**Noname manuscript No.**
(will be inserted by the editor)

# Quality-Aware Predictor-based Adaptation of Still Images for the Multimedia Messaging Service

**Steven Pigeon · Stéphane Coulombe**

the date of receipt and acceptance should be inserted later

**Abstract** The Multimedia Messaging Service (MMS) allows users with heterogeneous terminals to exchange structured messages composed of text, images, sound, and video. The MMS market is growing rapidly, posing the problem of MMS adaptation, which is necessary to ensure terminal interoperability. Message adaptation involves technological challenges, especially considering the high volume of messages that this service can handle. In this work, we propose novel predictor-based dynamic programming approaches to MMS adaptation, which provide a framework for explicit maximization of the user experience, rather than relying on heuristics to deliver adapted messages satisfactorily. We show that the proposed solutions lead to noticeably superior image quality and faster transcoding times than comparable algorithms offered in products currently on the market and those described in the literature.

**Keywords** MMS, image adaptation, JPEG, optimization, predictor, dynamic programming, SSIM.

## 1 Introduction

The Multimedia Messaging Service (MMS) allows users with heterogeneous terminals to exchange structured messages composed of text, audio, still images, and video [29], and is a great source of revenue for mobile operators. According to Portio Research, 207 billion MMS messages were sent in 2011, and this number is expected to rise to 276.8 billion by 2016. With this volume of messages, MMS would maintain its position as the second most successful non voice mobile service to date, behind the highly lucrative Short Message Service (SMS) [4, 5]. Informa sees even higher MMS volumes by 2016, predicting that 387.5 billion MMS messages will be sent, representing 10.6% of global messaging revenues, at US$20.7 billion [3].

MMS technical specifications have been defined by the 3rd Generation Partnership Project (3GPP) and the Open Mobile Alliance (OMA) (and adapted in 3GPP2). The overall architecture of the MMS implementation in cellular networks

Département d'informatique, Université du Québec à Rimouski E-mail: steven_pigeon@uqar.ca · Department of Software and IT Engineering, École de technologie supérieure E-mail: stephane.coulombe@etsmtl.ca

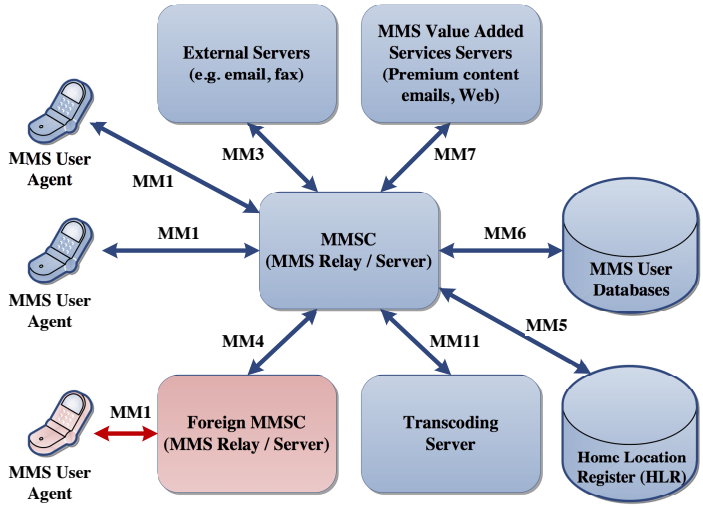


**Fig. 1** The multimedia messaging network architecture.

is shown in Fig. 1 [6,27]. The Multimedia Message Service Center (MMSC), also called MMS relay/server or MMS proxy-relay, is responsible for storing messages received from a user and relaying them to the intended recipient. It plays a similar role in MMS to that of the Short Message Service Center (SMSC) in SMS. However, since MMS is a premium service permitting the exchange of rich content between terminals with diverse capabilities (e.g. maximum message size and maximum image resolution), the MMSC must perform content adaptation to make the message suitable for viewing on the recipient's terminal.

As shown in Fig. 1, the MMSC provides access to the cellular network via the MM1 interface (or reference point) [6]. This is the interface used when two users having the same operator exchange multimedia messages. If the operators are different, then MM4 must be used between the users' respective MMS relays/servers. The MMSC provides access for value-added service providers (premium content emails, Web) via MM7. Access to external servers, such as email and fax servers, is performed through the MM3 reference point. MM11 allows the MMSC to access an external transcoding server to perform message adaptation, and is specified by the OMA Standard Transcoding Interface (STI) v1.0 [6,26]. To ensure interoperability, these interfaces are defined to operate over WAP/WSP and HTTP.

Figure 2 shows a typical message flow for multimedia message delivery between two mobile terminals having the same operator [28]. The steps are as follows:

1. The sender's terminal initiates a WAP POST request (a generalization of HTTP POST) to the MMSC in order to send a message. The message may, for instance, comprise several pictures taken by the sender's camera phone. This WAP POST operation uploads the message to the MMSC, which is then responsible for its delivery.
2. After the MMSC has stored the message, it sends a notification to the recipient's terminal to inform it that a new message has arrived. The notification is

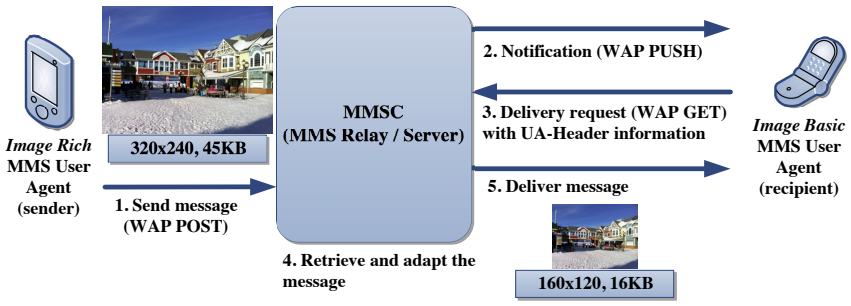


**Fig. 2** Typical multimedia message transactions between two heterogeneous terminals.

typically made using WAP PUSH (e.g. using SMS as the bearer). The notification contains a Uniform Resource Locator (URL) associated with the message (URLs being the standard mechanism used over the Internet to uniquely identify content, such as a Web page).

3. The notification triggers a WAP GET operation (a generalization of HTTP GET) on the recipient's terminal, which retrieves the message (using its URL) from the MMSC and sends it to the mobile device. That transaction contains information about the terminal type (UA-header). This information is useful to determine if the message needs to be adapted. For instance, since the UA-header value is unique to every device and software release, we can create a database containing the terminal capabilities of known UA-headers. This is one of several methods used to determine receiving terminal capabilities [11].
4. The MMSC retrieves the message from its database that corresponds to the URL. If required, it will adapt the message to meet the terminal's capabilities.
5. The MMSC sends the resulting message to the destination terminal. Since WAP/WSP or HTTP is used for the transfer, there is no risk of transmission error. Furthermore, the user is typically informed of the incoming MMS only when it has been completely received, thereby avoiding usability issues associated with a user impatiently monitoring the download of a message.
6. The terminal confirms reception of the message (not shown).
7. The MMSC sends a delivery report to the sender using WAP PUSH (not shown).

Clearly, MMS traffic will continue to grow very rapidly in the next few years, which will open up business opportunities, but also pose technological challenges [19]. As MMS usage increases and standards evolve, providers are being pressed to handle ever greater numbers of messages containing richer media content. This daunting task cannot be reduced to merely passing along messages, as server-side adaptation of messages, performed by the MMSC or an external transcoding server connected to it, is also necessary, in order to ensure interoperability among users [11]. The very high volume of messages to be handled by service providers will call for the most efficient adaptation algorithms and strategies to cope with the demand, the growth and evolution of standards, and the solution of scaling problems.

For MMS applications, a receiving terminal is characterized by its capabilities—or perhaps more accurately by its *limitations*—as defined by *profiles*. A profile determines the terminal's constraints, such as the maximum multimedia message size in bytes, the media types that the terminal can interpret, and the specific constraints of individual media types, such as maximum image resolution. A device supporting the "Content Rich" profile will support the JPEG and GIF image formats with resolutions up to $1600 \times 1200$ pixels and a maximum message size of 600 KB [29]. A device supporting the "Image Basic" profile will only support these image formats at resolutions up to $160 \times 120$ pixels and a maximum message size of 30 KB. We can see in Fig. 2 that the delivered message, which belonging to the Content Rich class was adapted to meet the lower resolution and memory capabilities of the receiving "Image Basic" class of terminal.

Server-side adaptation will ensure that not only each individual multimedia attachment is compatible with the receiving terminal but also that the message as a whole can be sent and correctly interpreted. Every attachment must be characterized and transformed, if need be, to satisfy the receiving terminal's constraints, whether by adjusting its format or its resolution. In the absence of server-side adaptation, a message exceeding the terminal's capabilities (either by message size or media type) can result in terminal-specific behavior that ranges from merely incomplete messages to terminal crash. If the server is capable of determining the capabilities of the terminal, but incapable or unwilling to perform adaptation, it can apply an alternative strategy, such as sending a text-only SMS along with the location of the original message, for the user to download or browse by other means [25]. While this ensures the delivery of content, it does not provide the user with a satisfactory experience, which means that server-side adaptation is preferable from the user's point of view.

Multimedia message adaptation, however, is not a trivial task as it does not suffice to apply heuristics such as successively reducing the file size or successively reducing the resolution of the various message attachments until a message satisfying the receiving terminal's various constraints is produced. We have shown in previous work that strategies involving both adaptation of JPEG compression parameters and scaling produce significantly better results than using either method alone [13]. In this work, we propose to extend this approach to the explicit optimization of compression parameters and scaling over a complete, multipart, image-only message, in order to achieve an adaptation that not only satisfies the constraints of the receiving device, but also maximizes overall perceived quality, with the expectation that the user experience will be maximized. While the techniques proposed in this work could be applied to other types of images, and to other media in general, we will, without loss of generality, restrict ourselves here to JPEG images. This is because the most popular use of MMS is to send photographs (JPEG images) from camera-equipped handsets [30]. Although it is also a popular method of delivering ringtones, these do not require transcoding.

*A priori*, adapting a single JPEG image to fit given constraints appears to be a trivial task. JPEG adaptation is, in fact, a costly process, especially considering the high volumes of images to be handled by server-side services. Solutions have been proposed to speed up transcoding under constraints such as maximum file size and maximum resolution. Some solutions address the problem of estimating the transcoded file size of a single JPEG image, but these are still computationally intensive (in addition to necessitate extensive modifications to existing JPEG

software libraries) or overly rigid, considering, for example only scaling by a power of two because they can be performed efficiently in the transform (DCT) domain [15, 24, 38]. Other solutions for image transcoding have been proposed in the broader context of Web or low-bandwidth resource access, if some solutions are designed to transcode an image so that it fits the constraints while minimizing transcoding time [16], others use a small number of fixed transcoding profiles setting both compression parameters and maximum image resolution to achieve adaptation [23]; neither quite expressing the problem in terms of explicit maximization of resulting image quality or user experience. More complex adaptation approaches, based on the understanding of message contents and image points of interest were also proposed [10,48], but, while promising, these techniques may be too computationally expensive for the type of high-volume transcoding needed by MMS providers [19]. Furthermore, these last methods only address the problem of adapting a single image, and not the problem of optimizing the global perceived quality of a multipart message.

Adapting an image, even in JPEG format, against maximum file size and resolution constraints, while maximizing perceived quality in a computationally efficient manner remains a challenge, as there are no established methods for estimating the resulting file size and quality of an image subject to changes in compression parameters and resolution. To this end, we have proposed predictors and systems in previous work designed to adapt images and messages [12,13,33]. These predictors have been exploited in [21,22] to develop a dynamic content adaptation framework applied to collaborative mobile presentations (e.g. OpenOffice Impress presentations). In this paper, we extend our previous work [35] where we proposed a general framework based on dynamic programming for the adaptation of image-only multipart messages, given receiving terminal constraints explicitly maximizing perceived quality of the whole message using a new algorithm, *step dynamic programming*. However, exact adaptation, even using dynamic programming, is very costly, as every possible adaptation parameter combination would have to be explicitly tried. To speed up optimization significantly, we propose to use predictors that yield a predicted file size and a predicted perceived image quality given particular adaptation parameters. In this work, we use a predictor that we presented in previous work [13], JQSP (for JPEG Quality and Size Predictor), which will serve as a "real life" prediction algorithm that exhibits moderate prediction errors, and compare it to *oracular* predictors that "predict" without error the resulting file size and image quality We show that the proposed optimization framework is resilient and shows graceful degradation in the presence of increased prediction error. We further show that our proposed optimization framework not only yields better perceived quality than approaches found in, and inspired by, prior art, but also that the transcoding times are significantly lower. This will greatly improve the performance of the multimedia messaging (MMS).

The work is organized as follows. Section 2 lays out the basic definitions, details the problem space, and presents the proposed solutions. Section 3 describes the test methodology and the predictors, the algorithms used for comparison. The test results of the proposed algorithms under various conditions are presented in section 4, and discussed in section 5. Finally, section 6 concludes the work.

## 2 Proposed Solution

Measuring the perceived visual quality of images subjected to transformation is a difficult task. Several algorithms and metrics have been developed to measure it objectively and automatically with a computer program. These can be classified as full-reference (FR), reduced-reference (RR), and no-reference (NR) methods [45]. In FR image quality assessment, the quality of a distorted image is evaluated by comparing it with a reference (the original) image. NR metrics assess the image quality without any reference (blind assessment), and RR metrics exploit some features of the reference image (partial information). The peak signal-to-noise ratio (PSNR) has been extensively used in literature as the FR estimate of perceived quality, but this metric has been shown to correlate poorly with perceived quality [37], in addition to being intolerant to transformations such as translation, scaling, and contrast—none of which necessarily translates into a degradation of quality to a human observer.

For the purposes of estimating the resulting quality, we use the widely known structural similarity index (SSIM) proposed by Wang *et al.* [44]. We chose the SSIM because of its popularity in the scientific community and its high level of accuracy (it correlates well to the Mean Opinion Scores (MOS)). A statistical evaluation of recent FR image quality assessment algorithms on various image databases has been performed in [47]. The results show that the SSIM is very accurate for various distortions, with an overall Spearman rank correlation coefficient of 0.948 (versus 0.876 for PSNR) on the Live Image Database, which comprises 779 images with various distortions, such as JPEG compression and blurring (which is similar to the scaling considered in this paper) [40]. Although the SSIM has been selected for our study, other metrics, such as Visual Information Fidelity (VIF) [39] or a less complex version of it [36], could also have been used to demonstrate the efficiency of the proposed MMS transcoding approach. The SSIM between an image region $x$ and an image region $y$ is defined as

$$\mathrm{SSIM} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_x^2 + C_2)} ,$$

where $\mu_x$ and $\sigma_x$ represent the mean and standard deviation of $x$, $\sigma_{xy}$ the cross-correlation of $x$ and $y$, $C_1 = (0.01\times255)^2$, and $C_2 = (0.03\times255)^2$. The SSIM is only applied on the luma, and this local measurement is applied to an entire image using a sliding window approach to obtain a quality map. The final SSIM score is the average of that quality map.

The SSIM is essentially a windowed correlation factor between original and distorted images, and, as such, yields results on $[-1, 1]$, but for our application we constrain the measure on $[0, 1]$, with any negative values mapped onto 0 (negative structural similarity values corresponding to cases where the local image structures are inverted [46], which should not occur when scaling or applying JPEG compression). Constraining the SSIM on $[0, 1]$ allows us to use the following objective function to represent the visual quality of the overall message:

$$\mathcal{Q}(M,T) = \prod_{i=1}^{n} Q\left(m_i, \mathcal{T}(m_i, t_i)\right) , \tag{1}$$

where $M = \{p\,;\, m_1, m_2, \ldots, m_n\}$ is the message to be adapted, with presentation $p$, and composed of the $n$ images $m_i$, with resolutions $R(m_i) = (w_i, h_i)$, and where

$T=\{t_1, t_2, \ldots, t_n\}$ is the series of transcoding parameters to be applied to the images. The transcoding parameters $t_i = (q_i, z_i)$ are such that $0 \leqslant q_i \leqslant 100$ is the output quality factor (using the semantics proposed by the IJG [2]) and $0 < z_i \leqslant 1$ a scaling factor used to resize the image. The function $\mathcal{T}(m_i, t_i)$, the transcoding function, applies transcoding parameters $t_i$ on image $m_i$, yielding an image with resolution $zR(m_i)=(z_i w_i, z_i h_i)$ compressed with quality factor $q_i$.

Finally, $0 \leqslant Q\left(m_i, \mathcal{T}(m_i, t_i)\right) \leqslant 1$ compares the original image $m_i$ to its transcoded version $\mathcal{T}(m_i, t_i)$ using the SSIM, as previously discussed. Since the images $m_i$ and $\mathcal{T}(m_i, t_i)$ may differ in resolution, that is, whenever $z \neq 1$, the image $\mathcal{T}(m_i, t_i)$ is rescaled to the original resolution $R(m_i)$ before comparison, estimating the viewing condition of the sender. Other strategies could be considered; for example, comparing at the scale reproducing the viewing condition of the receiver, or at an intermediate scale between sender and receiver conditions [13]. However, it seems realistic to use the sender's viewing conditions for the comparison, in order to estimate degradation in some absolute sense. Further, comparison is limited to the luma in order to avoid issues related to color subsampling. In fact, JPEG can use several chroma subsapling schemes [7, 32, 43], but 4:2:0 is the one most commonly used, leaving most of the image-quality information in the luma plane. In all cases, image scaling is performed using a Blackman filter, chosen because of its spectral properties [9].

Eq. (1), as a measure of the quality of the transcoded message, is to be maximized by a transcoding operation series $T$ under the constraints of a device $D$. The first constraint is that the total size of the transcoded images (plus the presentation metadata) must not exceed the maximum size allowed by the device for a message. The second constraint is a resolution constraint, where all the images must have a resolution lower than or equal to the device's maximum resolution.

The size constraint is expressed by

$$S(\mathcal{T}(M,T)) = \sum_{i=1}^{n} S\left(\mathcal{T}(m_i, t_i)\right) \leqslant S(D) - P(M,D)\,, \tag{2}$$

where $S(\mathcal{T}(M,T))$ is the file size in bytes of the transcoded message, $S\left(\mathcal{T}(m_i, t_i)\right)$ the file size in bytes of the transcoded image $\mathcal{T}(m_i, t_i)$, $S(D)$ the maximum message size in bytes for device $D$, and $P(M,D)$ the size, also in bytes, of the presentation information necessary for the adapted message $M$ to display correctly on device $D$. MMS uses the Synchronized Multimedia Integration Language (SMIL) as the presentation language. SMIL defines markup for the timing and layout of the audiovisual information, with enough flexibility to describe a slide show of several images, with text and audio [29]. The left-hand side of eq. (2) determines the *capacity*, the portion of the allowable byte budget used by the message using transcoding parameter series $T$. Let us also note that we will not interest ourselves further in the quantity $P(M,D)$, which would require adaptation of the presentation to estimate correctly. The presentation information is rather supposed to be a small, essentially negligible, part of the message budget.

The image resolution constraints for message $M$ and device $D$ are given by the orientation-independent resolution constraints

$$\begin{aligned} z_i \max(w_i, h_i) &\leqslant \max(w_D, h_D)\,, \\ z_i \min(w_i, h_i) &\leqslant \min(w_D, h_D) \end{aligned} \tag{3}$$

where $R(D) = (w_D, h_D)$, the receiving device's maximum image resolution, which is independent of the device's actual screen resolution. We therefore suppose that the receiving device scales and rotates the pictures on-screen for best viewing conditions.

The optimal transcoding series $T^*$ for a message $M$ and a device $D$ is therefore given by

$$T^* = \arg\max_{T \in \boldsymbol{T}(M,D)} Q(M,T)\,, \tag{4}$$

where $\boldsymbol{T}$ is the set of all possible series of transcoding parameters satisfying the constraints of eq. (2) and eqs. (3). The cardinality of $\boldsymbol{T}(M,D)$ can be very large (even infinite), if we do not constrain the transcoding parameters to a rather small set of discrete values to avoid a combinatorial explosion in the number of states examined by the algorithm solving eq. (4). The quality factor, as defined by the IJG, is an integer that takes values from 0 to 100 inclusive, but the scaling factor $0 < z_i \leqslant 1$ can take an infinite number of values. We solved this problem in previous work [12,13,33] by constraining the two transcoding parameters to take at most ten distinct values, that is, the quality factors were limited to the set $\{10, 20, \ldots, 100\}$ and scalings to $\{0.1, 0.2, \ldots, 1\}$, limiting the number of possible transcodings to 100. We use the same values in the experiments in this work.

To illustrate the notations that we have introduced, let us suppose that an MMS message is sent to an *Image Basic* class device $D$ (with a maximum resolution of $R(D) = (w_D, h_D) = (160, 120)$ and maximum message size $S(D) = 30\,\text{KB}$. The MMS message $M$ comprises a SMIL presentation part $p$ of size $P(M,D) = 2\,\text{KB}$ and two JPEG images, $m_1$ with a resolution of $(w_1, h_1) = (640, 480)$ of size $S(m_1) = 50\,\text{KB}$, and $m_2$, with a resolution of $(w_2, h_2) = (320, 240)$ of size $S(m_2) = 25\,\text{KB}$. We want to find, for each image, the best transcoding parameters $t_1 = (q_1, z_1)$ and $t_2 = (q_2, z_2)$ (quality factor and scaling) that meet the following constraints (from eq. (2) and eq. (3)):

$$S\left(\mathcal{T}(m_1, t_1)\right) + S\left(\mathcal{T}(m_2, t_2)\right) \leqslant S(D) - P(M,D) = 30\,\text{KB} - 2\,\text{KB} = 28\,\text{KB}$$

and

$$\begin{aligned} z_1 \max(w_1, h_1) = z_1 \times 640 &\leqslant \max(w_D, h_D) = 160\,, \\ z_1 \min(w_1, h_1) = z_1 \times 480 &\leqslant \min(w_D, h_D) = 120\,, \\ z_2 \max(w_2, h_2) = z_2 \times 320 &\leqslant \max(w_D, h_D) = 160\,, \\ z_2 \min(w_2, h_2) = z_2 \times 240 &\leqslant \min(w_D, h_D) = 120\,, \end{aligned}$$

where $\mathcal{T}(m_1, t_1)$ is the transcoded version of the first image using transcoding parameters $t_1 = (q_1, z_1)$, and similarly with $\mathcal{T}(m_2, t_2)$ for the second image. Combining these conditions, we obtain

$$\begin{aligned} S\left(\mathcal{T}(m_1, t_1)\right) + S\left(\mathcal{T}(m_2, t_2)\right) &\leqslant 28\,\text{KB}\,, \\ z_1 \leqslant 0.25 \text{ and } z_2 &\leqslant 0.5\,. \end{aligned}$$

Note that images $m_1$ and $m_2$, of size $50 + 25 = 75\,\text{KB}$, need to be reduced to occupy at most 28 KB (about a third of their original combined size). Among the transcoding parameters that satisfy these constraints, we select the set of

parameters that optimizes the objective cost function expressed by eq. (1), which we can rewrite in our example as

$$\mathcal{Q}(M,T) = Q\left(m_1, \mathcal{T}(m_1,t_1)\right) \times Q\left(m_2, \mathcal{T}(m_2,t_2)\right) \ .$$

The overall message's visual quality, as expressed in eq. (1), is the product of the visual quality of each transcoded message component. In this context, the product is more suitable than the sum, since the various qualities $Q\left(m_i, \mathcal{T}(m_i,t_i)\right)$, $1 \leqslant i \leqslant n$ are not compensatory, meaning that, if a transcoded component is of very poor quality, the overall message quality will be perceived as poor, no matter how good the quality of the other transcoded components. In fact, before combining two or more quality attributes to arrive at a single measure that reflects the nature of the problem at hand, these attributes should first be classified as either compensatory (i.e. can be summed) or non compensatory (i.e. can be multiplied). This is an aspect that is widely studied in the marketing and decision making fields [14, 20].

While maximizing eq. (1) is not the same as maximizing the average quality of the transcoded images, the expected average quality increases, and necessarily so, with increases in eq. (1), as a consequence of

$$\prod_{i=1}^{n} Q\left(m_i, \mathcal{T}(m_i,t_i)\right) \leqslant \min\left\{Q\left(m_i, \mathcal{T}(m_i,t_i)\right)\right\}_{i=1}^{n} ,$$

where $\{x_i\}_{i=1}^{n}$ denotes the sequence $\{x_1, \ldots, x_n\}$. The nature of $Q$ will furthermore cause the maximization of eq. (1) to reduce its variance [18, 41, 42]. Therefore, maximizing the proposed objective function will have the side effects of finding solutions with higher expected average quality and lower variance. Lower variance is particularly interesting, as it translates into a reduction of the risk of finding solutions where one transcoded image is of very poor quality and all the others are of good quality, in favor of solutions where quality is balanced among all the images.

The objective function eq. (1) is convex and separable, making it a Bellman equation [8], and therefore amenable to efficient optimization using dynamic programming. More specifically, the optimization problem considered in eq. (4) is a *distribution of effort* problem, where a finite quantity of resources is allocated strategically in order to maximize a gain function [8, 17]. For the current problem, the resources are transcoded file sizes, the sum of which is constrained by the maximum message size as stated in eq. (2); the gain from the allocation is the quality obtained as determined by the objective function, eq. (1), under the additional constraints of maximum resolution, as given by eqs. (3). This general class of problems has been studied extensively, and efficient polynomial-time algorithms exist to find which allocation of resources maximizes the objective function under the given constraints [17, 31].

Solving eq. (4) exactly is possible, but would require for all transcoding parameters examined by the algorithm that an actual transcoding is performed to measure resulting file size and quality, clearly a prohibitive process. But rather than performing a transcoding for every combination of transcoding parameters examined, we will resort to fast predictors that, given a (superficial) characterization of an image $m$ (such as original file size, quality factor, and resolution) and transcoding parameters $t$, will predict the resulting file size and quality of $\mathcal{T}(m,t)$, the transcoded image $m$ to which were applied transcoding parameters $t$.

We have presented such predictors in previous works [13,33], and in this study we use the file size and quality predictor presented in [13], which will be denoted JQSP (JPEG Quality and Size Predictor). To assess the proposed methods' resilience to prediction error, we will, in addition to JQSP, use oracular predictors, predictors with known characteristics, discussed in the next section, section 3.

The objective function using predictors is rewritten as

$$\widehat{\mathcal{Q}}(M,T) = \prod_{i=1}^{n} \widehat{Q}(m_i,t_i)\,, \tag{5}$$

where $\widehat{Q}(m_i,t_i)$ is the quality predictor taking an image $m_i$ (or, more precisely, its characterization, composed of its original quality factor, file size, and resolution) and a transcoding operation $t_i$, rather than using $Q(m_i,\mathcal{T}(m_i,t_i))$, which compares the actual transcoded image $\mathcal{T}(m_i,t_i)$ with the original image $m_i$.

The size constraint, eq. (2), must also be modified to accommodate predictors, and is rewritten as

$$\sum_{i=1}^{n} \widehat{S}(m_i,t_i) \leqslant S(D) - P(M,D)\,, \tag{6}$$

where $\widehat{S}(m_i,t_i)$ denotes the predictor of the size of image $m_i$ on which the transcoding parameters $t_i$ were applied. Eqs. (3), however, is unchanged, as resolution remains a deterministic function of $z_i$ and $R(m_i)$, and therefore contains no uncertainty.

The optimal predicted transcoding $\widehat{T}^*$ is given by:

$$\widehat{T}^* = \underset{\widehat{T}\in\widehat{\boldsymbol{T}}(M,D)}{\arg\max} \quad \widehat{\mathcal{Q}}(M,\widehat{T})\,, \tag{7}$$

where the $\widehat{T}$ are series of transcoding parameters drawn from the set $\widehat{\boldsymbol{T}}(M,D)$ of all series of transcoding parameters on message $M$ that (probably) satisfy the constraints eqs. (3) and eq. (6) of the device $D$.

To summarize, we formulate the problem of adapting an image-only MMS as a distribution of the effort problem amenable to dynamic programming. We further propose to reduce the complexity of the problem by replacing actual transcodings by size and quality predictors, in order to perform the optimization efficiently. In the next section, we discuss the generation of the set $\hat{\boldsymbol{T}}(M,D)$, the series of transcodings on message $M$ that (probably) satisfy the constraints of the device $D$. In the next section, we present the details of the proposed optimization algorithms and the details of the two comparison algorithms.

## 3 Transcoding Algorithms

JQSP introduced in [13], differs from the predictor introduced in [33], as it does not directly predict file sizes and quality from an image characterization and a transcoding operation but rather predicts transcoding parameters and resulting quality from an image characterization and a target file size. Either predictor can be used to create a set $\widehat{\boldsymbol{T}}(M,D)$ for a message $M$ and receiving device $D$, but, in this work, as we mentioned earlier, we use the JQSP. The JQSP predictor was trained on approximately 70 000 JPEG images gathered from the Internet in 2008,

using a web crawler starting at various popular sites [33]. The density of the JQSP predictions was adjusted so that target file sizes were set 5% apart, which limits the size of $\widehat{\boldsymbol{T}}(M, D)$ for optimization.

Proposing optimization methods based on a specific predictor, such as the JQSP, validates the predictor more than it validates the methods themselves. In order to show the properties of the proposed methods, such as resilience to predictor error, and ultimately determine the upper bound on attainable quality, we use *oracular* predictors. The oracular predictor "predicts" the exact file size and quality resulting from transcoding parameters applied to an image by actually performing the transcoding corresponding to the transcoding parameters. To further demonstrate that the proposed methods are resilient, that is, degrade gracefully in the presence of increasing errors in the predictors, we use predictors derived from the oracular predictor with a relative gaussian error on file size and quality of 1%, 2%, 5%, and 10%, 95% of the time. To populate $\widehat{\boldsymbol{T}}(M, D)$ using the oracular predictors, the transcoding parameters are limited to the quality factors of $\{10, 20, \ldots, 100\}$ and to scalings of $\{0.1, 0.2, \ldots, 1.0\}$, which limits the number of transcoding parameters to at most one hundred per image. We now present the details of our two proposed solutions to the problem of adaptation of JPEG-only messages, followed by those of the two comparison methods.

### 3.1 Dynamic Programming

The first of our proposed methods, which we refer to simply as "dynamic programming", is to solve eq. (7) directly by dynamic programming to obtain the predicted optimal transcoding parameter series. The optimal transcoding parameters found by explicit optimization are then used to perform message adaptation. If the adaptation yields a message larger than the maximum message size for the receiving device due to size prediction error—it cannot yield a message with images having incompatible resolutions—a new, smaller, maximum message size is set for the device and adaptation is retried. The maximum message size is adjusted by a factor $\alpha_1$ (set to 0.95 in our experiments) at each iteration, that is, we rewrite the size constraint eq. (6) as

$$\sum_{i=1}^{n} \widehat{S}(m_i, t_i) \leqslant \alpha_1^{k-1} S(D) - P(M, D) \tag{8}$$

for the $k$th iteration; at the first iteration, $k = 1$, the constraint is equivalent to eq. (6).

### 3.2 Step Dynamic Programming

However, as we discuss further in section 5, the experiments show that using dynamic programming as described above, prediction error cumulates, rather than being cancelled out—especially when the predictor is biased. The second of our proposed methods, which we refer to as "step dynamic programming", mitigates error propagation by proceeding by iterative refinement of the solution, again based on dynamic programming. Step dynamic programming first optimizes the

message globally and determines the predicted optimal transcoding parameter series, but transcodes only the first image (in attachment order). After the first image has been transcoded, its actual file size is observed and the budget for the remaining images is adjusted to take into account the transcoded image—and the corresponding prediction error. The remaining images are optimized jointly, and, again, only the first of the remaining images is transcoded, its transcoded size observed, and the budget adjusted, and so on, until all the images have been transcoded. Eq. (8) becomes

$$\sum_{i=1}^{s-1} S(\mathcal{T}(m_i, t_i^*)) + \sum_{i=s}^{n} \widehat{S}(m_i, t_i) \leqslant \alpha_1^{k-1} S(D) - P(M, D)$$

at step $s$ (with $s=2,3,\ldots,n-1$, while at $s=1$, we use eq. (8)) of the $k$th (with $k>1$) iteration. The $t_i^*$ denote the transcoding parameters already chosen (but not necessarily optimal in an absolute sense). The next step of optimization proceeds by solving the modified objective function on the last $n-s$ terms, corresponding to the images yet to be transcoded. The objective function eq. (5), at step $s=2,3,\ldots,n-1$, becomes

$$\widehat{\mathcal{Q}}_s(M,T) = \left( \prod_{i=1}^{s-1} \mathcal{Q}\left(m_i, \mathcal{T}(m_i, t_i^*)\right) \right) \left( \prod_{i=s}^{n} \widehat{Q}(m_i, t_i) \right) , \tag{9}$$

where the left part corresponds to images already transcoded (and therefore of known quality—which we do not necessarily need to observe), and the right part corresponds to the objective function to be maximized. At $s=1$, eq. (9) reverts to eq. (5).

Readjusting budget and re-optimizing at each transcoded image greatly reduces the propagation of prediction errors, with the consequence that, as we will see in section 4, the algorithm makes better use of capacity.

### 3.3 Comparative Algorithms

To compare our two proposed and novel solutions, we use two algorithms inspired by the fixed profile adaptation strategy of Mohan *et al.* [23], which, to the best of our knowledge, is the only relevant previous work.

The first comparative algorithm, "successive profiles", applies increasingly restrictive profiles to all the images until the transcoded message satisfies the constraints of the receiving device. For this algorithm, a *profile* defines both the quality factor and the maximum resolution. For example, a profile could limit the resolution to $640\times480$ with a quality factor of 90. The next profile could use the same resolution, but a quality factor of 80, the next could reduce the resolution to $320\times240$, but keep a quality factor of 80, and so on, each successive profile being more restrictive than the preceding one, reducing resolution, the quality factor, or both. The number and determination of useful profiles in this context depends on the performance objectives that are expected. The profiles used in our experiments are shown in Table 1. However, we will see in the next section that it is not useful to merely have a greater number of profiles.

**Table 1** Combination of resolution and quality factors to form the profiles used for the "successive profiles algorithm".

| Resolution | Quality Factors |
|---|---|
| 640×480 | 90, 80, 70, 60 |
| 320×240 | 90, 80, ..., 50 |
| 160×120 | 90, 80, ..., 40 |

The second comparative algorithm, "successive scaling", adjusts only the scaling, and using a fixed, but reasonable, quality factor of 85. Starting with the maximum resolution allowable for the receiving device, the algorithm successively scales down all the images (and compressing using the fixed quality factor) until the adapted message satisfies the device constraints. For each image $m_i$, the largest allowable scaling factor $0 < z_i \leqslant 1$, such that $z_i R(m_i) \leqslant R(D)$, is found. Adaptation proceeds by adjusting, at iteration $k = 1, 2, \ldots$, a global parameter $\beta_k$ (initially $\beta_1 = 1$) that is applied to every image, so that the scaling factor of image $m_i$ at step $k$ is $\beta_k z_i$, which yields an image of resolution $\beta_k z_i R(m_i)$. As scaling controls quadratically the file size (a scaling $z$ yields an image of relative surface $z^2$), a reasonable adjustment $\beta_{k+1}$ (for $k > 1$) is given by

$$\beta_{k+1} = \alpha_2 \sqrt{\frac{S(D) - P(M, D)}{S_k}} \,,$$

where $\alpha_2$ is a dampening factor (arbitrarily set to 0.95 in our experiments) to ensure that the budget is reduced further at each iteration and that the number of iterations is limited, $S(D)$ is the maximum message size for the receiving device $D$, $P(M, D)$ is the size of the presentation of message $M$ on device $D$, and $S_k$ is the size of the message obtained at step $k$. The adaptation terminates when a message satisfying the device constraints is produced.

Both comparative algorithms attempt to heuristically maintain a reasonable balance between quality factors and image scaling—one by using a fixed but relatively high quality factor, adjusting only resolution, and the other by using predefined profiles—in order to provide better image quality. We could also conceive of an algorithm where the images are reduced to $R(D)$, the maximum resolution for the receiving device, and where further adaptation is achieved only by using increasingly coarse quality factors until the message satisfies the receiving device constraints. However, that strategy would lead to the undesirable result of relatively high resolution images compressed with very low quality factors, showing conspicuous blocking artifacts. This algorithm would likely produce worse results than either of the proposed comparative algorithms, and so is not worth pursuing, in our opinion.

## 4 Experimental Results

For our experiments, we created four groups of 1000 MMS, with two (being the minimum to qualify as a "multipart" message) to five attached images. The images with resolutions between 320×200 and 3000×2000 were uniformly randomly chosen from a database of 370 000 images obtained by crawling the Web in the fall

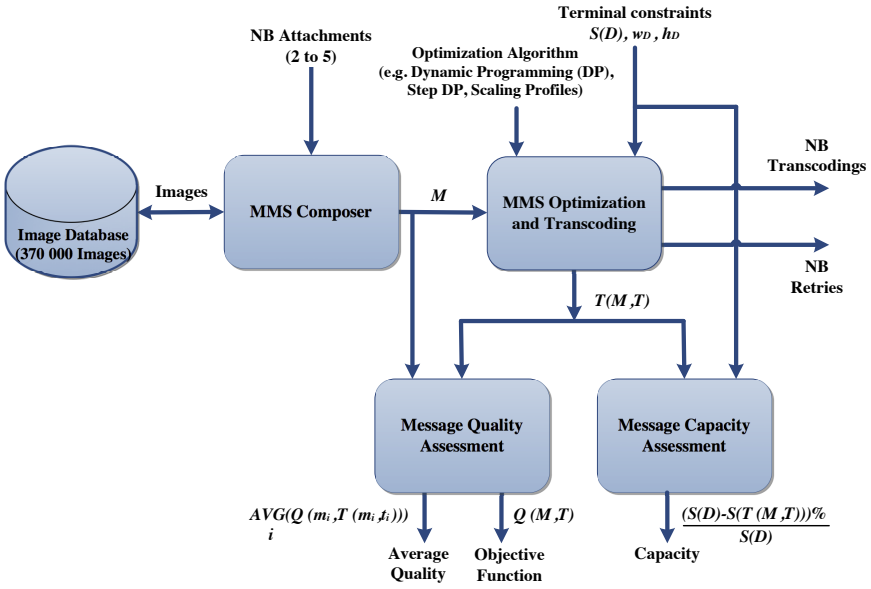


**Fig. 3** MMS adaptation validation architecture.

of 2010 [35], rather than from the database used in previous works [13, 33]. The profile chosen to test adaptation in our experiments was "Image Rich" (supporting images with resolutions up to 640×480 and a maximal message size of 100 KB). Forcing messages into the "Image Rich" profile from the original MMS (with an average message size of 284 KB, 563 KB, 790 KB, 1.2 MB, and 1.4 MB, for 1, 2, 3, 4, and 5 attachments respectively) demonstrates that the various algorithms tested were stressed with adaptation ratios of up to ≈14:1. The overall test architecture is shown in Fig. 3. For each number of attachments and each optimization algorithm and predictor precision level (varying from 1% to 10%, and then using JQSP), the *MMS Composer* created 1000 MMS messages $M$ that were transcoded in the *MMS Optimization and Transcoding* module to generate $\mathcal{T}(M,T)$. The *Message Quality Assessment* module then measured the resulting objective function as in eq. (1), and defining the average message quality as

$$\mathrm{AVG}(M,T) = \sum_{i=1}^{n} Q\left(m_i, \mathcal{T}(m_i,t_i)\right) \ .$$

The *Message Capacity Assessment* module measured the relative difference between the maximum message size $S(D)$ and the transcoded message size $S(\mathcal{T}(M,T))$. Finally, the number of transcodings performed, as well as the number of retries, was measured. All these recorded statistics made it possible to compare the various MMS transcoding (optimization) algorithms.

Let us describe the validation process in greater detail. The MMS in all groups (number of attachments) were transcoded using each of the compared algorithms. All the experiments used the same series of MMS, and the oracular predictors with

**Table 2** Summary for 2 attachments.

| Optimization Algorithm | Predictor | Average Transcodings | Average Retries | Average Capacity | Average Quality | Objective Function |
|---|---|---|---|---|---|---|
| Dynamic Programming | Oracle | 2.00 | 0.00 | 0.95 | 0.85 | 0.73 |
| | ±1% | 2.04 | 0.03 | 0.93 | 0.85 | 0.72 |
| | ±2% | 2.08 | 0.05 | 0.90 | 0.85 | 0.72 |
| | ±5% | 2.16 | 0.08 | 0.84 | 0.84 | 0.71 |
| | ±10% | 2.25 | 0.13 | 0.79 | 0.83 | 0.70 |
| | JQSP | 2.00 | 0.00 | 0.45 | 0.80 | 0.64 |
| Step Dynamic Programming | Oracle | 2.00 | 0.00 | 0.95 | 0.85 | 0.73 |
| | ±1% | 2.02 | 0.01 | 0.93 | 0.85 | 0.72 |
| | ±2% | 2.03 | 0.01 | 0.89 | 0.85 | 0.72 |
| | ±5% | 2.04 | 0.02 | 0.81 | 0.84 | 0.71 |
| | ±10% | 2.06 | 0.03 | 0.75 | 0.83 | 0.69 |
| | JQSP | 2.00 | 0.00 | 0.45 | 0.80 | 0.64 |
| Scalings | — | 4.55 | 1.28 | 0.93 | 0.82 | 0.67 |
| Profiles | — | 6.90 | 2.45 | 0.86 | 0.83 | 0.69 |

gaussian noise (described in section 3) used the same seed (and therefore the same pseudo-random sequence). Furthermore, in all the methods compared, we scaled images using a Blackman filter [9], with the actual image processing performed by ImageMagick's Magick++ library [1]. The experiments were performed on a Dell PowerEdge R210, with an Intel i3 540 CPU running at 3.07GHz, 4GB RAM, Ubuntu 11.04 with kernel 2.6, Magick++ 6.6.2, and G++ 4.5.2., a plausible setup for a transcoding node at the time of writing.

Tables 2 to 5 summarize the experimental results, showing, for each combination of number of attachments, optimization algorithms, and predictors, the resulting number of transcodings performed, the average number of retries (the number of times a new round of optimization is required because adaptation failed, see section 3), the capacity, the average quality of the transcoded images, and the objective function score. Figures 4 and 5 show the distribution of resulting capacity and image quality, respectively, for 5 attachments using box-plots using the usual configuration: 5% to 95% percentiles marking the whiskers, first and second quartile marking the limits of the box, the median crossing the box, and with dots showing outliers, observations below 5% or above 95%. Finally, table 6 presents the average transcoding times (in seconds) for both proposed algorithms using the JQSP predictor versus the two comparative algorithms. Figure 7 shows the distribution of transcoding times for 5 attachments, also using the JQSP predictor. Note that oracular times are excluded from the results as oracular predictors perform a great number of transcodings in order to formulate their "predictions".

## 5 Discussion

We can reasonably hypothesize that maximizing capacity (the portion of the allowable message used by the transcoded images) is essentially equivalent to maximizing perceived quality, and *vice versa*. This is the hypothesis underlying the heuristic approach involving successive scalings and successive profiles that merely attempts to find the largest images (one considering arbitrary resolutions, the other considering only profile-specific resolutions) that fit into the message in order to maximize quality. However, examining Tables 2 to 5, and figs. 4 and 5, we see that

**Table 3** Summary for 3 attachments.

| Optimization Algorithm | Predictor | Average Transcodings | Average Retries | Average Capacity | Average Quality | Objective Function |
|---|---|---|---|---|---|---|
| Dynamic Programming | Oracle | 3.00 | 0.00 | 0.98 | 0.84 | 0.59 |
| | ±1% | 3.17 | 0.08 | 0.97 | 0.84 | 0.59 |
| | ±2% | 3.32 | 0.13 | 0.95 | 0.83 | 0.58 |
| | ±5% | 3.66 | 0.25 | 0.92 | 0.83 | 0.57 |
| | ±10% | 3.89 | 0.32 | 0.88 | 0.82 | 0.55 |
| | JQSP | 3.01 | 0.01 | 0.61 | 0.80 | 0.51 |
| Step Dynamic Programming | Oracle | 3.00 | 0.00 | 0.98 | 0.84 | 0.59 |
| | ±1% | 3.05 | 0.02 | 0.96 | 0.84 | 0.59 |
| | ±2% | 3.06 | 0.02 | 0.95 | 0.83 | 0.58 |
| | ±5% | 3.13 | 0.04 | 0.91 | 0.83 | 0.57 |
| | ±10% | 3.11 | 0.04 | 0.87 | 0.82 | 0.55 |
| | JQSP | 3.02 | 0.01 | 0.63 | 0.80 | 0.51 |
| Scalings | — | 8.59 | 1.86 | 0.93 | 0.78 | 0.49 |
| Profiles | — | 14.83 | 3.94 | 0.86 | 0.78 | 0.48 |

**Table 4** Summary for 4 attachments.

| Optimization Algorithm | Predictor | Average Transcodings | Average Retries | Average Capacity | Average Quality | Objective Function |
|---|---|---|---|---|---|---|
| Dynamic Programming | Oracle | 4.00 | 0.00 | 0.99 | 0.83 | 0.47 |
| | ±1% | 4.38 | 0.14 | 0.97 | 0.82 | 0.46 |
| | ±2% | 4.62 | 0.21 | 0.96 | 0.82 | 0.46 |
| | ±5% | 5.21 | 0.35 | 0.94 | 0.81 | 0.44 |
| | ±10% | 5.88 | 0.51 | 0.90 | 0.80 | 0.42 |
| | JQSP | 4.03 | 0.03 | 0.70 | 0.78 | 0.38 |
| Step Dynamic Programming | Oracle | 4.00 | 0.00 | 0.99 | 0.83 | 0.47 |
| | ±1% | 4.03 | 0.01 | 0.98 | 0.82 | 0.46 |
| | ±2% | 4.09 | 0.02 | 0.97 | 0.82 | 0.46 |
| | ±5% | 4.18 | 0.05 | 0.94 | 0.81 | 0.44 |
| | ±10% | 4.30 | 0.08 | 0.92 | 0.80 | 0.42 |
| | JQSP | 4.12 | 0.03 | 0.75 | 0.79 | 0.39 |
| Scalings | — | 11.94 | 1.99 | 0.94 | 0.76 | 0.34 |
| Profiles | — | 23.62 | 4.91 | 0.84 | 0.76 | 0.34 |

**Table 5** Summary for 5 attachments.

| Optimization Algorithm | Predictor | Average Transcodings | Average Retries | Average Capacity | Average Quality | Objective Function |
|---|---|---|---|---|---|---|
| Dynamic Programming | Oracle | 5.00 | 0.00 | 0.99 | 0.81 | 0.35 |
| | ±1% | 5.57 | 0.17 | 0.97 | 0.80 | 0.34 |
| | ±2% | 6.08 | 0.29 | 0.96 | 0.80 | 0.33 |
| | ±5% | 6.93 | 0.45 | 0.94 | 0.79 | 0.32 |
| | ±10% | 7.75 | 0.60 | 0.92 | 0.78 | 0.30 |
| | JQSP | 5.11 | 0.06 | 0.74 | 0.76 | 0.26 |
| Step Dynamic Programming | Oracle | 5.00 | 0.00 | 0.99 | 0.81 | 0.35 |
| | ±1% | 5.04 | 0.01 | 0.98 | 0.80 | 0.34 |
| | ±2% | 5.08 | 0.02 | 0.98 | 0.80 | 0.33 |
| | ±5% | 5.22 | 0.05 | 0.96 | 0.79 | 0.32 |
| | ±10% | 5.33 | 0.06 | 0.94 | 0.78 | 0.30 |
| | JQSP | 5.22 | 0.04 | 0.82 | 0.77 | 0.28 |
| Scalings | — | 15.02 | 2.00 | 0.94 | 0.73 | 0.23 |
| Profiles | — | 33.36 | 5.67 | 0.88 | 0.75 | 0.22 |

**Table 6** Times, in seconds

| Number of Attachments | Dynamic Programming | Step Dynamic | Scalings | Profiles |
|---|---|---|---|---|
| 2 | 0.09 | 0.09 | 0.19 | 0.31 |
| 3 | 0.13 | 0.12 | 0.33 | 0.62 |
| 4 | 0.18 | 0.17 | 0.47 | 0.99 |
| 5 | 0.21 | 0.20 | 0.55 | 1.30 |

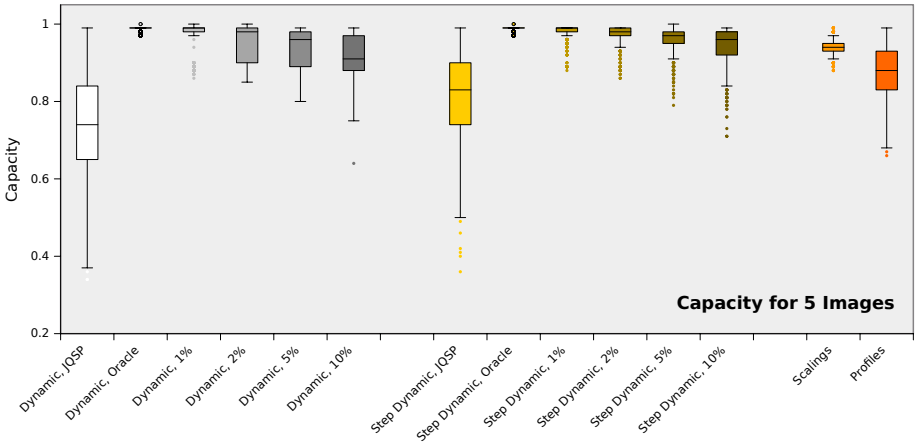


**Fig. 4** Box-plots of capacities resulting from the various algorithms, for 5 images per message.

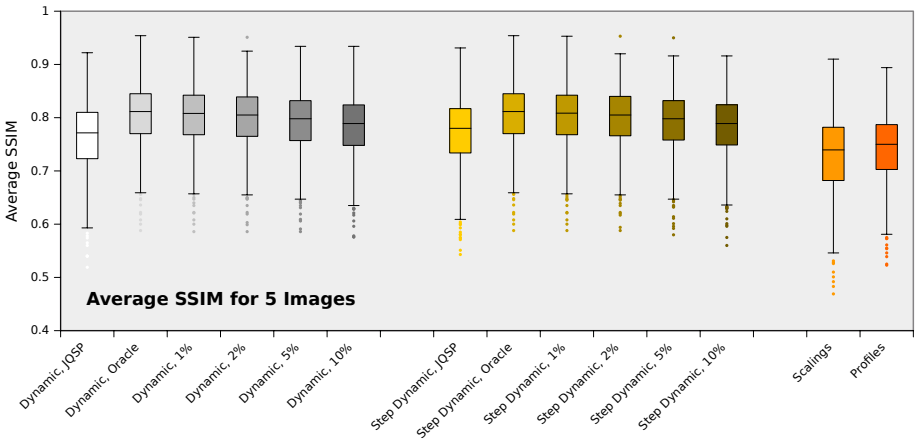


**Fig. 5** Box-plots of per-message average SSIM resulting from the different algorithms, for 5 images per message.

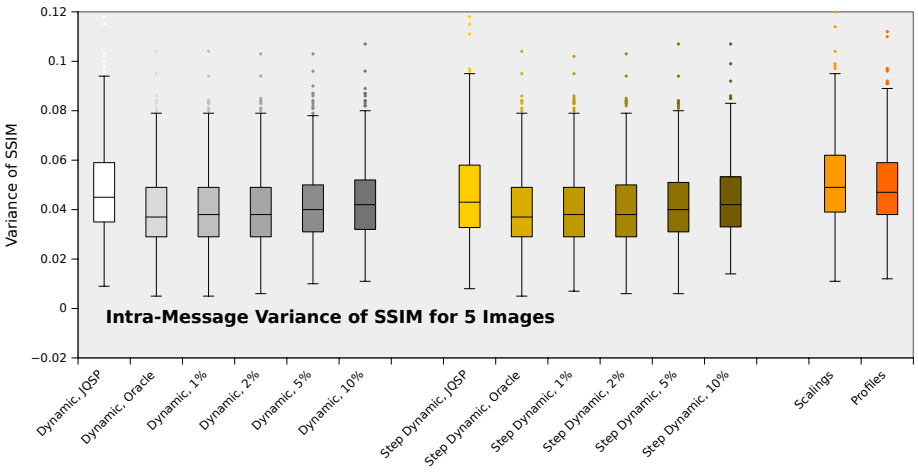


**Fig. 6** Box-plots of the intra-message SSIM variance resulting from the different algorithms, for 5 images per message.

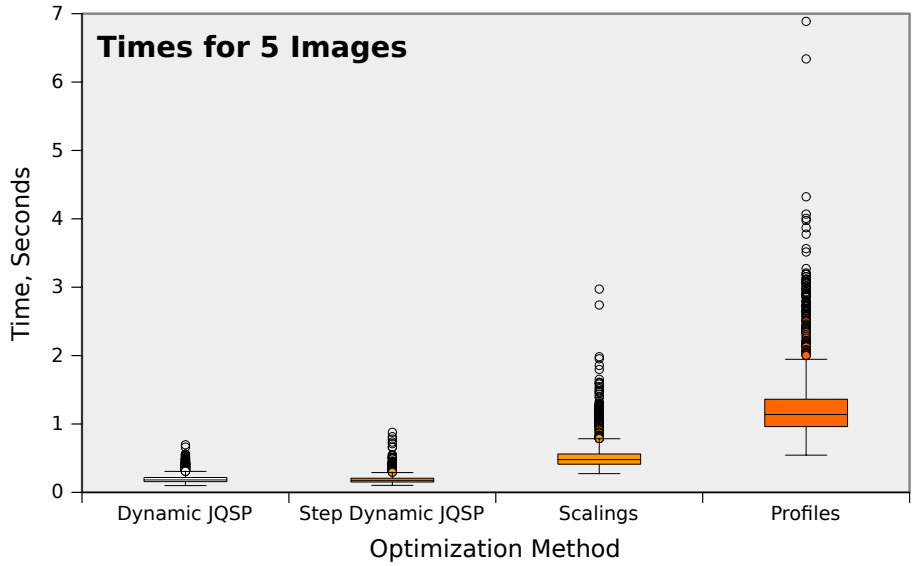


**Fig. 7** Box-plots of the times, in seconds, for the different algorithms on 5 images per message.

this hypothesis is not verified. While it is true that capacity and resulting message quality are correlated, it does not suffice to maximize capacity to maximize quality. Indeed, the successive scalings adaptation method usually yield high capacity with resulting quality only comparing to the successive profiles method, but with much worse quality than the two proposed explicit optimization methods.

In a similar way, we could hypothesize that it suffices to maximize the average image quality as the objective function rather than an objective function such as

eq. (1) (or eqs. (5) and (9)). Again, while for maximization, the average and the product of image quality are correlated, it is preferable to maximize the product as it has the distinct advantage of rejecting solutions where one or more of the transcoded images are of very poor quality, as maximizing the product (especially of values between 0 and 1) also requires maximizing individual image quality, with the side-effect of reducing variance [18, 41, 42]. Using an average or sum-like objective function, we could find ourselves with the case of a transcoding solution for a message with five images with four high-quality images but one image with exceedingly poor quality (which would be unacceptable) being chosen over a preferable solution where all five images are of approximately equally good quality (therefore with a small variance), simply because the *average* quality of the first solution is higher.

We also examined the order in which the step dynamic algorithm proceeds to optimize attachments. There are tree possible strategies for ordering attachment prior to optimization: images sorted in increasing file sizes, images sorted in decreasing file sizes, and the random order in which they are found in the message. While our experiments show no significant difference between orderings (which is expected when prediction error is small), we decided to rely on the natural, or random, message order. The rationale behind this decision is that since most of the traffic is composed of photographs, it is not unreasonable to assume that the photographs are taken from the same device (and therefore show similar characteristics such as resolution, file size, and compression parameters) and that, therefore, message order will not differ significantly from any other order.

Although the predictor and optimization algorithms are disjoint pieces of the proposed system, the expected accuracy and precision of the predictor will nonetheless play a major role in the quality of the transcoded messages. However, we show that increased predictor error will cause the performance of the proposed algorithms to degrade gracefully, first first by using increasingly noisy oracular predictor, second by using a "real life" low-cost predictor, JQSP, that exhibit prediction error. Examining Fig. 4 for capacity, and Figs. 5 and 6 for the resulting quality, we see that, indeed, the performance of adaptation only degrades progressively as predictor error increases. With the error-free oracular predictor, both proposed algorithms, as expected, find very good solutions, using essentially all capacity yielding a high average quality solution (but due to the quantization of transcoding parameters, discussed in section 3, the oracle may not find a solution using exactly 100% capacity). The JQSP predictor is doing much worse than the oracular predictors (as it is biased and overestimates file size [34]) but would compare to an oracular predictor with $\approx$15% error.

Resilience to predictor error and bias is a major problem in optimization. If the predictor error is symmetric (and perhaps vaguely gaussian), the errors would tend to cancel each other out; but if, like JQSP, the errors are asymmetric, an algorithm such as one-shot dynamic programming would not be able to cope with accumulated errors. However, the step dynamic programming solution can compensate for accumulated error as it transcodes a single image, observes the transcoded size, and adjusts its size constraint accordingly. Looking at Fig. 4, it is clear that this step-by-step strategy allows the optimization algorithm to make much better use of the capacity (with lower variance) than the one-shot dynamic programming approach, an effect that is also seen on the resulting average image quality, although to a lesser extent, as shown in Fig. 5. Fig. 6 shows the distribution

of the variance of the quality within messages, which indicates, as hypothesized earlier, that the objective function forces the intra-message variance to remain low.

The SSIM, being a windowed correlation factor between the original and the distorted image, is difficult to interpret intuitively. SSIM measures quality nonlinearly, and a SSIM score of 0.85 (see, for example, table 2) cannot be interpreted as 5% better than a score of 0.8: the difference tells us, in fact, that the first image is *exponentially* better than the second one [44, fig. 8 d)]. The useful range of the SSIM is $\approx[0.7, 1]$, which maps to MOS of about 30 to 100 (30 being poor, and 100 perfect). Images with scores significantly lower than 0.75 are of very poor quality, and an adaptation algorithm could use this lower bound as an explicit constraint for optimization. In our experiments, however, we only maximize quality explicitly, and the form of the objective function, eq. (1), only imposes an indirect constraint on the lower bound of image quality.

While absolute execution times of the algorithms are something of an implementation detail, it is nonetheless interesting to examine how implementations compare. Consider table 6 and fig. 7. First let us compare dynamic programming versus step dynamic programming. Times show that both variants perform essentially the same number of transcodings and that the execution times are comparable, and that the variance in execution time of the two proposed method is much lower than for comparative algorithms. This means that the optimization process is entirely dominated by the time for the actual transcodings, and that the time spent in the optimization *per se* and in querying the predictors is comparatively negligible.

For the optimization time to remain negligible compared to transcoding time, not only do we need the optimization algorithm to be fast (i.e. well implemented), but the predictors must be very efficiently computed as well. The JQSP predictor, as discussed in section 3, predicts transcoding parameters from an image characterization and a target file size, an operation that reduces, at worst, to a $O(\log N)$ search if the table contains $N$ irregularly spaced target file sizes for a given characterization [13]. The cost can be further reduced, to $O(1)$, for regularly spaced target file sizes, if the query parameters are quantized and the quantized values are used to index a multidimensional look-up table. Oracular predictors cannot be considered in the context of high-volume transcoding, since they are far too expensive to invoke—as a "prediction" is formulated from actual transcodings—but, if they cannot be used for comparing execution times, they are a good means for measuring the graceful degradation of quality in a controlled way, as the prediction error grows.

Another factor that will greatly influence optimization time is the number of transcoding parameters examined while solving eq. (5). While solving eq. (5) using dynamic programming is computationally optimal, it remains an $O(nm^2)$ algorithm, even if we suppose the predictors to be in constant time, where $n$ is the number of images to adapt and $m$ the average number of transcoding parameters tested per image [17, 31]. As $n$ is fixed (therefore excluding the possibility of dropping images), a speed-up can only be gained by the reduction of $m$. The set of transcoding parameters series, whether the exact $\boldsymbol{T}(M, D)$ or the predicted $\widehat{\boldsymbol{T}}(M, D)$, can be pruned without affecting optimality by excluding transcodings yielding images exceeding either the maximum message size or maximum resolution for the receiving device. We can further reduce the complexity by considering prunings that affect the optimality of the solution. For example, one could exclude

transcoding parameters that would yield very poor quality, defined by a user-specified threshold. One could also prune the set of possible transcodings to have transcoding parameters that yields (predicted) relative file sizes set at least 5% apart, or any other such heuristic that yields a satisfactory trade-off between parameter density, optimization speed, predictor error, retries, probability of failing to find a solution, and resulting adapted message quality.

In the same way, we could accelerate the successive profile algorithm by considering even fewer profiles. This would make it faster, but also coarser, yielding even worse results. We could be tempted to *add* profiles. However, it would not be sufficient to merely add profiles, certainly not without changing how profiles are applied. The profiles are applied in the order shown in table 1, stepping down resolution only when solutions using the lowest quality factors given the currently examined resolution have been explored. A better strategy would be to consider a profile, say, Image Rich, but with intermediate resolutions, such as $600\times450$ and $533\times400$ (or other $4:3$ aspect ratio resolutions), in combination with different quality factors. However, rather than trying a resolution with all its listed quality factors, and then moving on to the next resolution if no solution is found, it would be preferable to try resolution and quality factor combinations in descending order of expected resulting file size. This would allow the successive profile algorithm to find solutions with smaller images encoded with a larger quality factor, although this would not speed up the transcoding process.

For a high-volume service provider, such as a telecommunications operator, an algorithm that produces satisfactory adaptation of messages at the lowest possible computational cost (as adaptation rather mundanely translates into server racks, floor space, and electricity bills) is the preferable algorithm. In this work, we show that the proposed dynamic programming-based algorithms are interesting solutions, faring significantly better than comparative algorithms (both largely inspired from what is currently found in commercial products and in the literature). However, it is an open question as to how much sub-optimality—and how one defines optimality in this context—the users are willing to accept without taking notice of image/message degradation, and therefore which trade-off are available to service providers. One can surely consider using varying strategies depending on time of day and network traffic, possibly even adapting computational effort depending on the subscribers' data plans, network traffic, or other transient considerations. All these are user-experience considerations that cannot be addressed by the current work, but are certainly worthwhile exploring further.

## 6 Conclusion

In this work, we have shown that the two proposed predictor-based dynamic programming multipart message adaptation algorithms maximize quality explicitly (as a proxy for user experience), and also make better use of message capacity (the portion of the allowable message size used) than the comparative algorithms inspired by products currently on the market and described in the literature. We have also shown that, while predictor accuracy is important, our proposed algorithms degrade very gracefully with increases in predictor error, making them robust to prediction errors. Furthermore, our proposed algorithms are significantly

faster and better than earlier solutions, and would be of great benefit to the Multimedia Messaging Service.

**Acknowledgements** This work was funded by Vantrix Corporation and by the Natural Sciences and Engineering Research Council of Canada under the Collaborative Research and Development Program (NSERC-CRD 428942-11).

## References

1. ImageMagick and Magick++. http://www.imagemagick.org/ [Last accessed: 10 February 2013]
2. The Independent JPEG Group. http://www.ijg.org/ [Last accessed: 10 February 2013]
3. Informa telecoms and media: SMS will remain more popular than mobile messaging apps over next five years. http://blogs.informatandm.com/4971/press-release-sms-will-remain-more-popular-than-mobile-messaging-apps-over-next-five-years/ [Last accessed: 10 February 2013] (2012)
4. mobiThinking: Global mobile statistics 2012 Part C: Mobile marketing, advertising and messaging. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/c [Last accessed: 10 February 2013] (2012)
5. Portio Research: Mobile Messaging Futures 2012-2016. http://www.portioresearch.com/en/reports/current-portfolio/mobile-messaging-futures-2012-2016.aspx [Last accessed: 10 February 2013] (2012)
6. 3GPP TS 23.140 V6.16.0: Multimedia Message Service (MMS); Functional Description; Stage 2 (Release 6) (2009). http://www.3gpp.org/ftp/Specs
7. Acharya, T., Tsai, P.S.: JPEG 2000 Standard for Image Compression: Concepts, Algorithms, VLSI Architectures. Wiley-Interscience (2004)
8. Bellman, R.: Dynamic Programming. Dover, New York (2003)
9. Blackman, R.B., Tukey, J.: The Measurement of Power Spectra, from the Point of View of Communications Engineering. Dover (1959)
10. Chen, R.C.S., Yang, S.J.H., Zhang, J.: Enhancing The Precision of Content Analysis in Content Adaptation using Entropy-Based Fuzzy Reasoning. Expert Systems with Applications **37**, 5706–5719 (2010)
11. Coulombe, S., Grassel, G.: Multimedia Adaptation for the Multimedia Messaging Service. IEEE Communication Magazine **42**(7), 120–126 (2004)
12. Coulombe, S., Pigeon, S.: Quality-Aware Selection of Quality Factor and Scaling Parameters in JPEG Image Transcoding. In: Procs. IEEE 2009 Computational Intelligence for Multimedia, Signal, and Video Processing (CIMSVP)
13. Coulombe, S., Pigeon, S.: Low-Complexity Transcoding of JPEG Images with Near-Optimal Quality Using a Predictive Quality Factor and Scaling Parameters. IEEE Trans. Image Processing **19**(3), 712–721 (2010)
14. Dieckmann, A., Dippold, K., Dietrich, H.: Compensatory versus Noncompensatory Models for Predicting Consumer Preferences. Judgment and Decision Making **4**(3), 200 – 213 (2009)
15. Dugad, D., Ahuja, A.: A Fast Scheme for Image Size Change in the Compressed Domain. IEEE Trans. Circuits and Systems for Video Technology **11**(4), 461–474 (2001)
16. Han, R., Bhagwat, P., LaMaire, R., Mummert, T., Perret, V., Rubas, J.: Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing. IEEE Personal Communications Magazine **5**(6), 8–17 (1998)
17. Hillier, F.S., Lieberman, G.J.: Introduction to Operations Research, 9th edn. McGraw-Hill Science (2009)
18. Ishihara, T.: The Distribution of the Sum and the Product of Independent Uniform Random Variables Distributed at Different Intervals. Trans. Japanese Soc. for Industrial and Applied Mathematics **12**(3), 197–207 (2002)
19. J. Dwyer, III: MMS to Prosper as Mobile Marketing Becomes Mainstream. Wireless Week (2011)
20. Lee, L., Anderson, R.: A Comparison of Compensatory and Non-Compensatory Decision Making Strategies in IT Project Portfolio Management (2009). URL http://aisel.aisnet.org/irwitpm2009/9

21. Louafi, H., Coulombe, S., Chandra, U.: Quality Prediction-Based Dynamic Content Adaptation Framework Applied to Collaborative Mobile Presentations. IEEE Transactions on Mobile Computing **99**(PrePrints), 1–1 (2012). DOI 10.1109/TMC.2012.173
22. Louafi, H., Coulombe, S., Chandra, U.: Efficient Near-Optimal Dynamic Content Adaptation Applied to JPEG Slides Presentations in Mobile Web Conferencing. In: to appear in Procs. IEEE International Conference on Advanced Information Networking and Applications (AINA-2013) (2013)
23. Mohan, R., Smith, J.R., Li, C.S.: Adapting Multimedia Internet Content for Universal Access. IEEE Trans. Multimedia **1**(1), 104–114 (1999)
24. Mukherjee, J., Mitra, S.K.: Image Resizing in the Compressed Domain using Subband DCT. IEEE Trans. Circuits and Systems for Video Technology **12**(7), 620–627 (2002)
25. Nokia: How to Create MMS Services. Whitepaper (2003)
26. Open Mobile Alliance: Standard Transcoding Interface Specification Version 1.0 (2007). OMA-TS-STI-V1_0-20070515-A
27. Open Mobile Alliance: Multimedia Messaging Service Architecture Version 1.3 (2011). OMA-AD-MMS-V1_3-20110913-A
28. Open Mobile Alliance: Multimedia Messaging Service Client Transactions (2011). OMA-TS-MMS_CTR-V1_3-20110913-A
29. Open Mobile Alliance: Multimedia Messaging Service Conformance Document (2011). OMA-TS-MMS_CONF-V1_3-20110913-A
30. Pannu, P., Tomar, Y.: ICT4D Information Communication Technology For Development. IK International Pvt Ltd (2010)
31. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover (1998)
32. Pennebaker, W.B., Mitchell, J.L.: JPEG Still Image Data Compression Standard. Kluwer Academic Publishers (1992)
33. Pigeon, S., Coulombe, S.: Computationally Efficient Algorithms for Predicting the File Size of JPEG Images Subject to Changes of Quality Factor and Scaling. In: Procs. 24th Queen's University Biennial Symposium on Communications (2008)
34. Pigeon, S., Coulombe, S.: Efficient Clustering-based Algorithm for Predicting File Size and Structural Similarity of Transcoded JPEG Images. In: Procs. IEEE International Symposium on Multimedia (ISM), pp. 137–142 (2011)
35. Pigeon, S., Coulombe, S.: Optimal Quality-Aware Predictor-Based Adaptation of Multimedia Messages. In: Procs. The 6th IEEE Int. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, pp. 496–499 (2011)
36. Rezazadeh, S., Coulombe, S.: Low-complexity computation of visual information fidelity in the discrete wavelet domain. In: Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on, pp. 2438–2441 (2010)
37. Rezazadeh, S., Coulombe, S.: A Novel Discrete Wavelet Domain Error-Based Image Quality Metric with Enhanced Perceptual Performance. International Journal of Computer and Electrical Engineering **4**(2), 390–395 (2012)
38. Ridge, J.: Efficient Transform-Domain Size and Resolution Reduction of Images. Signal Processing: Image Communication **18**(8), 621–639 (2003)
39. Sheikh, H., Bovik, A., de Veciana, G.: An information fidelity criterion for image quality assessment using natural scene statistics. Image Processing, IEEE Transactions on **14**(12), 2117–2128 (2005)
40. Sheikh, H.R., Seshadrinathan, K., Moorthy, A.K., Wang, Z., Bovik, A.C., Cormack, L.K.: Image and video quality assessment research at LIVE. http://live.ece.utexas.edu/research/quality/ [Last accessed: 10 February 2013]
41. Springer, M.D., Thompson, W.E.: The Distribution of Products of Independent Random Variables. SIAM Journal on Applied Mathematics **14**(3), 511–526 (1966)
42. Springer, M.D., Thompson, W.E.: The Distribution of the Products of Beta, Gamma and Gaussian Random Variables. SIAM Journal on Applied Mathematics **18**(4), 721–737 (1970)
43. Taubman, D., Marcellin, M.: JPEG2000: Image Compression Fundamentals, Standards and Practice. Springer (2002)
44. Wang, Z., Bovick, A.C., Sheikh, H.R., Simoncelli, E.P.: Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Trans. Image Processing **13**(4), 600–612 (2004)
45. Wang, Z., Bovik, A.: Modern Image Quality Assessment. Synthesis Lectures on Image, Video, & Multimedia Processing. Morgan & Claypool (2006). URL http://books.google.ca/books?id=F6lYVwyZJz4C

46. Wang, Z., Bovik, A.C., Simoncelli, E.P.: Structural Approaches to image quality assessment, 2nd edn. Academic Press (2005)
47. Wang, Z., Li, Q.: Information Content Weighting for Perceptual Image Quality Assessment. Image Processing, IEEE Transactions on **20**(5), 1185–1198 (2011)
48. Yan, W.Q., Kankanhalli, M.S.: Multimedia Simplification for Optimized MMS Synthesis. ACM. Trans. Multimedia Computing, Communications, and Applications (TOMCCAP) **3**(1) (2007)