-optimal performance of homogeneous shared link networks by letting the number of packets $B \to \infty$. This scheme is based on a greedy constrained local coloring (GCLC) approach, which is a generalization of greedy constrained coloring (GCC) used in [51]. We can prove that when $B \to \infty$, GCLC achieves the upper bound of the average rate for heterogeneous shared link networks given by (5.4). It is also easy to verify that GCLC achieves the same performance as the algorithm given in [51] for the worst-case demand setting in the homogeneous network setting.

We then present a novel coded multicasting algorithm called hierarchical greedy local coloring (HgLC) that fully exploits the structure of the problem and exhibits also a polynomial-time complexity comparable to GCLC. In Section 5.3, we show that for finite file packetization, while GCLC loses the multiplicative caching gain, HgLC is still able to approach the limiting performance and recover a significant part of the multiplicative caching gain.

5.2.1 GCLC (Greedy Constrained Local Coloring)

The GCLC algorithm works by computing two valid local colorings of the conflict graph $\mathcal{H}_{\mathbf{C},\mathbf{W}}$, referred to as GCLC_1 and GCLC_2. GCLC compares the rate achieved by the two coloring solutions and constructs the transmission code based on the coloring with minimum rate. Let $f(v)$ be the packet represented by vertex v, we defined $\mathcal{K}_v = \{\forall u \in \mathcal{U} : f(v) \in \mathbf{W}_u \cup \mathbf{C}_u\}$, where $\mathbf{W}_u$ is the set of all the requested packets by user u and $\mathbf{C}_u$ is the set of all the cached packets by user u. Then GCLC_1 is given by Algorithm 8.

GCLC_1 computes a coloring and the local coloring number of the conflict graph $\mathcal{H}_{\mathbf{C},\mathbf{W}}$. Note that both the outer while-loop starting at line 3 and the inner for-loop starting at line 6 iterate at most $|\mathcal{V}|$ times, respectively. For other operations inside the loops, it take constant time. Therefore, the complexity of GCLC_1 is $O(|\mathcal{V}|^2)$ or equivalently $O(n^2B^2)$ since $|\mathcal{V}| \leq nB$, which is polynomial in $|\mathcal{V}|$ (or n, B).

Algorithm 8 GCLC_1

Let $\mathcal{C} = \emptyset$;
Let $\mathbf{c} = \emptyset$;
while $\mathcal{V} \neq \emptyset$ **do**
  Pick an arbitrary vertex v in $\mathcal{V}$; Let $\mathcal{I} = \{v\}$;
  Let $\mathcal{V}' = \mathcal{V} \setminus \{v\}$;
  for all $v' \in \mathcal{V}'$ with $|\mathcal{K}_{v'}| = |\mathcal{K}_{\tilde{v}}|$ **do**
    if {There is no edge between v' and $\mathcal{I}$} **then**
      $\mathcal{I} = \mathcal{I} \cup v'$;
    end if
  end for
  Color all the vertices in $\mathcal{I}$ by $c \notin \mathcal{C}$;
  Let $\mathbf{c}[\mathcal{I}] = c$;
  $\mathcal{V} = \mathcal{V} \setminus \mathcal{I}$.
end while
return $\max_{v \in \mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|$ and the corresponding $\mathbf{c}(\mathcal{N}^+(v))$ for each v;

On the other hand, GCLC_2 computes the minimum coloring of $\mathcal{H}_{\mathbf{C},\mathbf{W}}$ subject to the constraint that only the vertices representing the same packet are allowed to have the same color. In this case, the total number of colors is equal to the number of distinct requested packets, and the coloring can be found in $O(|\mathcal{V}|^2)$. Then, it remains to find $\max_{v \in \mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|$. It can be seen that this scheme achieve the same results as the random combination of all the requested packets.

Indeed, [52] shows that GCLC is order-optimal when $B \to \infty$ for the homogeneous network with L requests per user when the worst-case demand is considered instead of the average demand.

5.2.2 Hierarchical greedy Local Coloring (HgLC)

Similarly as GCLC, HgLC also works by first computing two valid local colorings of the conflict graph $\mathcal{H}_{\mathbf{C},\mathbf{W}}$, referred to as HgLC_1 and HgLC_2, then it compares the rate achieved by the two coloring solutions and constructs the transmission code based on the coloring with minimum rate.

Let $\mathcal{G}_i = \{v : |\mathcal{K}_v| = i\}$. We consider $\mathcal{G}_i$ as the original hierarchies. For HgLC_1, we start from hierarchy n. After color the vertices in $\mathcal{G}_n$ with the same $|\mathcal{K}_v|$ by the same colors, we merge the rest of the vertices in $\mathcal{G}_n$ with $\mathcal{G}_{n-1}$ ($\mathcal{G}_{n-1} = \mathcal{G}_{n-1} \cup \mathcal{G}_n$, line 33 of Algorithm 9) to result a new hierarchy $n-1$. In the hierarchy $n-1$,

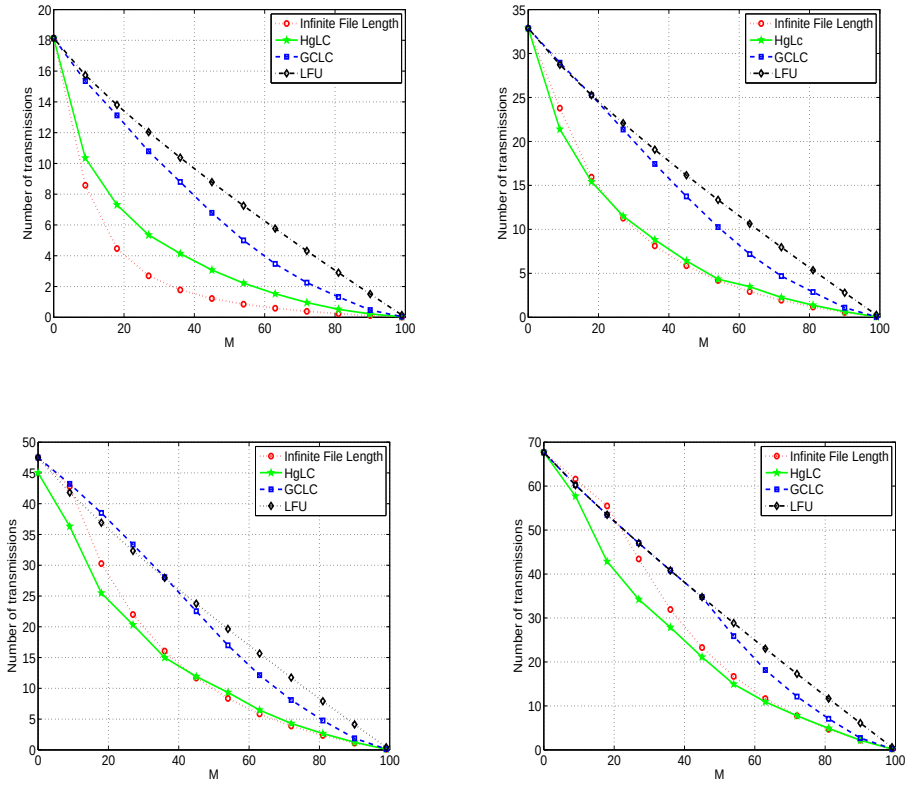


Figure 5.2: Average number of transmission for a shared multicast link with $n = 20$, $n_1 = 5$, $L_1 = \{1, 5, 10, 20\}$, $n_2 = 15$, $L_2 = 1$, $m = 100$, $B = 100$ and $\alpha = 0.2$. a) $L_1 = 1$; b) $L_1 = 5$; c) $L_1 = 10$; d) $L_1 = 20$. Infinite File Length indicate the rate of GCLC when $B \to \infty$ given in Theorem 2.

we again first color the vertices in $\mathcal{G}_n$ with the same $|\mathcal{K}_v|, v \in \mathcal{G}_n$ by the same colors, and then we try to color the rest of the vertices in $\mathcal{G}_{n-1}$. The criteria are that first we randomly pick a vertex v from $\mathcal{W}_2 \in \mathcal{G}_{n-1}$ shown in line 17 in Algorithm 9, where $\mathcal{W}_2$ denote a set of vertices with "small" $|\mathcal{K}_v|, v \in \mathcal{G}_{n-1}$ or "large degree" in $\mathcal{H}_{\mathbf{C},\mathbf{W}}$ and the value of $a \in [0,1]$ control the size of $\mathcal{W}_2$. For example, if $a = 0$, then $\mathcal{W}_2$ denotes the vertex with the smallest $|\mathcal{K}_v|, v \in \mathcal{G}_{n-1}$, which is $n-1$. Second, we try to color the picked vertex v and the other vertices v' whose $|\mathcal{K}_{v'}|, v' \in \mathcal{G}_i \setminus \{v\}$ are "close" to $|\mathcal{K}_v|$ in a greedy manner. Similarly as the parameter $a \in [0,1]$, this closeness is captured by another parameter $b \in [0,1]$ as shown in line 20 in Algorithm 9. For example, if $b = 0$, then we start from the vertex v' such that $|\mathcal{K}_{v'}| - |\mathcal{K}_v|$ is minimized. Here, we are looking for the independent set with size at least i in the ith hierarchy in a greedy manner. After this second coloring procedure, we union the uncolored vertices with the vertices of next hierarchy, which, in this case, is $\mathcal{G}_{n-2}$. Then, we repeat the same procedure for all the hierarchies. Finally, we use a function called LocalSearch to further reduce the number of colors needed in line 37 of Algorithm 9.

The detailed HgLC$_1$ is given by Algorithm 9. Let $\mathcal{N}(j)$ denote the neighbors of vertex j excluding vertex j. The function of LocalSearch is given by Algorithm 10. HgLC$_2$ is the same as GCLC$_2$. It can be shown that the complexity of HgLC$_1$ is given by $O(n^3B^2)$.

5.3 Simulations and Discussion

For finite file packetization by assuming the distributed random popularity-based caching policy in Algorithm 1.

$$\bar{R}^{\text{LFU}} = \sum_{f=\min_u\{M_u\}+1}^{m} \left(1 - \prod_{u \in \mathcal{U}_{\{M_u<f\}}} (1-q_{f,u})^{L_u}\right), \tag{5.12}$$

where $\mathcal{U}_{\{M_u<f\}}$ denote the set of users with $M_u < f$.

For simplicity and to illustrate the effectiveness of HgLC, we consider a homogeneous network scenario, in which users request files according to a Zipf demand distribution with parameter $\gamma = 0.2$ and all caches have size M files. We assume

two types of users: n_1 users with L_1 requests and n_2 user with L_2 requests. Moreover, we let the caching distribution to be uniform, which means that $\mathbf{P}$ is chosen as a m-dimensional vector taking value of $\frac{1}{m}$.[5]

Fig. 5.2 plots the average rate for a network with $n = 20$ users consisting of $n_1 = 5$ and $n_2 = 15$, $L_1 \in \{1, 5, 10, 20\}$, $L_2 = 1$, $m = 100$ files and $B = 100$ packets. Observe how the significant caching gains (with respect to LFU) quantified by the upper bound are completely lost when using GCLC with finite packetization $B = 100$. On the other hand, observe how HgLC remarkably preserves most of the promising multiplicative caching gains for the same values of file packetization. For example, in Fig. **??**, if M doubles from $M = 25$ to $M = 50$, then the rate achieved by HgLC essentially halves from 11.5 to 5. For the same regime, it is straightforward to verify that neither GCLC nor LFU exhibits this property.[6]

Note from Fig. 5.2(a), that in oder to guarantee a rate of 7, GCLC requires a cache size of $M = 45$, while HgLC can reduce the cache size requirement to $M = 18$, a $2.5\times$ cache size reduction.

Indeed, [52] shows that as the increase of L_u, the gain obtained by local coloring is also increasing, which can be observed for both GCLC and HgLC in Fig. **??**-**??**. It is worth to notice that Fig. **??**-**??** also show a almost same or even better performance of HgLC compare to GCLC with $B \to \infty$, which tells that as L increases, the local coloring can also compensate the loss due to the finiteness of the packetization of each file. This can be explained by the fact that under a finite library size, more requests can result in more correlation between the requests of users. By using this correlation, each user can decode its requests jointly or can use the previously decoded information to decode new packets.

5.4 Conclusions

In this paper, we show that the promising multiplicative caching gain analytically quantified for the shared link caching network can be completely lost in finite

[5]The caching distribution $\mathbf{P}^*$ can be obtained by minimizing $\bar{R}^{\mathrm{GCLC}}(\mathbf{P}, \mathbf{Q})$ in (5.4) among all $\mathbf{P}$ described by a m-dimensional vector taking value in $\{\frac{1}{m}, 0\}$ in practice, as suggested in [51].

[6]While LFU can only provide an additive caching gain, additive and multiplicative gains may show indistinguishable when M is comparable to the library size m. Hence, one needs to pick a reasonably small M ($\frac{m}{n} < M \ll m$) to observe the multiplicative caching gain of HgLC.

regimes of the system parameters. We first extend the analysis of this caching network to the case of heterogeneous cache sizes, demand distributions and number of requests per user, providing an upper bound on the limiting average performance when the number of packets per file goes to infinity. We then focus on finite regimes of all system parameters and show that the greedy constrained local coloring (GCLC) scheme used to quantify this upper bound quickly loses the multiplicative caching gain for finite file packetization. We then design a novel polynomial-time coded multicasting scheme based on a greedy hierarchical local coloring (HgLC), which is able to recover a significant part of the multiplicative caching gain with the same finite file packetization. Our results, while initially negative, shed light on the possibilities to preserve the multiplicative caching gain via careful design of coded multicasting schemes for finite values of the system parameters.

Algorithm 9 HgLC_1

$\mathcal{C} = \emptyset$;
$\mathbf{c} = \emptyset$;
choose $a \in [0, 1]$
choose $b \in [0, 1]$
for all $i = n, n-1, \cdots, 2, 1$ **do**
  for all $v \in \mathcal{G}_i$ and $|\mathcal{K}_v| = i$ **do**
    $\mathcal{I} = \{v\}$;
    for all $v' \in \mathcal{G}_i \setminus \mathcal{I}$ with $|\mathcal{K}_{v'}| = |\mathcal{K}_v|$ **do**
      if {There is no edge between v' and $\mathcal{I}$} **then**
        $\mathcal{I} = \mathcal{I} \cup v'$;
      end if
    end for
    if $|\mathcal{I}| = i$ **then**
      Color all the vertices in $\mathcal{I}$ by $c \notin \mathcal{C}$;
      $\mathbf{c}[\mathcal{I}] = c$, $\mathcal{C} = \mathcal{C} \cup c$;
      $\mathcal{G}_i = \mathcal{G}_i \setminus \mathcal{I}$;
    end if
  end for
  for all Randomly pick a $v \in \mathcal{W}_1 \subset \mathcal{G}_i$, with $\mathcal{W}_1 = \{v \in \mathcal{G}_i : \min_{v \in \mathcal{G}_i} |\mathcal{K}_v| \leq |\mathcal{K}_v| \leq \min_{v \in \mathcal{G}_i} |\mathcal{K}_v| + \lfloor a \left(\max_{v \in \mathcal{G}_i} |\mathcal{K}_v| - \min_{v \in \mathcal{G}_i} |\mathcal{K}_v|\right) \rfloor\}$ **do**
    $\mathcal{I} = \{v\}$;
    $\mathcal{Q}_i = \mathcal{G}_i \setminus \mathcal{I}$;
    for all Randomly pick a $v' \in \mathcal{W}_2 \subset \mathcal{Q}_i$, with $\mathcal{W}_2 = \{v' \in \mathcal{Q}_i : \min_{v' \in \mathcal{Q}_i} |\mathcal{K}_{v'}| \leq |\mathcal{K}_{v'}| \leq \min_{v' \in \mathcal{Q}_i} |\mathcal{K}_{v'}| + \lfloor b \left(\max_{v' \in \mathcal{Q}_i} |\mathcal{K}_{v'}| - \min_{v' \in \mathcal{Q}_i} |\mathcal{K}_{v'}|\right) \rfloor\}$ **do**
      if {There is no edge between v' and $\mathcal{I}$} **then**
        $\mathcal{I} = \mathcal{I} \cup v'$;
        $\mathcal{Q}_i = \mathcal{Q}_i \setminus \{v'\}$;
      else
        $\mathcal{Q}_i = \mathcal{Q}_i \setminus \{v'\}$;
      end if
    end for
    if $|\mathcal{I}| \geq i$ **then**
      Color all the vertices in $\mathcal{I}$ by $c \notin \mathcal{C}$;
      $\mathbf{c}[\mathcal{I}] = c$, $\mathcal{C} = \mathcal{C} \cup c$;
      $\mathcal{G}_i = \mathcal{G}_i \setminus \mathcal{I}$;
    else
      $\mathcal{G}_i = \mathcal{G}_i \setminus \{v\}$, $\mathcal{G}_{i-1} = \mathcal{G}_{i-1} \cup \{v\}$;
    end if
  end for
end for
$\mathbf{c}$ =LocalSearch($\mathcal{H}_{\mathbf{C},\mathbf{W}}, \mathbf{c}, \mathcal{C}$);
return $\max_{v \in \mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|$ and the corresponding $\mathbf{c}(\mathcal{N}^+(v))$ for each v;

Algorithm 10 LocalSearch($\mathcal{H}_{\mathbf{C},\mathbf{W}}, \mathbf{c}, \mathcal{C}$)

for all $c \in \mathcal{C}$ **do**
  Let $\mathcal{J}_c$ be the set of vertices whose color is c;
  Let $\mathcal{B} = \emptyset$;
  Let $\hat{\mathbf{c}} = \mathbf{c}$;
  for all $i \in \mathcal{J}$ **do**
    $\mathcal{A} = \emptyset$;
    for all $j \in \mathcal{N}(j)$ **do**
      $\mathcal{A} = \mathcal{A} \cup \mathbf{c}[j]$;
      if $\mathcal{C} \setminus \mathcal{A} \neq \emptyset$ **then**
        c' is randomly picked from $\mathcal{C} \setminus \mathcal{A}$;
        $\hat{\mathbf{c}}[i] = c'$;
        $\mathcal{B} = \mathcal{B} \cup \{i\}$;
      end if
    end for
    if $|\mathcal{B}| = |\mathcal{J}|$ **then**
      $\mathbf{c} = \hat{\mathbf{c}}$;
      $\mathcal{C} = \mathcal{C} \setminus c$;
    end if
  end for
end for
return $\mathbf{c}$;

Chapter 6

Coding for Caching in 5G Networks

This Chapter aims at bridging the gap between theory and practice in order to validate the benefits of caching-aided coded multicasting by designing a fully working prototype implementation and testing it in a large network testbed. Such testbed and prototype implementation provide a cornerstone for the evaluation of future schemes with more advanced wireless caching protocols and cache-enabled PHY layer techniques such as joint source-channel coding. We first provide an overview of the caching and coded multicasting framework and discuss the key concepts behind the ability to provide load reductions that are proportional to the aggregate cache size. We then introduce a new frame structure that includes specific fields to account for all the practical aspects required for a fully working real-world implementation. The primary role of the newly designed frame structure is to allow decoding of coded data at each receiver. Our MAC layer frame design is combined with an orthogonal frequency division multiplexing (OFDM) PHY layer, which makes it compatible with long term evolution (LTE) advanced mobile networks or further PHY layer standards. The resulting fully working prototype is implemented in a large-scale testbed facility, CorteXlab [28], composed of tens of highly flexible radio nodes deployed in a controlled and reproducible environment. We present experimental results in the context of key 5G challenges related to transmission delay, bandwidth usage, and energy efficiency. Our experimentation validates the fact that memory can be effectively turned into bandwidth leading to substantial network throughput gains.

6.1 Implementation of caching-aided coded multicasting

While Chapter 3 describes state of the art wireless caching and transmission code design, the impact of real protocol overheads on the multiplicative caching gain remains an open question that we address via a real prototype implementation in the following.

Our prototype implementation is based on the following components: i) a simplified application layer for generating and combining the requested chunks, ii) a MAC layer extended with additional header fields to allow decoding of coded packets, and iii) a PHY layer compliant with LTE standards. Our basic MAC layer frame implementation does not account for a complete standardized frame structure and the generated data is not encapsulated through the protocol stack, since our main goal is a proof-of-concept of caching-aided coded multicasting and its real-time feasibility. In the following, we describe in detail the MAC layer frame structure.

6.1.1 Frame structure

For a clear understanding of the implementation process, the basic frame structure is given in Fig. 3.3. Every accumulated packet is composed of two parts: header and payload. The payload represents the coded packet (divided as: $payload_1, \cdots, payload_K$); a mixture of original data chunks with elements in the Galois Field of order two GF(2), making it easy to encode and decode with a simple XOR operation. The header illustrated in Fig. 3.3.c contains the minimal information required for a successful extraction of each individual chunk. It carries the combined chunks identities and consists of the following information:

- Header Length: This is the first element of the header, and its size is fixed to one byte. It carries the number of header bytes.
- File IDs: These are the IDs of the files to which the combined chunks (payload) belong. Each ID requires a multiple of one byte, depending on the number of content files in the library.

- Chunk IDs: These are the IDs of the combined chunks within the transmitted packet. Each ID requires a multiple of one byte, depending on how many chunks a file is partitioned into.

- Chunk sizes: These are the sizes of the combined chunks, and are encoded with a multiple of four bytes to make the receiver able to recognize the size of each chunk.

In a practical network scenario, it is unusual to have a header length exceeding one byte since the number of requests and the number of simultaneously served users is generally limited. Notice that the uncoded design will necessitate the same header structure but only for one target user (unless the same packet is destined to multiple users) because no packet combination is performed. This means that in an uncoded scheme each packet is separately transmitted with its associated header, without the need for additional overhead information. This is due to the fact that we are assuming multicast transmission over the downlink shared channel (DL-SCH). This LTE physical layer transport channel is the main channel for downlink data transfer and it is used by many logical channels. The fact that the header information in the coded scheme depends on the number of served users implies a variable header length. An example of the header decomposition is illustrated in Fig. 3.3.c, where the number of files and chunks are assumed to not exceed one byte each, and the maximum size of a chunk is limited to four bytes. The payload length of a coded packet is equal to the largest chunk's length among the combined ones. Before being transmitted, the coded packet is partitioned into small packets and numbered such that the receiver can rebuilt the original coded packet. Each coded packet (see Fig. 3.2) is dedicated to users with IDs indicated in the header information. Aiming at decreasing the packet error probability (PER), the first small packet will be limited to the header data, and the others are charged with the payload. A $32-$bit cyclic redundancy check (CRC) is appended to each small packet for error detection. In the header information, if the CRC detects some errors the whole coded packet is lost and the user drops all related small packets. Otherwise, each user checks whether concerned or not. If so, the user proceeds to the small packet decoding, based on its cached data and the reported files and chunks IDs. Conversely, if the user is not concerned the packet is dropped and

the user waits for the next header to check out its affiliation. In case of channel erasure, the small packets are replaced with dummy bytes.

6.1.2 CorteXlab platform

The resulting fully working prototype is implemented in a large-scale testbed facility, CorteXlab [28] which is a testbed for cutting edge radio experimentation, composed of a mix of radio nodes, including SISO and MIMO software defined radio (SDR) nodes. The testbed shown in Fig. 6.2 is installed in a large (180 m^2) shielded room partly covered with electromagnetic wave absorbing material. User nodes are placed over a regular grid with an inter-node distance of 1.8 meters, and accept any PHY layer implementations on both hardware and software. A unified server is available for starting, coordinating, and collecting the results of experiments. As a development tool, the GNU Radio software is employed for real-time experimentation.

6.2 End-to-end performance results and perspectives

6.2.1 Setup Environment

Our SLN experimentation consists of one radio source node and $n = 10$ radio user nodes. Each user requests $L = 10$ files from a library $\mathcal{F}$ of $m = 20$ binary files, each of size 2.8 Mb. A cache of size M files is deployed at every user. Such a scenario can be seen as if the users are APs carrying multiple requests from different UEs, and the source is the eNB having access to the content library. The file request distribution is drawn from a Zipf distribution with Zipf parameter α: $\alpha = 0$ returns a uniform request distribution; the higher the Zipf parameter α, the more skewed is the request distribution. The binary files are partitioned into equally sized $B = 100$ chunks yielding a library of $m_b = 2000$ chunks. Both RLFU with $\tilde{m}$ optimized as in [30] and LFU caching policies are adopted for the coded and naive multicasting delivery schemes, respectively.

The output of multicast encoder goes into an OFDM modulator with the following transmission parameters. Each PHY frame is decomposed into small packets

of size 100 bytes to which a CRC-32 and an OFDM header are appended for error detection and OFDM demodulation, respectively. The OFDM header and payload data are mapped into a binary phase shift keying (BPSK) and a quadrature PSK (QPSK), respectively, and each symbol is transmitted over a sample duration of $T_s = 1\mu$s. The data is carried over $L_f = 48$ subcarriers spaced by $\Delta f = 15$KHz and the central frequency is set to 2.45 GHz.

6.2.2 Experimentation results

The focus herein is on the gain at the MAC layer that is based on counting the total number of required bytes to serve all UEs. Assuming the same number of requests L from all users, the normalized minimum rate (NMR) is defined as $R_t/(L \times \text{file size})$, where R_t is the total number of required bytes at the MAC layer to satisfy all user demands. Note that NMR is in general a non-decreasing function of the number of users, and a decreasing function of cache size, M; in particular, for $M = 0$, the NMR is equal to the total number of distinct user requests. In the following, we provide a numerical validation of the prototype performance in terms of NMR. Specifically, we analyze the performance of our prototype solutions prot-HGC and prot-GRASP in terms of NMR compared with: *i*) HGC and GRASP for finite file packetization simulated in Matlab environment without taking into account implementation overhead *ii*) Naive multicasting with LFU caching policy at the rate of the worse channel receiver, and *iii*) the benchmark upper bound GCC when $B = \infty$ (see [30]). The trend in terms of NMR demonstrated by the prototype confirms the gains predicted by the theory. Figs. 6.3.a and 6.3.b show the NMR as function of the cache size and the Zipf parameter α respectively. This metric is specially illustrative of the amount of bandwidth resources the wireless operator needs to provide in order to meet the receiver demands. In Fig. 6.3.a, we assume a Zipf parameter $\alpha = 0$. Observe first the performance of naive-multicast. As expected, the load reduces approximately linearly with the cache size M. Observe, now, how the significant multiplicative caching gains (w.r.t. naive-multicast) quantified by the upper bound (RLFU-GCC with $B = \infty$) are remarkably preserved by the prototype solutions (prot-HGC and prot-GRASP) which achieve an NMR almost indistinguishable from the corresponding schemes

implemented in Matlab environment without taking into account the encoding and decoding overheard. Fig. 6.3.a clearly shows the effectiveness of the proposed implementation in allowing receivers to decode their requested files at an NMR very close to the theoretically optimal NMR. From Fig. 6.3.a, it is also apparent that the two coloring algorithms have similar performance for $\alpha = 0$. The effectiveness of coded multicasting is highly influenced by the Zipf parameter α, as illustrated in Fig. 6.3.b for cache size $M = 2$ files. Observe how the reduction in NMR enabled via coded multicasting is much more attractive in the region of $\alpha \leq 1$.

In order to illustrate the behavior of the multiplicative gains, Figs. 6.3.c and 6.3.d show the prototype NMR gains as a function of the cache size M and the Zipf parameter α, respectively. The gain of a given scheme is defined as the ratio between the NMR achieved by naive multicasting with LFU caching policy and the NMR achieved by that scheme. In particular, when the scheme is a prototype implementation, then the NMR of naive multicasting is computed with its associated overhead. From Fig 6.3.c, we can observe that the gain is a monotonic non-decreasing function of the cache size. Note that we do not plot the point at $M = 20$ since it is well known that the NMR is zero for all the schemes, and hence the gain is given by an indeterminate form of type 0/0. Fig 6.3.c shows that the gains achieved by prot-GRASP and prot-HgC are very close to the gains achieved by the corresponding MATLAB simulated schemes, confirming the little impact of the implementation overhead on the overall performance. Furthermore, it is worth noticing that due to the reduced number of transmitted coded packets compared to the number of uncoded packets transmitted by naive multicasting, the total overhead size is also smaller. That is, even though each packet header length is larger, the total number of header bytes over all transmissions is also reduced.

In terms of the Zipf parameter α, Fig. 6.3.d shows that for $M = 2$ files, a gain close to 1.3 is obtained under uniform popularity ($\alpha = 0$), and this gain tends to 1 as the popularity distribution becomes more skewed.

6.2.3 Turning memory into bandwidth

In this section, we evaluate the physical layer bandwidth gains enabled by coded multicasting. To do so, we assume a fixed video transmission delay (e.g., according

to users' QoS) and evaluate the bandwidth required to serve the video segment requests of all users. Fig. 6 illustrates the bandwidth gain (BG) evolution at the PHY layer with respect to the number of users for different cache sizes. We define the PHY BG as the bandwidth required to serve all requests via naive multicasting over the bandwidth required via the use of coded multicasting. The increase in BG can be clearly observed with respect to both the cache size and the number of users. For instance, assuming a cache size $M = 10\%$ of the library size, the gain starts with a value around 1.1 for 5 users and goes up to 1.31 for 40 users. Similarly, at $M = 30\%$, the gain increases from around 1.4 for 5 users and reaches around 1.68 for 40 users. The increase of the BG with respect to the number of users is specially relevant, as it illustrates the scalability benefits of coded multicasting.

6.2.4 Future directions

In the above, coding overhead and computational complexity have been proven not to limit the performance gain of wireless caching for coded multicasting. However, several open problems related to PHY layer protocols are currently under investigation, among which we cite the following:

- Variation of the channel characteristics: Regarding the variations of channel statistics across users (e.g., different SNRs), the work in [3] provided a theoretical analysis that takes into account the wireless channel characteristics in the presence of any combination of unicast/multicast transmission and wireless edge caching. They proposed a channel-aware caching-aided coded multicast scheme based on joint source-channel coding with side information. Such scheme is able to guarantee a rate to each receiver, within a constant factor of the optimal rate, had the remaining users experienced its same channel conditions. The scheme preserves the cache-enabled multiplicative throughput gains by completely avoiding throughput penalization from the presence of receivers experiencing worse propagation conditions. The implementation of this scheme in CorteXlab is part of our next steps for future work. As opposed to network emulation platforms such as in [71], CorteXlab will allow us to properly test user mobility and realistic channel degradation across wireless end points.

- Combination with MIMO schemes: The use of MIMO schemes is an interesting topic with significant active research. Undergoing studies such as [72] have shown that in a MIMO setting, coded multicasting is indeed complementary to MIMO, and the combination of both provides cumulative gains in most practical scenarios. This is also object of future work, and again, CorteXlab represents a key advantage in order to easily include next generation radio technologies.

- Dynamic scenarios: Our current implementation setting is limited to static scenarios with respect to file popularity and number of users. Ideas related to cache adaptation with respect to dynamic popularity distributions and varying number of users are of interest for future work and currently under investigation.

6.3 Conclusions

This chapter discusses the potential of caching-aided coded multicasting for improving bandwidth efficiency in next generation wireless access networks. A real-time implementation for performance evaluation in real environments has been presented. On the way from theory to practical evaluation, a complete frame structure for the transmitter and the receiver has been proposed. Interestingly, the additional coding overhead does not compromise performance and leads to an overall positive multicasting gain, reducing bandwidth requirements and transmission delay when compared to the best uncoded schemes. We have integrated the coded multicast design in an OFDM based PHY layer, and deployed the scenario in CorteXlab, a shielded experimentation room, using radio nodes. Our work also shows the potential of such facility to validate new concepts relative advanced radio technologies for 5G networks. Finally, we have briefly described interesting open problems related to PHY layer protocols that are currently under investigation.

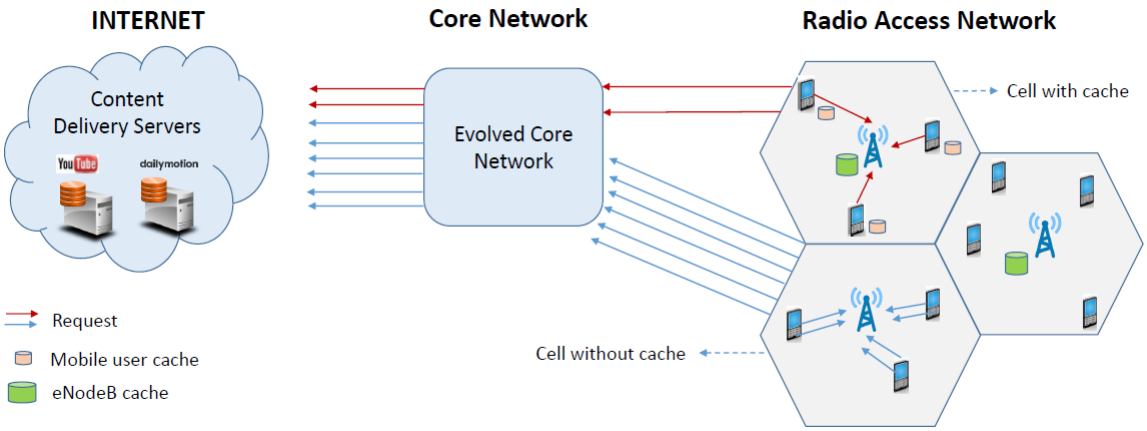


Figure 6.1: Caching within the radio access network: impact on network load and traffic congestion.

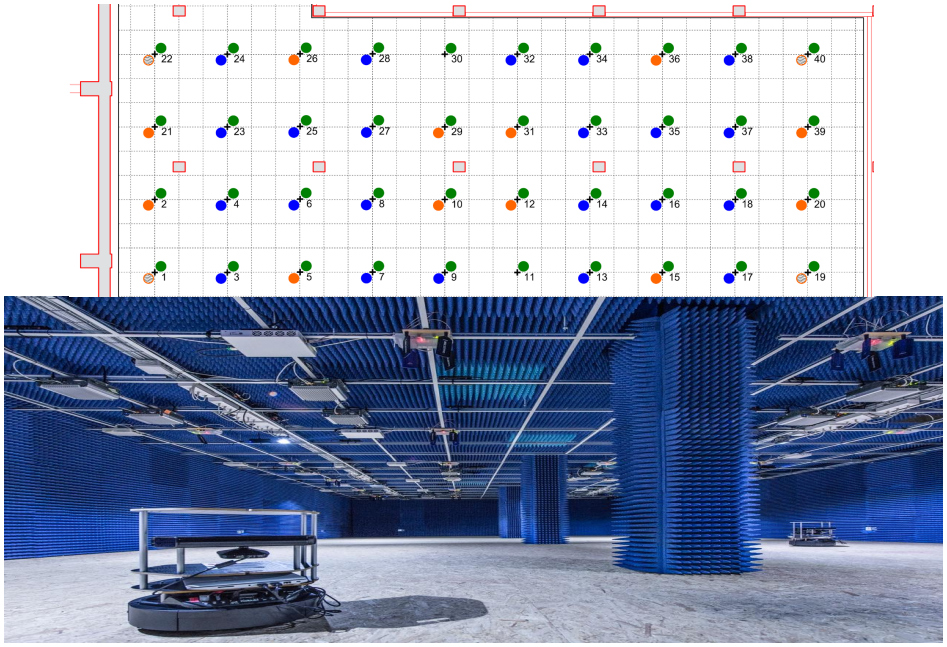


Figure 6.2: CorteXlab platform and the nodes placement map.

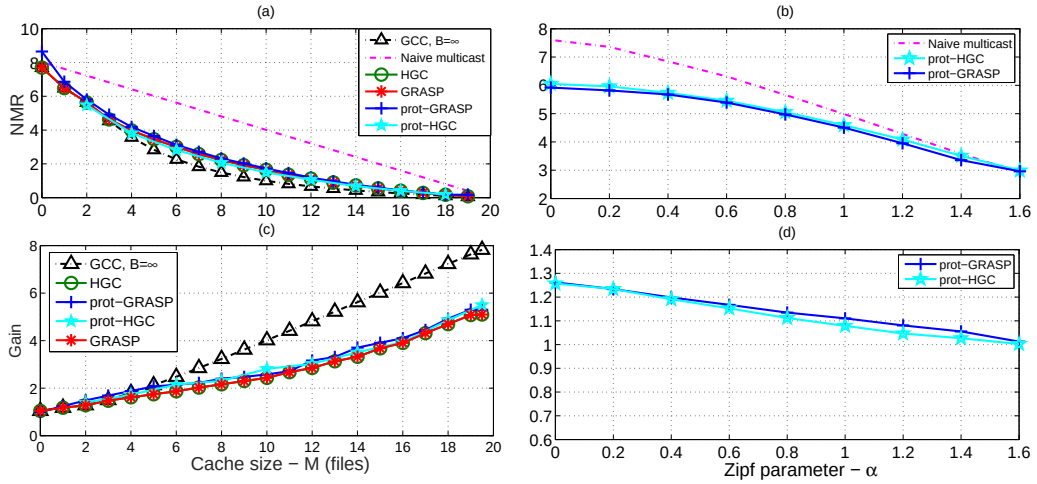


Figure 6.3: Comparison of the total minimum rate normalized by the 1-user network minimum rate with respect to : a) the cache size with $\alpha = 0$, and b) the Zipf α parameter with $M = 2$ files, and the gain of the coded over the uncoded scheme with respect to: c) the cache size with $\alpha = 0$, and d) the Zipf α parameter with $M = 2$ files.

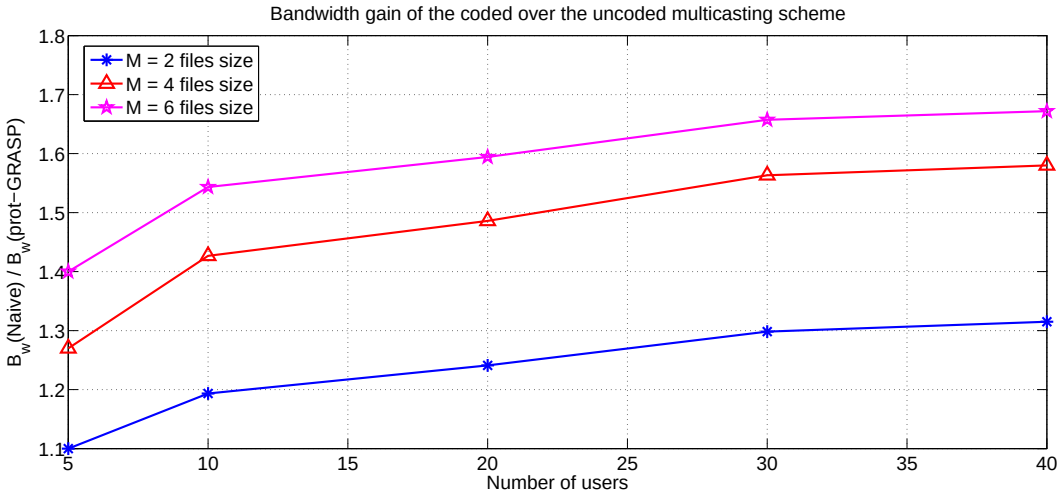


Figure 6.4: The bandwidth gain of coded multicasting over naive multicasting for different memory cache sizes.

Chapter 7

Video Coding Using Receiver-Side Memory and Index Coding

In this Chapter, we utilize the tools of index coding to efficiently identify correlations between video frames already in the receiver-side memory and frames that are demanded by receivers in future requests. This results in a coded message that aims to satisfy all the requests with minimal redundancy and thus saves bandwidth utilization. Using video frames as side information has been extensively studied in the context of Distributed Video Coding (DVC) –see [21] and references therein, however to our knowledge none of these works consider video data already stored in a receiver cache, instead, they make use of alternate frames or other subsets of newly transmitted video as side information.

The remainder of this Chapter is organized as follows. We first introduce in the Section 7.1 index coding and its application to video. The achievable correlation-aware caching scheme are presented in Section 7.2. Section 7.2.1 describes the proposed polynomial-time algorithm which exploits the library correlation in order to minimize the total number of *Conflict Graph's* edges. Finally, Section 7.3 presents the simulation results to show the performance gain of video transmitted with correlation-aware caching scheme over conventionally transmitted video, and we conclude the Chapter in Section 7.4.

7.1 Index Coding

Index coding – see [69, 13, 7, 9, 57], is a coded multicast technique that aims to satisfy the demands of multiple receivers while making using of distributed side information stored in receiver-side caches. In the usual setting, multiple receivers are attached to a single transmitter over a single bottleneck link. In the *placement* phase, data is distributed and stored at receiver-side caches without definite knowledge of future demands that the receivers may make. This is assumed to occur at off-peak hours when network resources are abundant, or in the past to satisfy earlier user demands, and not incur a cost. During the second *delivery* phase, receivers make their demands known to the transmitter, and the transmitter uses its knowledge of the contents of the caches to transmit a codeword that satisfies the demands, utilizing the information already at the receivers to reduce bandwidth requirements. Since placement occurs without knowledge of the demand distribution, it may be implemented uniformly [69], or according to a probability distribution [57]. In the DVR scenario considered here, placement occurs due to previous viewer requests, and future requests are assumed to be at least partly correlated to the previous ones due to unchanging viewer tastes.

Given the distributed cache contents and the new demand from the receivers, a codeword is generated based on chromatic number index coding [13] or rank minimization [7].

7.1.1 Application to Video

We illustrate index coding with a simple example. Consider two receivers R0 and R1 with memories that each contain one bit value, denoted **a** and **b**. They now request **b** and **a** respectively. An encoder that is aware of the contents of the memories at R0 and R1 needs to transmit only $a \oplus b$ to satisfy both requests, at a cost of 1 bit transmitted (instead of 2 bits for a naive encoder).

Now assume that an encoder contains *uncoded* video frames $F0_u$ and $F1_u$, and receiver R0 requests $F0_u$ and R1 requests $F1_u$. Let $C_b(X)$ denote intracoding of a frame X with b bits per pixel. The encoder transmits the coded sequences $C_b(F0_u)$ and $C_b(F1_u)$ at a total cost of 2b bits per pixel. At Ri, $i \in \{1, 2\}$, $C_b(Fi_u)$ is decoded to produce a reconstructed version Fi that is distorted with

respect to Fi_u. Next R0 and R1 demand $F1$ and $F0$ respectively. (They may demand $F1_u$ and $F0_u$, but are satisfied with $F1$ and $F0$). The simplest method is to xor the coded representations of $F0$ and $F1$, but additional bitrate savings are possible by exploiting correlations between these frames. This motivates the following methods for the encoder to satisfy the request.

1. Coded Domain XOR: Transmit $C = C_b(F0_u) \oplus C_b(F1_u)$, where $\oplus$ is bitwise xor, with rate b bpp. At R0, $C \oplus C_b(F0_u) \rightarrow C_b(F1_u) \rightarrow F1$, and similarly R1 recovers $F0$. This reconstructs $F0$ and $F1$ exactly with rate b bpp.

2. Pixel Domain XOR:

   (a) Let $F_u = F0 \oplus F1$, where $\oplus$ is bitwise xor in the pixel domain. Transmit $C_{b_p}(F_u)$, with rate b_p bpp, and reconstruct F, a distorted version of $F0 \oplus F1$, at each receiver. Reconstruct at R0: $F \oplus F0 \rightarrow F1_p$, a distorted version of $F1$; At R1: a distorted version $F0_p$.

   (b) Similar to above but with $F_u = F0_u \oplus F1_u$.

3. Motion Estimation [78]:

   (a) Let $E_{b_e}(F0, F1)$ represent motion estimation of F1 using F0 as reference followed by residual coding using a total bit rate of b_e bpp. The motion vectors and residual are transmitted with cost b_e bpp. At R0: Frame $F0$ is used with the motion vectors and residual to reconstruct $F1_r$, a distorted version of $F1$. At R1: a distorted version $F0_r$ is obtained using reversed motion vectors and $F1$ as reference.

   (b) Similar to above but with $E_{b_e}(F0_u, F1_u)$.

Correlation may also be exploited in method (1), but correlation in the pixel domain may translate inexactly into bitwise correlation between the coded representations $C_b(F0_u)$ and $C_b(F1_u)$. Also, an additional coding step such as run length encoding is needed to further reduce the bit rate below b bpp. It is unclear which method yields the best rate-distortion performance for a given pair of frames $F0$ and $F1$.

We will exploit the correlation between frames.

7.1.2 Generating the codeword

We utilize chromatic number index coding[13, 57], further developed in [45] to handle inexact correlations between cache contents and data demands. We assume n receivers $R_1, R2, ..., R_n$ each with a cache that contains stored video frames. Each frame in a cache stored in coded form, for example $C_F(Fi_u)$, requires F bits, and each cache has a capacity M frames. We denote by $C = [C_1, ..C_n]$ the cache configuration, where C_u denotes the frame cached at receiver R_u. Let $Q = [Q_1, ..Q_n]$ denote the demands from the receivers, where Q_u is the set of frames requested by receiver R_u and not already in its cache.The encoder knows both Q and C. In our case, we consider:

1. $C_i \neq C_j, \forall C_i, C_j \in C, i \neq j$;
2. $Q_i \neq Q_j, \forall Q_i, Q_j \in Q, i \neq j$;
3. $C_i \neq Q_j, \forall C_i \in C, Q_j \in Q.$;

We first describe a simple graph coloring based technique that exploits reciprocity between the cache contents and the demanded frames to identify index codewords. This is then extended using clustering to increase possible coding opportunities.

Finding best-matching frame: For each frame $a' \in Q$ that is requested by a receiver, the encoder searches all frames known to be in the caches (these are also assumed present at the encoder) and all frames known to be in the demands, to find the frame a that best matches a'. There are a lot of methods to compute the best matching, for instance, we could computed the best matching in the sense of minimizing the mean square error $\frac{1}{N}||a - a'||^2$ where N is the frame size in pixels. Other measures are also possible, such as the conditional entropy $H(a'|a)$. In our case we have used a very common algorithm used to compress the frames in video. In the field of video compression a video frame is compressed using different algorithms with different advantages and disadvantages, centered mainly around amount of data compression. These different algorithms for video frames are called picture types or frame types. The three major picture types used in the different video algorithms are I, P and B [78] (see Figure 7.1). They are different in the following characteristics:

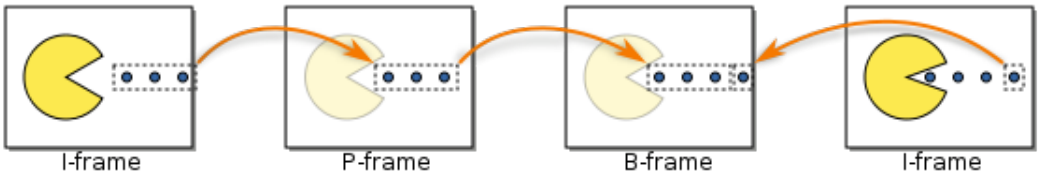


Figure 7.1: A sequence of video frames, consisting of two key-frames (I), one forward-predicted frame (P) and one bi-directionally predicted frame (B).

- I-frames (Intra-coded picture) are the least compressible but don't require other video frames to decode. An I-frame is a complete image, like a JPG or BMP image file. P and B frames hold only part of the image information (the part that changes between frames), so they need less space in the output file than an I-frame.

- P-frames (Predicted picture) can use data from previous frames to decompress and are more compressible than I-frames. For example, in a scene where a car moves across a stationary background, only the car's movements need to be encoded. The encoder does not need to store the unchanging background pixels in the P-frame, thus saving space. P-frames are also known as delta-frames.

- B-frames (Bidirectional predicted picture) can use both previous and forward frames for data reference to get the highest amount of data compression.

Let $H(a')$ represent the bit rate of coding frame a' without any reference (up-to a certain distortion), and $H(a'|a)$ the bit rate of coding a' using a as reference(up-to the same distortion). In our case the I-frame is a reference frame and the P-frame is the requested frame. After the coding process, the coded P-frame will be our refinement in order to get a' from a. These costs include the bits needed to identify a' and a in sets C and Q, which we treat as generalized motion vectors. Denote by $C(a')$ and by $C(a'|a)$ the coded representations of a' without any reference, and with a as reference. These include index bits needed to identify a' and a in sets C and Q. Define a transmission list T which contains the set of codewords that will be transmitted.

Graph Coloring A *conflict graph* $G = (V, E)$ is constructed as follows:
For each frame $a' \in Q$ requested by receiver R_i, if $H(a') > H(a'|a)$:

1. One vertex $v \in V$ defined as $v = \{a, R_i\}$ is constructed, unless

   (a) It is already present.

   (b) a is present at the same receiver cache R_i that requested a'.

   If the frame is requested by more than one receiver, a separate vertex is constructed for each such request.

2. If any vertex containing a is constructed, $T \leftarrow T \cup C(a'|a)$.

If $H(a') \leq H(a'|a)$, $T \leftarrow T \cup C(a')$.

1. One vertex $v \in V$ defined as $v = \{a', R_i\}$ is constructed, unless it is already present.
   If the frame is requested by more than one receiver, a separate vertex is constructed for each such request.

2. If any vertex containing a' is constructed, $T \leftarrow T \cup C(a')$.

Denote by $B(v)$ the frame indicated in vertex v, and by $R(v)$ the receiver indicated in vertex v.

Create an edge between vertices v_1 and v_2 if

1. $B(v1) \neq B(v2)$, and

2. $B(v1)$ is not present in cache at $R(v2)$, or $B(v2)$ is not present in cache at $R(v1)$.

Next consider a minimum vertex coloring of the conflict graph $G = (V, E)$. All vertices with the same color are used to generate a single codeword, which contains (i) indices to identify the frames in these vertices in the receiver caches, (ii) an encoding of the frames themselves. This may be done by Coded Domain XOR, Pixel domain XOR, or arithmetic difference. (For example, sum of first half - sum of remaining frames). For each vertex set, all these options may be computed and

the most compact one selected. If $B(v) = a$, $\forall v \in V$, the sender will send (in Unicast) the refinement to $R(v)$ in order to allow the receiver to get the frame a' from its reference frame a.

Improved coding: Selecting the single best-matching frame in the caches for the requested frame a' may not be optimal. For example, consider the 3-cache configuration $C = \{\{b_1\}, \{a\}, \{b_2\}\}$ and demands $Q = \{\{a'\}, \{b'\}, \{\}\}$ where $||b' - b_2||^2 < ||b' - b_1||^2$ and $||a' - a||$ is small. Here the best matching frame for b' is b_2, but b_1 provides a reciprocal coding opportunity. (Transmit $a \oplus b_1$, then a' encoded with reference to a, and b' with reference to b_1.)

To take advantage of such opportunities, we extend the above. For each frame $a' \in Q$ that is requested by a receiver, define a list $L_{a'}$ of frames a in the caches such that $H(a'|a) < H(a')$, i.e. $L_{a'}$ contains frame that can be used as references for coding a' at less cost than coding a' without reference. Assume that there are n frames $a'_1, a'_2..., a'_n$ in Q.

Create a separate instance of the conflict graph G, denoted $G(\alpha_1, \alpha_2..\alpha_n)$, for each selection $(\alpha_1, \alpha_2..\alpha_n) \in L_{a'_1} X L_{a'_2}..X L_{a'_n}$, where X denotes Cartesian product.

Then the optimal graph coloring is the one that yields minimum total rate over all $G(\alpha_1, \alpha_2..\alpha_n)$.

7.2 Achievable Scheme

In this section, we present an achievable correlation-aware caching scheme based on random popularity-based caching and index coding based delivery.

7.2.1 Replacement Algorithm (REPA)

The sender (i.e. Base Station), in order to send to each receiver the requested frames, it will construct a *Conflict Graph* [83] $G = (V, E)$. In order to identify the total number of data transfers needs to satisfy all the users requests, the *Conflict Graph*'s vertices have to be colored, for this step the sender will use the *HGLC* [54] algorithm which will provide to color each vertex based on the following constraint: two vertices connected by a edge cannot be colored with the same color. The total number of colors is the total number of data transfers, and the set of the frames

indicated by the vertices which have the same color will be coded (Coded Domain XOR) and will be sent together. The method that we used to reduce considerably the number of colors tries to decrease the total number of edges in the *Conflict Graph*, in detail, one vertex has a hight degree (total number of edges connected to it) if the frame indicated by itself is required or is cached by few users, for this reason the idea of our method is to try to find for each requested frame its best reference frame which is requested or cached by a higher number of users. We denote with C the cache configuration, with Q the demand, with $a' \in Q$ the generic frame requested by the receiver and with $a \in Q \cup C$ the reference frame of a', moreover we denote with $RAP_q(a')$ the set of reference frames of a' in the demand, with $RAP_c(a')$ the set of reference frames of a' in the cache, with $RAP(a') = RAP_q(a') \cup RAP_c(a')$, with $B(v)$ the frame indicated by the vertex $v \in V$ and, with $R(v)$ the receiver indicated by the vertex $v \in V$.
For each $a \in RAP(B(v)), \forall v \in V$, we compute the following quantities:

1. The $score(a)$ is equal to the number of users have in their cache the frame a plus the number of users have requested the frame a. This quantity is also computed for $B(v)$;

2. The $w(a)$ is equal to the number of sets RAP which contain a divided by the total number of the reference frames, in detail:

$$\frac{\sum_{B(v),\forall v \in V} isInSet(a, RAP(B(v)))}{\sum_{B(v),\forall v \in V} |RAP(B(v))|}.$$

   where:

$$isInSet(a, RAP(B(v))) = \begin{cases} 1, & \text{if } a \in RAP(B(v)) \\ 0, & \text{otherwise} \end{cases} \tag{7.1}$$

3. The $dist(B(v), a)$ is the size in bits of the refinement;

4. The $rank(a)$ is the total number of vertices indicating a, in detail:

$$rank(a) = \sum_{B(v),\forall v \in V} isInGraph(a, B(v))$$

where:

$$isInGraph(a, B(v)) = \begin{cases} 1, & \text{if } a = B(v) \\ 0, & \text{otherwise} \end{cases} \tag{7.2}$$

We want to find for each $B(v), \forall v \in V$, the best reference frame a in terms of hight *score*, low *weighted distance* and *hight rank*. We find the best reference frame a_q in the demand and the best reference frame a_c in the cache and we first look at the demand and we replace $B(v)$ with a_q if $score(a_q) \geq score(B(v))$ otherwise, we look at the cache and we replace $B(v)$ with a_c if $score(a_c) > score(B(v))$. When a replacement has been done the *score* and the *rank* will be updated. The algorithm works in two steps the first one is the following:
For each $v \in V$, it computes the following sets:

1. $RAP_{q_{mr}} = \{argsmax_{a \in RAP_q(B(v))}(rank(a))\}$;
2. $RAP_{c_{mr}} = \{argsmax_{a \in RAP_c(B(v))}(rank(a))\}$;
3. $RAP_{q_{ms}} = \{argsmax_{a \in RAP_{q_{mr}}}(score(a))\}$;
4. $RAP_{c_{ms}} = \{argsmax_{a \in RAP_{c_{mr}}}(score(a))\}$;
5. $a_q = \{argmin_{a \in RAP_{q_{ms}}}(dist(B(v), a) \cdot w(a))\}$;
6. $a_c = \{argmin_{a \in RAP_{c_{ms}}}(dist(B(v), a) \cdot w(a))\}$;

In the second step the algorithm checks the following conditions: if $score(a_q) \geq score(B(v))$ then it does the replacement and updates $score(a_q)$, $score(B(v))$, $rank(a_q)$, otherwise, if $score(a_c) > score(B(v))$ then it does the replacement and updates $score(a_c)$, $score(B(v))$, $rank(a_c)$.
The running time of the designed algorithm is $O(|V|mRap|n|)$, where:

- n is the total number of the receivers;
- $mRap = max_{B(v), \forall v \in V}(|RAP(B(v))|)$.

Algorithm 11 Replacement Algorithm

for each $v \in V$ **do**
  $RAP_{q_{mr}} = \{argsmax_{a \in RAP_q(B(v))}(rank(a))\}$;
  $RAP_{c_{mr}} = \{argsmax_{a \in RAP_c(B(v))}(rank(a))\}$;
  $RAP_{q_{ms}} = \{argsmax_{a \in RAP_{q_{mr}}}(score(a))\}$;
  $RAP_{c_{ms}} = \{argsmax_{a \in RAP_{c_{mr}}}(score(a))\}$;
  $a_q = \{argmin_{a \in RAP_{q_{ms}}}(dist(B(v), a) \cdot w(a))\}$;
  $a_c = \{argmin_{a \in RAP_{c_{ms}}}(dist(B(v), a) \cdot w(a))\}$;
  if $(score(a_q) \geq score(B(v)))$ **then**
    $v_{old} = B(v)$;
    $B(v) = a_q$;
    $score(a_q) + +$;
    $score(v_{old}) - -$;
    $rank(a_q) + +$;
  else
    if $(score(a_c) > score(B(v)))$ **then**
      $v_{old} = B(v)$;
      $B(v) = a_c$;
      $score(a_c) + +$;
      $score(v_{old}) - -$;
      $rank(a_c) + +$;
    end if
  end if
end for

7.2.2 LFU algorithm for correlation among the frames (LFU-CRR)

For simulations purpose we implemented a modified version of the *LFU* algorithm in order to use the frames correlation, we get this choice because we want to measure the performance of our scheme in a true and in a correct way. The *LFUCRR* algorithm is simple and it's based on the video compression scheme (see Subsection 7.1.2), in detail, let L the number of requests per user, Q_i the set of the requests of user i, $ni = 1$ the number of I frames, np the number of P frames, then we divide, for each user, the set Q_i in two sets: $\hat{I}_i$ which is the set of I frames requested by the user i and $\hat{P}_i$ which is the set of P frames requested by the user i, this two sets are defined for each user in the following way:

- $\hat{I}_i = \{q_{i_1}, q_{i_{1+(np+ni)}}, q_{i_{1+2*(np+ni)}} \ldots, q_{i_{n-np}}\} \Rightarrow |\hat{I}_i| = \frac{L}{np+ni}$;
- $\hat{P}_i = Q \setminus \hat{I}_i \Rightarrow |\hat{P}_i| = L - \frac{L}{np+ni}$.

For each $q_{\hat{I}_k} \in \hat{I}_i$ and $q_{\hat{P}_j} \in \hat{P}_i$, let $H(q_{\hat{I}_k})$ represent the bit rate of coding frame $q_{\hat{I}_k}$ without any reference, and $H(q_{\hat{P}_j}|q_{\hat{I}_k})$ the bit rate of coding $q_{\hat{P}_j}$ using $q_{\hat{I}_k}$ as reference, then the algorithm, for each user, in the first step builds the sets $\hat{I}_i$ and $\hat{P}_i$, and in the second step, for each $q_{\hat{I}_k} \in \hat{I}_i$, it sends $H(q_{\hat{I}_k})$ and $H(q_{\tilde{P}_j}|q_{\hat{I}_k})$, where $q_{\tilde{P}_j} \in \tilde{P}$ and $\tilde{P} = \{q_{\hat{P}_j} \in \hat{P}_i | j = k+1, k+2, \ldots, k+np\}$. In this way the sender will send $\sum_i |\hat{I}_i|$ frames and $\sum_i |\hat{P}_i|$ refinements.

Algorithm 12 LFU algorithm with frames correlation

```
for each user i do
  build Î_i and P̂_i;
  for each q_{Î_k} ∈ Î_i do
    send H(q_{Î_k});
    build P̃;
    for each q_{P̃_j} ∈ P̃ do
      send H(q_{P̃_j}|q_{Î_k});
    end for
  end for
end for
```

7.3 Simulations and Discussion

In this section, we numerically analyze the performance of the scheme illustrated in Section 7.2.1, specifically, assuming the distributed random popularity-based caching policy [54], we compare the average performance of *HGLC* with and without our scheme *REPA*.

For comparison, we plot:

- The average ratio between the total number of bytes sent with the classical coloring, and the total number of bytes sent with the classical coloring with *REPA*;
- The average ratio between the total number of bytes sent with the *LFUCRR*, and the total number of bytes sent with the classical coloring with *REPA*;
- The average ratio between the total number of bytes sent with the *LFU*, and the total number of bytes sent with the classical coloring;

For simplicity and to illustrate the effectiveness of *REPA*, we consider a homogeneous network scenario, in which users request frames according to a Zipf popularity distribution with parameter $\gamma = 0.2$ and all caches have size M frames. Fig. 7.2 plot the average ratio of bytes sent for a network with $n = 4$ users, $L = 26$ requests per user (each user requests L consecutive frames), $m = 1040$ frames, $M = \{0, 128, 256, 512\}$ memory, Zipf parameter $\gamma = 0.2$, and $Q \cap C = \emptyset$ where Q is the demand set and C is the cache set and, $q_i \neq q_k, \forall q_i, q_k \in Q$.

In Fig. 7.2, you can see that: 1) Since $Q \cap C = \emptyset$ and in Q there are no repetitions the ratio between *LFU* and *HGLC* without our scheme is equal to 1 because the *Conflict Graph* is a complete graph; 2) The *HGLC* with our scheme shows a very relevant gain which grows when the memory increases, of course the performance of the *LFUCRR* algorithm is better then *HGLC* without our scheme because it takes advantage of frames correlation but still the *HGLC* with our scheme preserve significant gain.

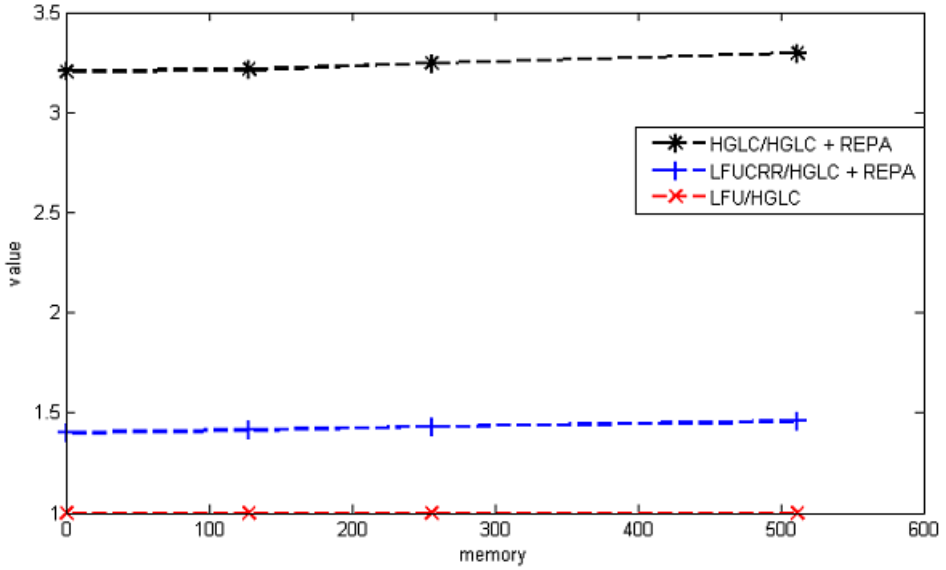


Figure 7.2: Average ratio of bytes sent for a shared multicast link.

7.4 Conclusions

In this Chapter, we have shown how exploiting the correlation among the library frames can result in more efficient content delivery over cache-aided networks. We proposed a correlation-aware caching scheme in which receivers store frames based on their popularity in the caching phase, and receive the coded representations of the requested frames saving a byte transfer by exploiting correlations between the library frames during the delivery phase. The proposed scheme is shown to significantly outperform state of the art approaches that treat library frames as mutually independent. Ongoing and future work entail investigating how to improve our scheme in order to get more important gain when the memory increases because now when the memory grows we get a negligible gain.

Part II

Object Recognition Problem

Chapter 8

Introduction

When the aim is to design and implement a framework for the recognition of some of the components of a natural image, simply applying classification is not a solution as natural images classifiers only based on information extracted from the images, can be, in the most general case, error prone. Our aim is to improve the performance of a natural images classifier introducing in the loop knowledge coming from the real world, expressed in terms of probability of a set of spatial relations between the objects in the images. Not only the probabilistic ontology can be made available for the considered domain: it could also be built or enriched by using entities and relations extracted from a document related to the image. For example, the picture could have been extracted from a technical report or a book, where the text gives information which are related to the considered images. We wish to stress the fact that we are not thinking of a text directly commenting or describing the image, but of a text which is completed and illustrated by the image. In this case, both the classes of objects which can appear in the image and the relations connecting them could be mentioned in the text and could therefore be automatically extracted [5]. A probability can then be associated with them on the basis of the reliability of the extraction or the frequency of the item in the text. The framework presented in this part aims at integrating the output of standard classifiers on different image parts with some domain knowledge, encoded in a probabilistic ontology. In fact, while standard ontologies are quite widespread as a means to manage a-priori information, they fail in the important task of dealing with real world uncertainty. Probabilistic ontologies aim at filling

this gap by associating probabilities to the coded information, and provide then an adequate solution to the issue of coding the context information necessary to correctly understand the content of an image. Such information is then combined with the classifier output in order to correct possible classification errors on the basis of surrounding objects.

The system we are considering, the logical scheme of which is depicted in Figure 9.1, and better detailed in Section 9.1, aims at determining a set of keywords describing the content of an image and the relations existing among them. The idea is to design a system that, starting from an image, will first hypothesise the presence of some objects in the scene through a battery of image based classifiers. Considering for example the image of a building close to a water pool with some boats, it is likely that a classifier might label the reflection of the building on the water beneath the boats as a building, that is a wrong classification. We advocate that such a mis-classification can be corrected introducing the spatial relation between the boat-segment and reflected building, and the external knowledge that an image segment beneath a boat and surrounded by water is more likely to be water than a building. This world knowledge, that we plan to formalise in a probabilistic ontology [20], together with the output of the classifier, will be fed to a probabilistic model [10], in order to improve the performance of the single classifiers.

The classes associated with each segment combined by the spatial relations which can be directly extracted from an analysis of the image are eventually organized in a schematic description of its content. Relations could be further specialized by better specifing the reciprocal position of the segments. For example, the fact that a segment is in the middle, or in the upper right part of the picture, and so on.

The framework presents two main aspects of novelty. First, the use of a probabilistic ontology for a computer vision problem has, at the best of our knowledge, never been proposed before. A second element of novelty is the integration of a probabilistic model with a probabilistic ontology. A preliminary description of the general idea of the approach has been sketched in [1] in a very concise way. In Chapter 9, we discuss all details and a first preliminary experimental assessment.

8.1 Related work

Due to the large amount of images available on the web, for answering to textual image queries, it will be very helpful being able to automatically describe the content of an image. However such a task is not easy at all for a machine, as it requires a visual understanding of the scene, that is almost each object in the image must be recognized, how the objects relate to each other in the scene, and in what they are involved must be understood [85]. This task is tackled in two different ways. The most classical one [35, 61, 23, 34] tries to solve the single sub-problems separately and combines the solutions to obtain a description of an image. A different approach [85, 18, 59] proposes a framework that incorporates all the sub-problems in a single joint model. A method trying to merge the two main approaches has been proposed recently in [88] using a semantic attention model. The problem is, however, very far from being solved. In the context of textual image queries, it can be enough to extract from the images a less complex description (image annotation [86]), such as a list of entities represented in the image, and information about their position and mutual spatial relation in the image. The work proposed in this thesis addresses this task, that is also, as mentioned above, a necessary sub-task of the more general problem of generating a description in natural language. The use of ontologies in the context of image recognition is not new [81]. For instance, in [73] it is proposed a framework for an ontology based image retrieval for natural images, where a domain ontology was developed to model qualitative semantic image descriptions. An ontology of spatial relations, in order to guide image interpretation and the recognition of the structures it contains was proposed in [47]. In [70], low-level features describing the color, position, size and shape of segmented regions are extracted and automatically mapped to descriptors forming a simple vocabulary termed object ontology. At the best of our knowledge, a probabilistic ontology has never been used for the task of image recognition and annotation. On the other hand spatial relations have been used for image recognition in the past, for instance in the context of face recognition [80] or medical image analysis [14], and it has been already shown [6] that the use of spatial relations can decrease the response time and error rate, and that the presence of objects that have a unique interpretation improves the identification

of ambiguous objects in the scene. In the same way the use of probabilistic models is not new in computer vision, in particular a probabilistic model combining the statistics of local appearance and position of objects was proposed already in [75] for the task of face recognition, and in [74] in an image retrieval task, showing that adding a probabilistic model in the loop would improve the recognition rate. In [90] it is proposed a probabilistic semantic model in which the visual features and the textual words are connected via a hidden layer. More recently in the context of 3D object recognition, a system that builds a probabilistic model for each object based on the distribution of its views was proposed in [87]. In [89] a weakly supervised segmentation model learning the semantic associations between sets of spatially neighboring pixels, that is the probability of these sets to share the same semantic label. Finally [33], in the context of action recognition, presents a generative model that allows for characterizing joint distributions of regions of interest, local image features, and human actions. In our case, in order to recognize the objects in a image we tried to design an heuristic algorithm but when we tested it we got either, a bad results and a running-time problems, for this reason we have chosen to treat it with machine learning methods therefore we present a system aims at determining a set of keywords describing the content of an image and the relationships existing among them.

Chapter 9

Exploiting context information for image description

Integrating ontological knowledge is a promising research direction to improve automatic image description. In particular, when probabilistic ontologies are available, the corresponding probabilities could be combined with the probabilities produced by a multi-class classifier applied to different parts in an image. This combination not only provides the relations existing between the different segments, but can also improve the classification accuracy. In fact, the context often gives cues suggesting the correct class of the segment. This Chapter discusses a possible implementation of this integration, and the first experimental results shows its effectiveness when the classifier accuracy is relatively low. For the assessment of the performance we constructed a simulated classifier which allows the a priori decision of its performance with a sufficient precision. The Chapter is organized as follows. Section 9.1 is devoted to the description of the different modules of the system, with a few details about the probabilistic ontology (Section 9.1.1), and to the model adopted to combine classification and ontology probabilities (Section 9.1.2). Experimental assessment is considered in Section 9.2. Some conclusions and proposals for extensions of the presented work conclude the Chapter.

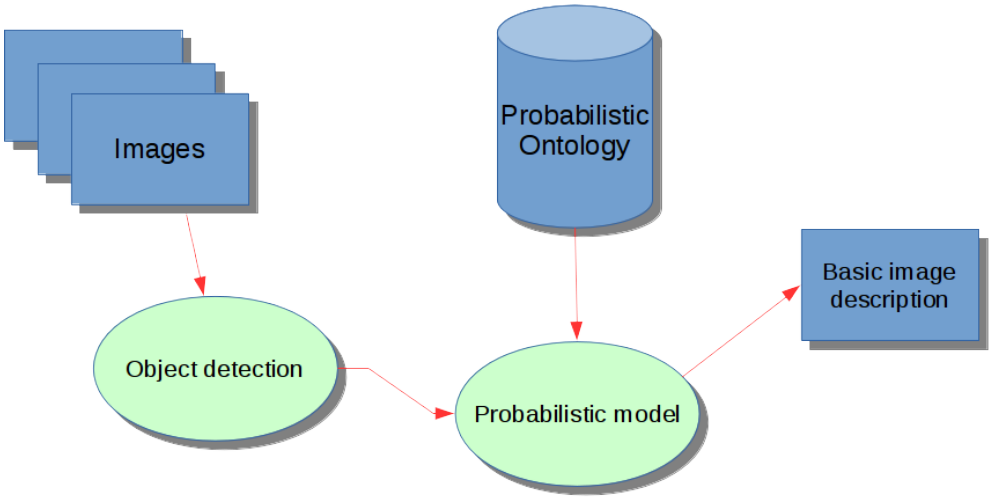


Figure 9.1: Scheme of the proposed framework.

9.1 System Architecture

The proposed framework, depicted in Figure 9.1, is a chain of several logical modules, each corresponding to an element of a computational pipeline. The first step is a classifier, or a set of classifiers, detecting a predefined set of interesting objects in the image, identifying then a set of segments of interest in the image.

The hypotheses formulated for each segment in the image by a statistical classifier are then fed to a probabilistic model, that has been trained off-line. The task of this module is to validate, or correct, the hypothesis formulated in the previous step, integrating the output of the classifier with the world knowledge given by a probabilistic ontology, and expressed in terms of probability of a spatial relationship between instances of two classes of image objects. The class associated with each segment, together with the relations existing between segment pairs, constitute the image description output by the system.

9.1.1 Probabilistic Ontology

This section discusses the construction of a fragment of Probabilistic Ontology (PO) providing the information needed by our system. We need such fragment for the experimental assessment.

Class	**# of items**
sink	371
chair	3,604
table	558
computer/monitor	256+417=673
bed	407
flower	1,822
total	7,435

Table 9.1: Data set statistics.

The main drawback of ontologies when facing real world problems is related to their inability to cope with uncertain information. Due to this, in the last years much work has been devoted to the design of effective tools to attach probabilities to the information contained in ontologies, among whose the most important is probably PrOWL [19]. From the so obtained POs, it is therefore possible to obtain a priori knowledge for applications effective also in complex contexts.

As a consequence, the research area concerning POs is very active and we expect that a number of POs in different domains will be available soon. However, we need a PO in the domain of the image data set we will adopt to assess the system performance, before we can start experimentation. We therefore design and implement an ontology to use in the experiments. In particular, the schema of the ontology will contain the classes to associate to the segments and the spatial relations among them considered in our analysis. On the other hand, probabilities are estimated from the training set after segments are automatically classified and spatial relations are constructed between segment pairs. In particular, we estimate the probability that two classes are in a given relation by the frequency of such event in the data set. More precisely denoting with D a set of segments used to compute the probabilities, with $R = \{r_1, \ldots, r_i\}$ the set of types of relation, with C the set of segments classes, we compute the probability that $c_1 \in C$ is in relation $r \in R$ with $c_2 \in C$ as:

$$\Pr(r, c_1, c_2) = \frac{D_r(c_1, c_2)}{\sum_{c_x \in C, c_y \in C} D_r(c_x, c_y)} \tag{9.1}$$

where $D_r(c_x, c_y)$ is the number of times that pairs of segments in D of classes respectively c_1 and c_2 satisfy the relation r. In general, as the relations are not necessarily symmetric, we have $\Pr(r, c_1, c_2) \neq \Pr(r, c_2, c_1)$.

Since there are no tools for directly constructing a PO, we use Protégé [1] for the construction of the schema of the ontology, while we use Pronto [60] as a reasoner for POs, as it adopts the standard OWL 1.1. The import of the schema developed by Protégé into Pronto is performed by editing the corresponding XML files and adding the probabilities. An example is given in Figure 9.2, where the element tagged `pronto:certainty` is added to the axiom prepared by Protégé. Although Pronto accepts probability ranges, as we use simple values, the two extremes of the interval coincides (0.070990; 0.070990 in the example).

```
<owl11:Axiom>
        <rdf:subject rdf:resource="URI#x" />
        <rdf:predicate rdf:resource="&rdfs;subClassOf" />
        <rdf:object rdf:resource="URI#y" />
        <pronto:certainty>0.070990;0.070990</pronto:certainty>
</owl11:Axiom>
```

Figure 9.2: Piece of the XML of the PO corresponding to an axiom with an associated probability.

9.1.2 Combination Models

This section investigates which model to use to integrate the classifiers and the ontological knowledge. In the task we are considering the role of POs requires providing probabilities describing the domain of interest, to be integrated with the ones associated by the classifier to each class for each input segment. The main goal of our system is the classification of the segments in the input image. We aim to exploit the relations between pairs of segments to improve this classification. More formally, every image contains a set of segments S and there are a number of possible relations R connecting segment pairs, i.e. $R = \{isClose, intersects, isInside\}$. The first two relations are symmetrical, while the last one is asymmetrical: details

[1] Freely available from `http://protege.stanford.edu/`.

about this implementation in the assessment description. For each segment in the image, the classifier associates a probability distribution to the set of all possible classes C. When we consider only the classification step, we classify the segment with the most probable class: this represents our baseline, as it only considers the classifier output, without any information coming from the PO. However, we can see the output of the classifier for each segment s in the image as a random variable $c(s)$ with values in C. In the following we discuss how such random variable is integrated with the ontological probabilities.

In fact, the ontology produces, for every pair of classes $c_1, c_2 \in C$ and every possible relation $r \in R$, the probability $\Pr(r, c_1, c_2)$ that in the real world two segments of classes c_1 and c_2 respectively are in relation r: its expression is given in Equation 9.1. By integrating this information with the probabilities computed by the classifier, the classification performance could improve. Moreover, the solution output by this integration is likely to be consistent with the ontological knowledge, which can be an important feature in systems where the post-processing requires a set of properties on the considered candidates. In fact, whenever a relation can not hold between two classes, the corresponding ontological probability is null, and this also lowers the probability of the corresponding couple of classes.

We associate the following log-linear probability to the two classes associated with each context $x = (s_1, s_2, r : r(s_1, s_2))$ built around the relation type r connecting segments s_1 and s_2:

$$\Pr(c_1, c_2|x) = \frac{e^{v_{c_1} f_C(s_1,c_1)+v_{c_2} f_C(s_2,c_2)+v_{r,c_1,c_2} f_{PO}(r(s_1,s_2),c_1,c_2)}}{Z_{x,c_1,c_2}} \tag{9.2}$$

where $f_C(s, c) = \Pr(c(s) = c)$ and $f_{PO}(r, c_1, c_2) = \Pr(r(c_1, c_2))$, while Z_{x,c_1,c_2} is a normalisation factor depending on x and on the classes assigned to the two segments. Note that the features $f_C(\cdot)$ are produced by the classifier, while $f_{PO}(\cdot)$ depends on the probabilistic ontology. In conclusion, we consider two families of parameters: *class parameters* v_c for each class c and *relation parameters* v_{r,c_1,c_2} for each type of relation r and pair of classes (c_1, c_2). All in all, there are $|C|$ class parameters and $|R||C|^2$ *relation parameters*.

The parameters are estimated during the training, which maximises the likelihood of the training set. For this optimisation, we use the *Toolkit for Advanced*

Optimisation (TAO) library, which implements a variety of optimisation algorithms for several classes of problems (unconstrained, bound-constrained, and PDE-constrained minimisation, nonlinear least-squares, and complementarity). In our work we focus on unconstrained minimisation methods which are used to minimise a function of many variables without any constraints on the variables. The method that we have used is *Limited Memory Variable Metric*, it is a *quasi-Newton* optimisation solver and it solves the Newton step using an approximation factor which is composed using the *BFGS* update formula.

Once we have estimated all the parameters $V = \{v_c, v_{r,c_i,c_j}\}$ with $c, c_i, c_j \in C$ and $r \in R$, we aim to assign the correct class to each segment in the input image. To do so, we consider two different models: in the former, to which we refer as M1, we assign to the classes in a given context a score which is equal to the $\Pr(c_1, c_2|x)$ as given by Equation 9.2, while in the latter, M2, the score is given by its logarithm. In fact, when adopting, as in our case, a log-linear expression, only considering exponents is much more efficient than directly summing probabilities. We therefore obtain the following expressions for the scores sc_1 and sc_2 respectively corresponding to $M1$ and $M2$.

$$sc_1(c_1, c_2|x) = \Pr(c_1, c_2|x) = \frac{e^{v_{c_1} f_C(s_1,c_1)+v_{c_2} f_C(s_2,c_2)+v_{r,c_1,c_2} f_{PO}(r(s_1,s_2),c_1,c_2)}}{Z_{x,c_1,c_2}} \tag{9.3}$$

$$\begin{aligned} sc_2(c_1, c_2|x) = \log \Pr(c_1, c_2|x) &= v_{c_1} f_C(s_1, c_1) + v_{c_2} f_C(s_2, c_2) + \\ &+ v_{r,c_1,c_2} f_{PO}(r(s_1, s_2), c_1, c_2) - \log Z_{x,c_1,c_2} \end{aligned} \tag{9.4}$$

For each context x, we then compute the score that a given class c is associated with one segment, by summing the scores that every class is associated with each segment and that the relation assumes any of all possible relation types. We then associate to the first segment the class which maximises such a score in all segment pairs including it:

$$SC(c|s) = \max_{s_2:\exists r, r(s_1,s_2)} \sum_{c_2 \in C} \sum_{r \in R} sc(c, c_2|(s_1, s_2, r : r(s_1, s_2)). \tag{9.5}$$

In this expression, sc stays for sc_1 or sc_2 depending on the adopted model. Note that since all relation types we consider are symmetrical, for every context $x =$

$(s_1, s_2, r : r(s_1, s_2))$ also the symmetrical one $x' = (s_2, s_1, r(s_2, s_1))$ is defined, and therefore we can express the score as considering only the first of the two cases. However, when asymmetrical relations are also considered, the expressions can be easily generalised.

Finally, we assign to each segment the class which maximises the score of the class given the segment:

$$c^*(s) = \arg\max_{c \in C} SC(c|s) \tag{9.6}$$

To complete the textual description, the relations existing between segment pairs and used for determining the contexts defined above are added.

9.2 Experimental Assessment

This section describes and discusses the quantitative assessment of the performance of the proposed approach.

9.2.1 Experimental Protocol

The system performance is evaluated in terms of classification accuracy, i.e. the rate of segments which have been correctly classified. In particular, we considered six classes obtained by clustering the data set ones and then taking the six with a larger number of items: the adopted classes and the number of times they occur in the data set are reported in Table 9.1. Furthermore, we considered three relation types corresponding to the relative position of two segments in an image: *near*, *very near* and *intersecting*. Clearly, all three the relations are symmetrical.

The role of the classifier in our system is to produce a probability distribution on the set of classes for every input segment. The literature on object recognition is very rich [79]. The risk in choosing one approach or the other is that the final results would depend on this choice and its influence can not be distinguished by the one of the combination model. We therefore decided to substitute the actual classification with a random simulation able to produce any given performance. In this way, it is possible to describe the dependence of the system performance on the classification accuracy. All in all, we therefore need a method to simulate the behaviour of a multi-class classifier with an assigned accuracy a.

For this goal, we use the strategy described by the pseudo-code in Alg. 13. Given a segment, we randomly choose a score in $[0, 1]$ by the function $U(0, 1)$ for each class in the class set C. We then assign, with a probability given by the desired accuracy a, the maximum score to the gold class, while the other scores are randomly assigned to the remaining classes. The scores are finally normalised to obtain a probability distribution. As the classifier assigns to each segment the maximum probability class, we have that this corresponds to the right choice in the a percentage of cases, resulting in the desired accuracy. The use of a simulated classifier is not novel (see, for instance, [91]).

Algorithm 13 Pseudo-code of the simulated classifier.

```
maxClassProb←0.0;
BestClass← ∅;
for CurrentClass ∈ ClassSet do
    NewClassProb ∼ U(0, 1);
    ClassProb[CurrentClass]← newClassProb;
    if ClassProb[CurrentClass] > MaxClassProb then
        MaxClassProbValue← ClassProb[CurrentClass];
        BestClass ← CurrentClass;
    else
        if TossingACoin == Head then
            RandomClass ← CurrentClass;
        end if
    end if
end for
Accuracy ∼ U(0, 1);
Gold ← GoldClass(Segment);
if Accuracy < DesiredAccuracy then
    Swap(ClassProb[BestClass],ClassProb[Gold]);
else
    swap(ClassProb[RandomClass],ClassProb[Gold]);
end if
normalize(ClassProb);
```

As we aim to assess the improvement we can obtain by introducing the ontological knowledge, we compare the system performance with a baseline consisting in the (simulated) classifier alone. The two approaches discussed in Section 9.1.2 are applied to combine the PO into the system: M1 and M2.

9.2.2 Dataset Used

Since the main goal of the project is to assess the goodness of the probabilistic model and probabilistic ontology proposed, and due to the fact that image segmentation is a problem that is very far from being solved, especially for complex

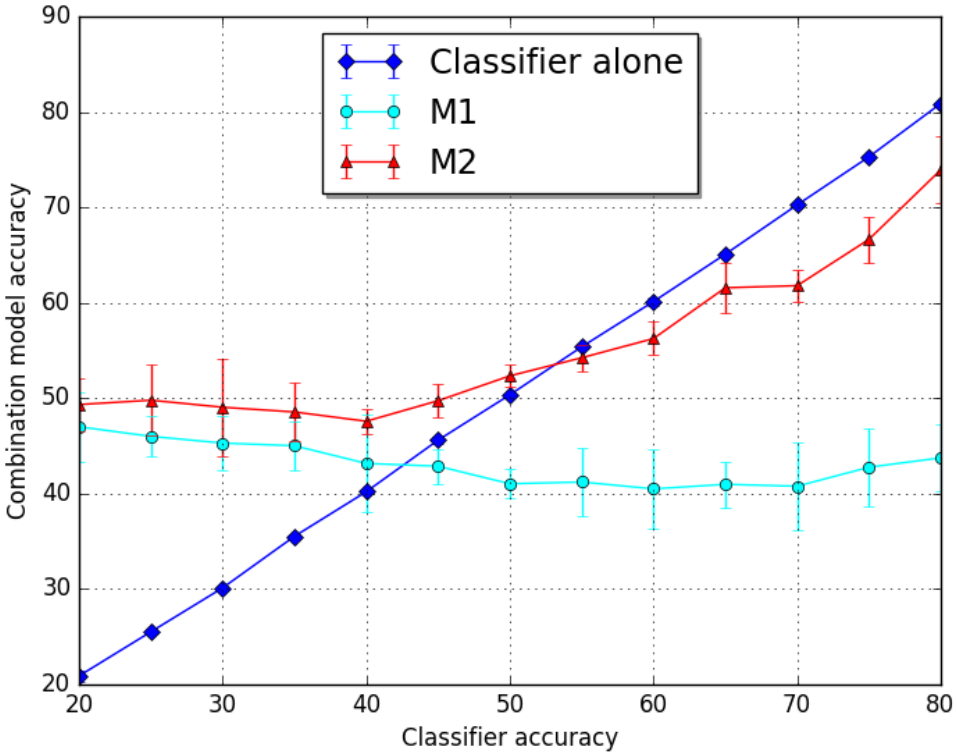


Figure 9.3: Performance of the two systems compared with the baseline. Error bars give the 95% confidence intervals.

natural images, we prefer to avoid any uncertainty introduced by the segmentation algorithm by skipping the implementation of this step and work on the available 1,700 manually segmented images in the *MIT-Indoor Data-set*. These pictures are taken in indoor surroundings, including kitchens, bedrooms, libraries, gyms and so on. Whenever an actual system based on the proposed approach is implemented, the best available solution for the segmentation will be included. We randomly divided the data in three parts: two of them, containing each the 30% of the data, are used to train the PO and the combination model respectively, while the remaining 40% of the data are used to assess the system performance. Note that in our view it is important that the data used to train the PO and the combination models are different, as in actual domains they usually have different origins.

9.2.3 Results and Discussion

The system accuracy of the approaches proposed in this Chapter are depicted in Figure 9.3 and compared with the accuracy of the statistical classifier applied alone.

For the sake of completeness, we considered a very wide range of accuracies for the simulated classifier: from 20% up to 80%, even if in actual conditions, the values of classifiers accuracy is more likely under $50 - 60\%$. However, in any case, we see that the M2 outperforms the M1, whose performance even deteriorates when the classifier accuracy improves. A possible explication for this behavior could be that too much confidence is given to the a priori PO score with respect to the actual input data evidence.

On the other hand, the M2 improves on the simple classifier when the latter performance are inferior than about 55%, that is in realistic experimental conditions. We can observe how performance of this model are much better than the classifier alone when the latter performance are worse than 30%, and this can be the case when the task is not too easy. Even for classifiers obtaining an accuracy between 30% and 55%, the adoption of an approach integrating PO knowledge is advantageous.

Last, but not least, we observe that even when M2 performs worse than the classifier alone, its accuracy improves with the classifier accuracy, so that the two curves are approximately parallel. This could suggest that a better ontology design, resulting in a better PO, could help the system to overcome the performance obtained by the classifier alone.

9.3 Conclusions

In this Chapter, we proposed and experimentally evaluated two different probabilistic models to integrate the probabilities derived from a probabilistic ontology with the ones produced by a statistical classifier. One of the two proved to perform in an acceptable way and could be used in an actual system.

For the sake of obtaining a clear view of the integration module performance, we tried to minimize the effect of the other modules. Therefore, we started from

images which had been manually segmented and simulated a classifier in such a way that its accuracy could be controlled. As a future work, we plan to assess the performance of the proposed approach when coupled with state-of-the-art classification and segmentation modules.

A fragment of a probabilistic ontology has been built by using three relations which could be automatically recognized in the input images, while the corresponding probabilities have been estimated from their frequencies. When more sophisticated ontologies will be available, containing information from large data sets, we expect the integration to give even better results.

Appendix A

Graph Coloring Problem

The *Graph Coloring* is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color. The goal is coloring all vertices of the graph with the minimum numbers of colors. A possible model for *Graph Coloring* reads:

$$min \sum_{h=1}^{n} y_h \qquad \text{(Objective function)}$$

$$s.t.$$

$$\sum_{h=1}^{n} x_{ih} = 1, \forall i \in V \qquad \text{(A.1)}$$

$$x_{ih} + x_{jh} \leq y_h, \forall (i,j) \in E, h = 1, ..., n \qquad \text{(A.2)}$$

$$x_{ih} \in \{0,1\}, \forall i \in V, h = 1, ..., n \qquad \text{(A.3)}$$

$$y_h \in \{0,1\}, h = 1, ..., n \qquad \text{(A.4)}$$

where $x_{ih} = 1$ if and only if at node i has been associated the color h, $y_h = 1$ if and only if the color h has been associated with the node $j \in V$. Constraint A.1 require that each vertex is colored, and constraint A.2 impose that at most one of a pair of adjacent vertices receive a color h, when the color is used.

Appendix B

Proof of Theorem 1

In this section, our goal is to prove the the right-hand side of (4.1) which is an upper bound of the average chromatic number of the random conflict graph $\mathsf{H}_{\mathsf{M},\mathsf{W}}$ achieved by GCC. To this end, in the following, we will compute the average achievable rate achieved by GCC_1 and GCC_2 respectively. Recall that the expectation is taken over the demands distribution. Note that $\mathbb{E}[\chi(\mathsf{H}_{\mathsf{M},\mathsf{W}})]$ is a random variable of M. Note that GCC_1 considers all the vertices in a realization $\mathcal{H}_{\mathbf{M},\mathbf{W}}$ as distinct objects although they may represent the same packets. On the other hand, GCC_2 tries to use the fact that different vertices may represent the same packet (naive multicasting). We will show that GCC_1 and GCC_2 achieve rate of $\psi(\mathbf{P},\mathbf{Q})$ and $\bar{m}$ with high probability respectively.

B.0.1 Performance of GCC_1

To compute the performance of GCC_1, we first see that for all $v \in \mathcal{I}$ obtained in this algorithm, $\mathcal{K}_v$ are identical. Given the requests vector $\mathbf{f}$, we denote the set $\mathcal{I}$ (see Fig. 2) whose elements have a particular $\mathcal{K}_v, \forall v \in \mathcal{I}$ consisting of a subset of users $\mathcal{U}^\ell \subset \mathcal{U}$ with cardinality ℓ as $\mathcal{I}(\mathcal{U}^\ell, \mathbf{f}, i)$, where i ranges from 1 to the number of independent sets corresponding to $\mathcal{U}^\ell$. Given $\mathcal{U}^\ell$, let $\mathcal{J}(\mathcal{U}^\ell, \mathbf{f}) = \{\mathcal{I}(\mathcal{U}^\ell, \mathbf{f}, i) : \forall i\}$.

We compute the average number of distinct colors obtained by GCC_1 in the following. By fixing a demands realization $\mathbf{f}$, for each $\mathcal{U}^\ell$, we compute the number of used colors $|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})|$ by using GCC_1. Given $\mathbf{f}$, we can see that $|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})|$ is a random variable which is a function of M. Let the indicator $1\{\mathcal{K}_{v_{f_u}} = \mathcal{U}^\ell\}$ denote the event that vertex v_{f_u} from file f_u requested by user $u \in \mathcal{U}^\ell$ is available in

all the users in $\mathcal{U}^\ell$ but u and the rest of the vertices $\mathcal{U} \setminus \mathcal{U}^\ell$, then $1\{\mathcal{K}_{v_{f_u}} = \mathcal{U}^\ell\}$ follows a Bernoulli distribution with parameter $(p_f M_u)^{\ell-1}(1-p_f M_u)^{n-\ell+1}$ such that its expectation is $(p_f M_u)^{\ell-1}(1-p_f M_u)^{n-\ell+1}B$. Then, we can see that given f, $\sum_{\forall v_{f_u}} 1\{\mathcal{K}_{v_{f_u}} = \mathcal{U}^\ell\} = (p_f M_u)^{\ell-1}(1-p_f M_u)^{n-\ell+1}B + o(B)$ with high probability [16]. Thus, as $B \to \infty$, we have that with high probability,

$$\begin{aligned}
&|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})| \\
&= \max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} \sum_{\forall v_{f_u}} 1\{\mathcal{K}_{v_{f_u}} = \mathcal{U}^\ell\} \\
&= \max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} \quad (p_{f,u} M_u)^{\ell-1}(1-p_{f,u} M_u)^{n-\ell+1}B \\
&\quad + o(B),
\end{aligned} \tag{B.1}$$

where $\mathbf{f}(\mathcal{U}^\ell)$ represent the set of files requested by $\mathcal{U}^\ell$.

Then, by averaging over the demands distribution, we obtain that with high probability ,

$$\begin{aligned}
&\mathbb{E}[\chi(\mathsf{H}_{\mathsf{M},\mathsf{W}})] \\
&\leq \mathbb{E}\left[\sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \in \mathcal{U}} \left|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})\right|\right] \\
&= \sum_{l=1}^{n} \sum_{\mathcal{U}^\ell \in \mathcal{U}} \mathbb{E}\left[\left|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})\right|\right] \\
&\overset{(a)}{=} \sum_{l=1}^{n} \sum_{\mathcal{U}^\ell \in \mathcal{U}} \mathbb{E}\left[\max_{f_u \in \mathsf{f}(\mathcal{U}^\ell)} \quad (p_{f,u} M_u)^{\ell-1}\right. \\
&\quad \left.(1-p_{f,u} M_u)^{n-\ell+1}B + o(B)\right] \\
&\overset{(b)}{=} \sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \in \mathcal{U}} \sum_{f=1}^{m} \sum_{u \in \mathcal{U}^\ell} \rho_{f,u,\mathcal{U}^\ell} \\
&\quad (1-p_{f,u} M_u)^{n-\ell+1}(p_{f,u} M_u)^{\ell-1} + \delta_1(B),
\end{aligned} \tag{B.2}$$

where (a) is by using (B.1) and (b) is obtained by computing the probability that the requested file f_u in $\mathbf{f}(\mathcal{U}^\ell)$ that maximizes $\big((p_{f,u} M_u)^{l-1}(1-p_{f,u} M_u)^{n-\ell+1}B\big)$. $\delta_1(B)$ denotes a smaller order term of $\sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \in \mathcal{U}} \sum_{f=1}^{m} \sum_{u \in \mathcal{U}^\ell} \rho_{f,u,\mathcal{U}^\ell}(1-p_{f,u} M_u)^{n-\ell+1}(p_{f,u} M_u)^{\ell-1}$. For any $\mathcal{U}^\ell$, we obtain that $\sum_f \sum_{u \in \mathcal{U}^\ell} \rho_{f,u,\mathcal{U}^\ell} = 1$, and $\rho_{f,u,\mathcal{U}^\ell}$ denotes the probability that file f is the file with memory assignment $p_{f,u}$ such that $\rho_{f,u,\mathcal{U}^\ell} \triangleq \mathbb{P}(f =$

$\arg\max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} (p_{f,u}M_u)^{\ell-1}(1-p_{f,u}M_u)^{n-\ell+1})$, where $\mathbf{f}(\mathcal{U}^\ell)$ denotes the set of files requested by a subset users $\mathcal{U}^\ell$. Thus, we normalize (B.2) by B and obtain that with high probability,

$$\begin{aligned}\bar{R}(\mathbf{P},\mathbf{Q}) &= \frac{\mathbb{E}[\chi(\mathsf{H}_{\mathsf{M},\mathsf{W}})]}{B} \\ &\leq \sum_{\ell=1}^{n}\sum_{\mathcal{U}^\ell\in\mathcal{U}}\sum_{f=1}^{m}\sum_{u\in\mathcal{U}^\ell}\rho_{f,u,\mathcal{U}^\ell} \\ &\quad (1-p_{f,u}M_u)^{n-\ell+1}(p_{f,u}M_u)^{\ell-1}, \\ &= \psi(\mathbf{P},\mathbf{Q}), \end{aligned} \tag{B.3}$$

which is the first term inside the minimum in (4.1) and we denote $(\sum_{\ell=1}^{n}\sum_{\mathcal{U}^\ell\in\mathcal{U}}\sum_{f=1}^{m}\sum_{u\in\mathcal{U}^\ell}\rho_{f,u,\mathcal{U}^\ell}(1-p_{f,u}M_u)^{n-\ell+1}(p_{f,u}M_u)^{\ell-1})$ as $\psi(\mathbf{P},\mathbf{Q})$.

B.0.2 Performance of GCC_2

We can see that by using GCC_2 and letting all the users cache the most $\lfloor M_u \rfloor$ popular files, the probability that file f will be transmitted by uncoded multicasting is $(1-\prod_{u=1}^{n}(1-q_{f,u}))$. Hence, $\mathbb{E}[\chi(\mathsf{H}_{\mathsf{M},\mathsf{W}})]$, which can be upper bounded by the average number of distinct requested packets, is given by that with high probability [16],

$$\begin{aligned}\mathbb{E}[\chi(\mathcal{H}_{\mathsf{M},\mathsf{W}}) &\leq \sum_{f=\lfloor\min_u\{M_u\}\rfloor+1}^{m}\left(1-\prod_{u=1}^{n}(1-q_{f,u})\right)B \\ &\quad +\delta_2(B) \\ &= \bar{m}B+\delta_2(B). \end{aligned} \tag{B.4}$$

where $\delta_2(B)$ is a smaller order term of $\bar{m}B$, where $\bar{m} \triangleq \sum_{f=\lfloor\min_u\{M_u\}\rfloor+1}^{m}(1-\prod_{u=1}^{n}(1-q_{f,u}))$.

We normalize (B.4) by B and obtain that with high probability,

$$\bar{R}(\mathbf{P},\mathbf{Q}) \leq \bar{m}, \tag{B.5}$$

which is the second term inside the minimum in (4.1).

Thus, by taking the minimum of (B.3) and (B.5), we obtain Theorem 1.

Bibliography

[1] Andrea Apicella, Anna Corazza, Francesco Isgrò, and Giuseppe Vettigli. Integrating a priori probabilistic knowledge into classification for image description. In *to appear as a shor paper in the proceedings of the 26th IEEE WETICE Conference*, 2017.

[2] F. Arbabjolfaei, B. Bandemer, Y.-H. Kim, E. Sasoglu, and L. Wang. On the capacity region for index coding. *arXiv:1302.1601*, 2013.

[3] M. Ji J. Llorca A. Tulino A.S. Cacciapuoti et al, M Caleffi. Speeding up future video distribution via channel-aware caching-aided coded multicast. *IEEE JSAC*, 34(8):2207–2218, 2016.

[4] C. Avanthay, A. Hertz, and N. Zufferey. A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151:379–388, 2003.

[5] Nguyen Bach and Sameer Badaskar. A review of relation extraction. *Language Technologies Institute, Carnegie Mellon University*, 2007.

[6] M. Bar and S. Ullman. Spatial context in recognition. *Perception*, 25(3):343–352, 1996.

[7] Ziv Bar-Yossef, Yitzhak Birk, TS Jayram, and Tomer Kol. Index coding with side information. *Information Theory, IEEE Transactions on*, 57(3):1479–1494, 2011.

[8] Y. Birk and T. Kol. Informed-source coding-on-demand (iscod) over broadcast channels. IEEE, 1998.

[9] Yitzhak Birk and Tomer Kol. Coding on demand by an informed source (iscod) for efficient broadcast of different supplemental data to caching clients. *IEEE/ACM Transactions on Networking (TON)*, 14(SI):2825–2830, 2006.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[11] Anna Blasiak, Robert D. Kleinberg, and Eyal Lubetzky. Index coding via linear programming. [13].

[12] Anna Blasiak, Robert D. Kleinberg, and Eyal Lubetzky. Index coding via linear programming. [13].

[13] Anna Blasiak, Robert D. Kleinberg, and Eyal Lubetzky. Index coding via linear programming. *CoRR*, abs/1004.1379, 2010.

[14] Isabelle Bloch, Olivier Colliot, Oscar Camara, and Thierry Géraud. Fusion of spatial relationships for guiding recognition, example of brain structure recognition in 3d {MRI}. *Pattern Recognition Letters*, 26(4):449 – 457, 2005.

[15] A.L. Blum. *Algorithms for approximate graph coloring*. PhD thesis, 1991.

[16] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[17] M. Chaudhry, Z. Asad, A. Sprintson, and M. Langberg. On the complementary index coding problem. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 244–248. IEEE, 2011.

[18] X. Chen and C. L. Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2422–2431, June 2015.

[19] Paulo Cesar G. Da Costa. *Bayesian Semantics for the Semantic Web*. PhD thesis, George Mason University, Fairfax, VA, USA, 2005. AAI3179141.

[20] Zhongli Ding and Yun Peng. A probabilistic extension to ontology language owl. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4 - Volume 4*, HICSS '04, pages 40111.1–, 2004.

[21] Frederic Dufaux, Wen Gao, Stefano Tubaro, and Anthony Vetro. Distributed video coding: Trends and perspectives. *J. Image Video Process.*, 2009:10:1–10:13, February 2009.

[22] S. El Rouayheb, A. Sprintson, and C. Georghiades. On the index coding problem and its relation to network coding and matroid theory. *Information Theory, IEEE Transactions on*, 56(7):3187–3195, 2010.

[23] Desmond Elliott and Frank Keller. Image description using visual dependency representations. In *EMNLP*, volume 13, pages 1292–1302, 2013.

[24] A. Osseiran et al. Scenarios for 5g mobile and wireless communications: the vision of the metis project. *IEEE Commun. Mag.*, 52(5):26–35, 2014.

[25] E. Zeydan et al. Big data caching for networking: Moving from cloud to edge. *IEEE Commun. Mag.*, 54(9):36–42, 2016.

[26] J. Llorca et al. Network-coded caching-aided multicast for efficient content delivery. *Proc. IEEE ICC*, pages 3557–3562, 2013.

[27] K. Shanmugam et al. Finite length analysis of caching-aided coded multicasting. *IEEE Trans. Inf. Theory*, 62(10):5524–5537, 2016.

[28] L. S. Cardoso et al. Cortexlab: A facility for testing cognitive radio networks in a reproducible environment. *Proc. EAI CROWNCOM*, pages 503–507, 2014.

[29] M. Ji et al. Order-optimal rate of caching and coded multicasting with random demands. In *preprint arXiv:1502.03124* [30].

[30] M. Ji et al. Order-optimal rate of caching and coded multicasting with random demands. *preprint arXiv:1502.03124*, 2015.

[31] N. Golrezaei et al. Femtocaching: Wireless video content delivery through distributed caching helpers. *Proc. of IEEE INFOCOM*, pages 1107–1115, 2012.

[32] X Wang et al. Cache in the air: exploiting content caching and delivery techniques for 5g systems. *IEEE Commun. Mag.*, 52(2):131–139, 2014.

[33] Abdalrahman Eweiwi, Muhammad Shahzad Cheema, and Christian Bauckhage. Action recognition in still images by learning spatial interest regions from videos. *Pattern Recognition Letters*, 51:8 – 15, 2015.

[34] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1473–1482, June 2015.

[35] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 15–29, 2010.

[36] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[37] P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys on Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[38] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP – Part I: Algorithms. *International Transactions in Operational Research*, 16(1):1–24, 2009.

[39] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP – Part II: Applications. *International Transactions in Operational Research*, 16(2):131–172, 2009.

[40] P. Festa and M.G.C. Resende. GRASP: Basic components and enhancements. *Telecommunication Systems*, 46(3):253–271, 2011.

[41] N. Funabiki and T. Higashino. A minimal-state processing search algorithm for graph coloring problems. *IEICE Transactions Fundamentals*, E83-A(7):1420–1430, 2000.

[42] M. Garey and D. Johnson. The Complexity of Near-Optimal Coloring. *Journal of the ACM*, 23:43–49, 1976.

[43] M. Garey and D. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W.H. Freeman, San Francisco, 1979.

[44] P. Hansen, M. Labbé, and D. Schindl. Set covering and packing formulations of graph coloring: algorithms and first polyhedral results. *Discrete Optimizatio*, 6:135–147, 2009.

[45] P Hassanzadeh, E Erkip, Llorca Jaime, and Antonia Tulino. Correlation-aware distributed caching coded delivery, 2016.

[46] Ishay Haviv and Michael Langberg. On linear index coding for random graphs. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 2231–2235. IEEE, 2012.

[47] Céline Hudelot, Jamal Atif, and Isabelle Bloch. Fuzzy spatial relation ontology for image interpretation. *Fuzzy Sets and Systems*, 159(15):1929 – 1951, 2008.

[48] S. A. Jafar. Topological interference management through index coding. *arXiv:1301.3106*, 2013.

[49] M. Ji, G. Caire, and A.F. Molisch. The throughput-outage tradeoff of wireless one-hop caching networks. [50].

[50] M. Ji, G. Caire, and A.F. Molisch. The throughput-outage tradeoff of wireless one-hop caching networks. *arXiv:1302.2168*, 2013.

[51] M. Ji, A.M. Tulino, J. Llorca, and G. Caire. Order optimal coded caching-aided multicast under zipf demand distributions. *arXiv preprint arXiv:1402.4576*, 2014.

[52] M. Ji, A.M. Tulino, J. Llorca, and G. Caire. Order optimal coded delivery and caching: Multiple groupcast index coding. *arXiv:1402.4572*, 2014.

[53] Mingyue Ji, Giuseppe Caire, and Andreas F. Molisch. Optimal throughput-outage trade-off in wireless one-hop caching networks. *CoRR*, abs/1302.2168, 2013.

[54] Mingyue Ji, Karthikeyan Shanmugam, Giuseppe Vettigli, Jaime Llorca, Antonia Maria Tulino, and Giuseppe Caire. An efficient multiple-groupcast coded multicasting scheme for finite fractional caching. In *2015 IEEE International Conference on Communications, ICC 2015, London, United Kingdom, June 8-12, 2015*, pages 3801–3806, 2015.

[55] Mingyue Ji, Karthikeyan Shanmugam, Giuseppe Vettigli, Jaime Llorca, Antonia Maria Tulino, and Giuseppe Caire. An efficient multiple-groupcast coded multicasting scheme for finite fractional caching. In *2015 IEEE International Conference on Communications, ICC 2015, London, United Kingdom, June 8-12, 2015* [54], pages 3801–3806.

[56] Mingyue Ji, Antonia M Tulino, Jaime Llorca, and Giuseppe Caire. On the average performance of caching and coded multicasting with random demands. In *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on* [57], pages 922–926.

[57] Mingyue Ji, Antonia M Tulino, Jaime Llorca, and Giuseppe Caire. On the average performance of caching and coded multicasting with random demands. In *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*, pages 922–926. IEEE, 2014.

[58] Mingyue Ji, Antonia Maria Tulino, Jaime Llorca, and Giuseppe Caire. Caching and coded multicasting: Multiple groupcast index coding. In *2014*

IEEE Global Conference on Signal and Information Processing, GlobalSIP 2014, Atlanta, GA, USA, December 3-5, 2014, pages 881–885, 2014.

[59] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):664–676, April 2017.

[60] Pavel Klinov and Bijan Parsia. *Uncertainty Reasoning for the Semantic Web II: International Workshops URSW 2008-2010 Held at ISWC and UniDL 2010 Held at FLoC, Revised Selected Papers*, chapter Pronto: A Practical Probabilistic Description Logic Reasoner, pages 59–79. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[61] G. Kulkarni, V. Premraj, S. Dhar, Siming Li, Yejin Choi, A. C. Berg, and T. L. Berg. Baby talk: Understanding and generating simple image descriptions. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1601–1608, 2011.

[62] M. Laguna and R. Martí. A GRASP for coloring sparse graphs. *Computational Optimization and Applications*, 19(2):165–178, 2001.

[63] E. Lubetzky and U. Stav. Nonlinear index coding outperforming the linear optimum. *Information Theory, IEEE Transactions on*, 55(8):3544–3551, 2009.

[64] M. Maddah-Ali and U. Niesen. Decentralized coded caching attains order-optimal memory-rate tradeoff. In *IEEE/ACM Trans. Netw.* [66], pages 1029–1040.

[65] M. Maddah-Ali and U. Niesen. Decentralized coded caching attains order-optimal memory-rate tradeoff. [66], pages 1029–1040.

[66] M. Maddah-Ali and U. Niesen. Decentralized coded caching attains order-optimal memory-rate tradeoff. *IEEE/ACM Trans. Netw.*, 23(4):1029–1040, 2015.

[67] Mohammad Ali Maddah-Ali and Urs Niesen. Fundamental limits of caching. In *Information Theory, IEEE Transactions on* [69], pages 2856–2867.

[68] Mohammad Ali Maddah-Ali and Urs Niesen. Fundamental limits of caching. [69], pages 2856–2867.

[69] Mohammad Ali Maddah-Ali and Urs Niesen. Fundamental limits of caching. *Information Theory, IEEE Transactions on*, 60(5):2856–2867, 2014.

[70] V. Mezaris, I. Kompatsiaris, and M. G. Strintzis. An ontology approach to object-based image retrieval. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 2, pages II–511–14 vol.3, Sept 2003.

[71] U. Niesen and M.A. Maddah-Ali. Coded caching for delay-sensitive content. *Proc. IEEE ICC*, pages 5559–5564, 2015.

[72] K. Ngo S. Yang and M. Kobayashi. Content delivery with coded caching and massive mimo in 5g. *Proc. IEEE ISTC*, pages 370–374, 2016.

[73] Sohail Sarwar, Zia Ul Qayyum, and Saqib Majeed. Ontology based image retrieval framework using qualitative semantic image descriptions. *Procedia Computer Science*, 22:285 – 294, 2013.

[74] C. Schmid. A structured probabilistic model for recognition. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, page 490 Vol. 2, 1999.

[75] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 45–. IEEE Computer Society, 1998.

[76] K. Shanmugam, A. G. Dimkakis, and M. Langberg. Local graph coloring and index coding. *arXiv:1301.5359*, 2013.

[77] Karthikeyan Shanmugam, Mingyue Ji, Antonia Maria Tulino, Jaime Llorca, and Alexandros G. Dimakis. Finite length analysis of caching-aided coded multicasting. In *52nd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2014, Allerton Park & Retreat Center, Monticello, IL, September 30 - October 3, 2014*, pages 914–920, 2014.

[78] Gary J Sullivan, Jens Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[79] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[80] James W. Tanaka and Joseph A. Sengco. Features and their configuration in face recognition. *Memory & Cognition*, 25(5):583–592, 1997.

[81] Anne-Marie Tousch, StéPhane Herbin, and Jean-Yves Audibert. Semantic hierarchies for image annotation: A survey. *Pattern Recogn.*, 45(1):333–345, January 2012.

[82] S. Unal and A.B. Wagner. General index coding with side information: Three decoder case. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 1137–1141, July 2013.

[83] Giuseppe Vettigli, Mingyue Ji, Antonia Maria Tulino, Jaime Llorca, and Paola Festa. An efficient coded multicasting scheme preserving the multiplicative caching gain. In *2015 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, Hong Kong, China, April 26 - May 1, 2015*, pages 251–256, 2015.

[84] Giuseppe Vettigli, Mingyue Ji, Antonia Maria Tulino, Jaime Llorca, and Paola Festa. An efficient coded multicasting scheme preserving the multiplicative caching gain. In *2015 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, Hong Kong, China, April 26 - May 1, 2015* [83], pages 251–256.

[85] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 3156–3164, 2015.

[86] Chong Wang, David M. Blei, and Fei-Fei Li. Simultaneous image classification and annotation. In *CVPR*, pages 1903–1910. IEEE Computer Society, 2009.

[87] Meng Wang, Yue Gao, Ke Lu, and Yong Rui. View-based discriminative probabilistic modeling for 3d object retrieval and recognition. *Trans. Img. Proc.*, 22(4):1395–1407, April 2013.

[88] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4651–4659, June 2016.

[89] Luming Zhang, Yi Yang, Yue Gao, Yi Yu, Changbo Wang, and Xuelong Li. A probabilistic associative model for segmenting weakly supervised images. *IEEE Trans. Image Processing*, 23(9):4150–4159, 2014.

[90] Ruofei Zhang, Zhongfei Zhang, Mingjing Li, Wei-Ying Ma, and Hong-Jiang Zhang. A probabilistic semantic model for image annotation and multimodal image retrieval. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 846–851 Vol. 1, Oct 2005.

[91] H. Zouari, L. Heutte, and Y. Lecourtier. Simulating classifier ensembles of fixed diversity for studying plurality voting performance. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 1, pages 232–235 Vol.1, Aug 2004.