Hypergraph Grammar-Based Model of Adaptive Bitmap Compression

Grzegorz Soliński[1], Maciej Woźniak[1(✉)], Jakub Ryzner[1], Albert Mosiałek[1], and Anna Paszyńska[2]

[1] Department of Computer Science, AGH University of Science and Technology, Kraków, Poland
macwozni@agh.edu.pl

[2] Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, Kraków, Poland
anna.paszynska@uj.edu.pl

Abstract. JPEG algorithm defines a sequence of steps (essential and optional) executed in order to compress an image. The first step is an optional conversion of the image color space from RBG (red-blue-green) to YCbCr (luminance and two chroma components). This step allows to discard part of chrominance information, a useful gain due to the fact, that the chrominance resolution of the human eye is much lower than the luminance resolution. In the next step, the image is divided into 8×8 blocks, called MCUs (Minimum Coded Units). In this paper we present a new adaptive bitmap compression algorithm, and we compare it to the state-of-the-art of JPEG algorithms. Our algorithm utilizes hypergraph grammar model, partitioning the bitmap into a set of adaptively selected rectangles. Each rectangle approximates a bitmap using MCUs with the size selected according to the entire rectangular element. The hypergraph grammar model allows to describe the whole compression algorithm by a set of five productions. They are executed during the compression stage, and they partition the actual rectangles into smaller ones, until the required compression rate is obtained. We show that our method allows to compress bitmaps with large uniform areas in a better way than traditional JPEG algorithms do.

Keywords: Hypergraph grammar · Bitmap compression · Adaptive projection-based interpolation

1 Introduction

Although baseline JPEG is still the most commonly used compression algorithm, a number of algorithms have emerged as an evolution to JPEG. The most prevalent one is JPEG2000 standard (ITU-T T.800 — ISO/IEC 15444-1), which introduces usage of the Discreet Wavelet Transform in place of the Discreet Cosine Transform used in traditional JPEG, as well as usage of a more sophisticated entropy encoding scheme [1]. The standard also introduces an interesting feature,

V. V. Krzhizhanovskaya et al. (Eds.): ICCS 2020, LNCS 12139, pp. 118–131, 2020.
https://doi.org/10.1007/978-3-030-50420-5_9

which gives a potential boost in compression ratio or quality on some images - Region of Interest (ROI) coding [2]. Region of Interest is a predetermined portion of the image of an arbitrary shape, that is coded before the rest of the image. ROI part of the image and the background can also be compressed with different qualities, which makes JPEG2000 possess some traits of adaptivity. The gain in compression performance of JPEG2000 in comparison to JPEG is mainly denoted by the lack of DCT-specific block artifacts of the compressed images, as well as generally better-quality images for compression ratios exceeding 20:1 [3].

Another standard meant as an evolution to baseline JPEG, originally developed by Microsoft as HD Photo, eventually got published as (ITU-T T.832 — ISO/IEC 29199-2), commonly referred to as JPEG XR. The new standard supports higher compression ratios than baseline JPEG for encoding an image with equivalent quality by introducing lossless color space transformation and integer transform employing a lifting scheme in place of JPEG's slightly lossy RGB to YCbCr linear transformation and the DCT, respectively. It also introduces a different organization of the blocks and another level of frequency transformation [4]. As the experiments carried out by the JPEG committee have shown, the compression performance of JPEG XR was typically very close to the performance of JPEG 2000, with the latter slightly outperforming JPEG XR. However, the difference was generally marginal in the scope of bit rates meaningful for digital photography [5].

Recent work performed by the JPEG committee and its subsidiaries, as well as other contributors, resulted in a series of JPEG related standards/extensions. One of them is JPEG XT (ISO/IEC 18477), a standard that specifies a number of backwards compatible extensions to the legacy JPEG standard. It addresses several plain points, that have stuck to JPEG over the years, such as support for compression of images with higher bit depths (9 to 16 bits-per-pixel), high dynamic range imaging and floating point coding, lossless compression and alpha channels coding [6]. Another recently presented standard is JPEG XS (ISO/IEC 21122). Unlike former standards, the focus JPEG XS was to provide a robust, cost-efficient codec for video-over-IP solutions, suitable for parallel architectures and ensuring a minimal end-to-end compression latency as well as minimal incremental quality degradation on re-compression [7]. As objectives of developing JPEG XT and JPEG XS standards were somewhat different than their e.g. that of JPEG2000, they rather introduce sets of features useful for some particular applications - such as HDR photography or video streaming, than contributing any particular gain in compression performance or visual fidelity.

In this paper we employ the adaptive algorithm for image compression. We use the hypergraph grammar-based approach to model the adaptive algorithm. The hypergraph grammar was originally created by A. Habel and H. Kreowski [8,9]. They were used to model adaptive finite element method computations [10,11], multi-frontal solver algorithm [12–14]. They define the hypergraph data structure and the abstraction of graph grammar productions. The process of adaptive compression of a bitmap can be expressed by a sequence of graph grammar productions. In our model we define five graph grammar productions

to describe the adaptive algorithm. We compare our approach with the JPEG standards described above. Alternative graph grammars include the CP-graph grammar, also used for modeling adaptive finite element method [15–19], but for the CP-graph grammar generates hierarchical structures with the history of refinements, while for the bitmap compression application the flat graph seems to be more applicable.

2 Hypergraph and Hypergraph Grammar

In this section the main definitions connected with hypergraphs and hypergraph grammar are presented.

A hypergraph is a special kind of graph consisting of nodes and so called hyperedges joining the nodes. One hyperedge can join two or more nodes. The nodes as well as hyperedges can be labeled with the use of a fixed alphabet. Additionally, the sets of attributes can be assigned to nodes and hyperedges. Figure 1 presents an exemplary hypergraph consisting of five nodes labeled by v, two hyperedges labeled by I and one hyperedge labeled by $F1$. The hyperedges labeled by I have attribute *break*. One hyperedge has attribute *break* equal to 1 and the second one has the attribute *break* equal to 0.

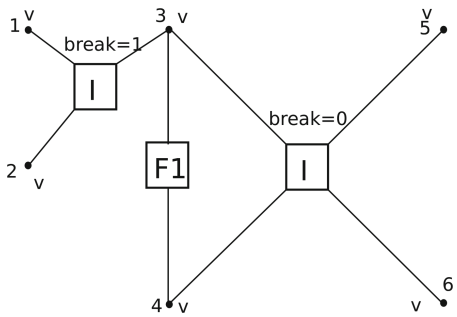

Fig. 1. An exemplary hypergraph with six nodes denoted by v, two hyperedges denoted by I and one hyperedge denoted by $F1$.

Definition 1. An undirected attributed labelled hypergraph over label alphabet C and attribute set A is defined as a system $G = (V, E, t, l, at)$, where:

- V is a finite set of nodes,
- E is a finite set of hyperedges,
- $t : E \to V^*$ is a mapping assigning sequences of target nodes to hyperedges,
- $l : V \cup E \to C$ is a node and hyperedge labelling function,
- $at : V \cup E \to 2^A$ is a node and hyperedge attributing function.

Hypergraph G is a subhypergraph of hypergraph g if its sets of nodes and edges are subsets of sets of nodes and edges of the hypergraph g, respectively

and the corresponding nodes and edges of both graphs have the same labels and attributes.

Definition 2. Let $G = (V_G, E_G, t_G, l_G, at_G)$ and $g = (V_g, E_g, t_g, l_g, at_g)$ be two hypergraphs. G is a subhypergraph of g if:

1. $V_G \subset V_g, E_G \subset E_g$,
2. $\forall e \in E_G \; t_G(e) = t_g(e)$,
3. $\forall e \in E_G \; l_G(e) = l_g(e), \forall v \in V_G \; l_G(v) = l_g(v)$,
4. $\forall e \in E_G \; at_G(e) = at_g(e), \forall v \in V_G \; at_G(v) = at_g(v)$.

In order to derive complex hypergraphs from the simpler ones the so-called graph grammar productions can be used. This approach allows to replace subhypergraph of the hypergraph by new hypergraph. The operation of replacing the subhypergraph of the hypergraph by another hypergraph is allowed only under the assumption, that for each new hypergraph so called sequence of its external nodes is specified and that both hypergraphs have the same number of external nodes (the same so-called type of a graph). These nodes correspond to target nodes of a replaced hypergraph.

Definition 3. A hypergraph of type k is a system $H = (G, ext)$, where:

- $G = (V, E, t, l, at)$ is a hypergraph over C and A,
- ext is a sequence of nodes from the node set V, called external nodes, with $|ext| = k$.

Figure 1 presents an exemplary hypergraph of type 6 with denoted external nodes.

Definition 4. A hypergraph production is a pair $p = (L, R)$, where both L and R are hypergraphs of the same type. Applying a production $p = (L, R)$ to a hypergraph H means to replace a subhypergraph h of H isomorphic with L by a hypergraph R and replacing external nodes from graph h with the corresponding external nodes of R.

Figure 6 presents an exemplary hypergraph production. Figure 2 presents an exemplary starting graph and Fig. 4 presents this graph after application of the production from Fig. 3.

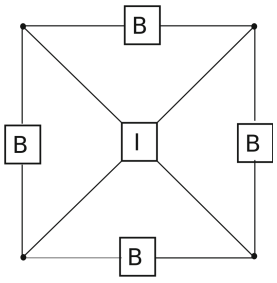

Fig. 2. An exemplary starting hypergraph.

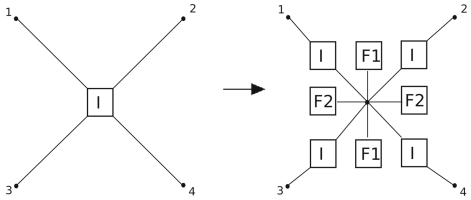

Fig. 3. An exemplary hypergraph production.

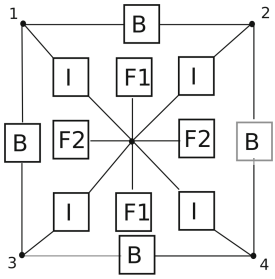

Fig. 4. The graph from Fig. 2 after application of the production from Fig. 3

The application of a production $p = (L, R)$ to a hypergraph H consists of replacing a subhypergraph of H isomorphic with L by a hypergraph isomorphic with R and replacing nodes of the removed subhypergraph isomorphic with external nodes of L by the corresponding external nodes of R.

Definition 5. Let P be a fixed set of hypergraph productions. Let G and G' be two hypergraphs.

We say that hypergraph H is directly derived from a hypergraph G ($G \Rightarrow H$) if there exists $p = (L, R) \in P$ such that:

- There exists h - a subhypergraph of G isomorphic with L,
- Let ext_h be a sequence of nodes of h consisting of nodes isomorphic with nodes of the sequence of external nodes of L (ext_L). The replacement of $h = (V_h, E_h, t_h, l_h, at_h)$ in $G = (V_G, E_G, t_G, l_G, at_G)$ by $R = (V_R, E_R, t_R, l_R, at_R)$ generates the the hypergraph $G' = (V'_G, E'_G, t'_G, l'_G, at'_G)$, where:

  - $V'_G = V_H - V_h \cup V_R$,
  - $E'_G = E_H - E_h \cup E_R$,
  - $\forall e \in E_R\ t'_G(e) = t_R(e)$,
  - $\forall e \in E_G - E_h$ with $t_G(e) = t_1, ..., t_n, t'_G(e) = t'_1, .., t'_n$, where each $t'_i = t_i$ if t_i does not belong to the sequence ext_h or $t'_i = v_j$ (v_j is an j-th element of the sequence ext_R) if t_i is an j-th element of the sequence ext_h,
  - $\forall e \in E_G - E_h\ l'_G(e) = l_G(e)$, $at'_G(e) = at_G(e)$, $\forall e \in E_R\ l'_G(e) = l_R(e)$, $at'_G(e) = at_R(e)$,

- $\forall v \in V_G - V_h \; l'_G(v) = l_G(v)$, $at'_G(v) = at_G(v)$, $\forall v \in V_R \; l'_G(v) = l_R(v)$, $at'_G(v) = at_R(v)$.

– H is isomorphic with the hypergraph G'.

Definition 6. A hypergraph grammar is a quadruple $G = (V, E, P, X)$, where:

– V is a finite set of labelled nodes,
– E is a finite set of labelled hyperedges,
– P is a finite set of hypergraph productions of the form $p = (L, R)$, where L and R are hypergraphs of the same type composed of nodes of V and hyperedges of E,
– X is an initial hypergraph called axiom of G.

3 Hypergraph Grammar for Bitmap Compression

In the section the hypergraph grammar productions modelling the bitmap compression based on projection based interpolation algorithm is presented. The finite element mesh is represented by hypergraph. The changes in the structure of the mesh are modeled by corresponding graph grammar productions.
A hypergraph modeling a mesh with rectangular elements is defined with the following sets of graph node and edge labels C:

$$C = \{C_1 \cup C_2 \cup C_3\} \tag{1}$$

where

– $C_1 = \{v\}$ is a singleton containing the node label which denotes a node of finite element,
– $C_2 = \{F1, F2, B\}$ is a set of edge labels which denote edges of finite elements,
– $C_3 = \{I\}$ is a singleton containing the edge label which denotes interior of finite element.

The following attributing functions are defined to attribute nodes and edges of the hypergraphs modeling the mesh.

– $geom : V \times C_1 \rightarrow R \times R$ is a function attributing nodes, which assigns the coordinates x, y to each node of the element,
– $break : E \times C_3 \rightarrow \{0, 1\}$ is a function attributing interiors, which assigns the value denoting if the adaptation should be performed, where 1 means that element should be broken, 0 means that element should not be broken.

The graph grammar for bitmap compression **(G1)** is defined as a set of hypergraph grammar productions: production **(P1)** for generation of the first element of the mesh, productions **(P2)**, **(P2')**, **(P2")**, **(P2"')** breaking the interior of the element, production **(P3)** for breaking of the boundary edge, production **(P4)** for breaking shared edge, productions **(P5)**, **(P5')**, **(P5")**, **(P5"')** for

making the decision about the adaptation, and the starting graph consisting of one node labeled by S. Figure 5 presents production **(P1)** for generation of the first element. Productions **(P2)**, **(P2')**, **(P2")** and **(P2"')** from Fig. 6 allow for breaking the interior of the element for the case of all adjacent edges having the same "level of adaptation" as the interior (production **(P2)**), three adjacent edges having the same "level of adaptation" as the interior (**(P2')**), two adjacent edges having the same "level of adaptation" as the interior (production **(P2")**) and finally only one adjacent edge having the same "level of adaptation" as the interior (production **(P2"')**). Our model distinguishes the edges of element being in the boundary of the mesh (hyperdees with label B), and shared edges of elements (labeled by $F1$ and $F2$). Thus we have two separate productions for breaking edges: production **(P3)** for breaking boundary edges and production **(P4)** for breaking shared edges (see Fig. 7, 8). The similar productions to productions from Fig. 7, 8 are defined for hyperedges labeled by $F2$ instead of $F1$ and $F1$ instead of $F2$. The last set of productions, presented in Fig. 9, allows for making decision about the adaptation. Figure 10 presents an exemplary derivation in our graph grammar.

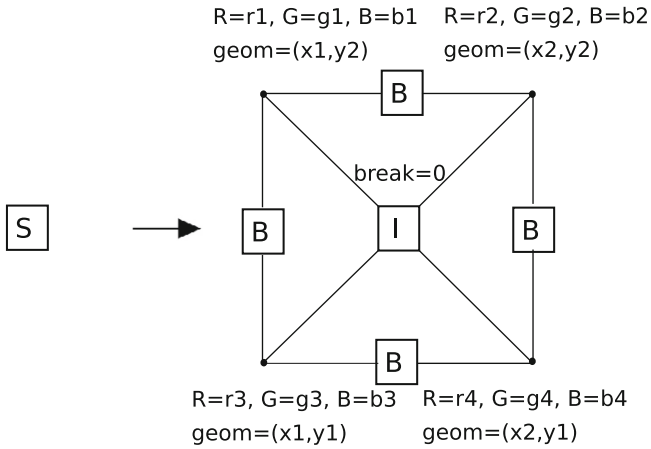

Fig. 5. Production P1 for generation of the first element

4 Results

In order to examine the validity of the idea of the presented hypergraph grammar-based Adaptive JPEG algorithm, we compare the compression ratio with the standard JPEG algorithm on three exemplary bitmaps, presented in Fig. 11. Compressed bitmaps with only luminance component's MCUs outlined in the images are presented in Fig. 12.

The Table 2 presents the compression ratio for different luminance and chrominance parameter values.

The main observation from looking at the Table 2 is that the algorithm (without taking into account visual quality) indeed gives higher compression ratios than baseline JPEG, with higher compression obtained from running the algorithm with larger quality parameter values. This is hardly a surprise, as theoretically the algorithm could give the same compression ratio as baseline

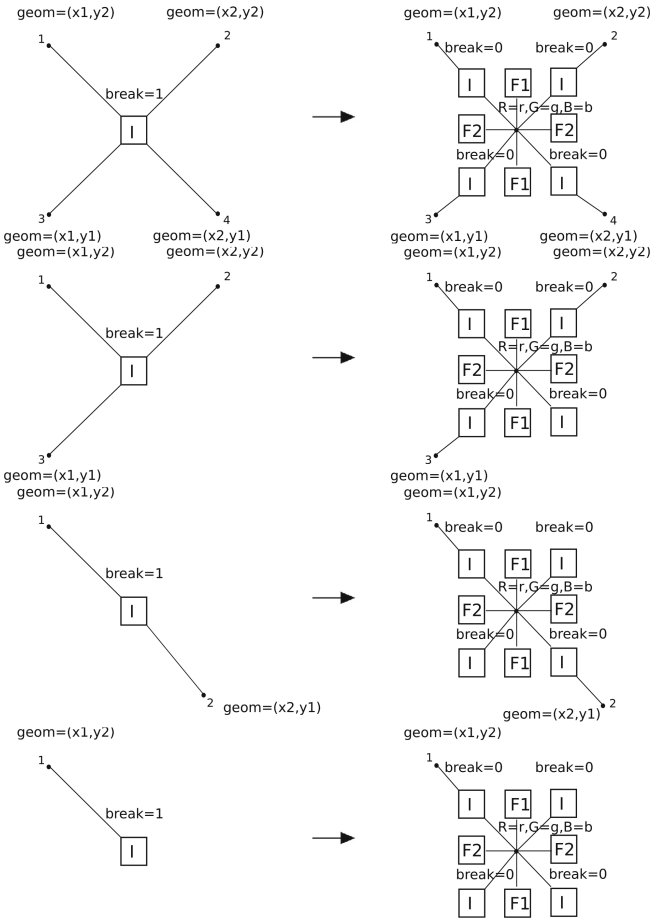

Fig. 6. Productions P2, P2', P2" and P2"' for breaking the interior of the element.

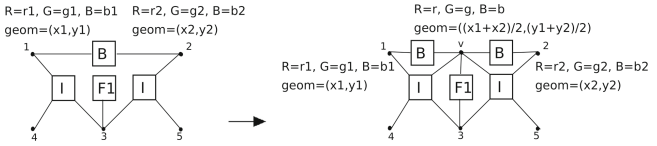

Fig. 7. Production P3 for breaking the boundary edge

Fig. 8. Production P4 for breaking common edge

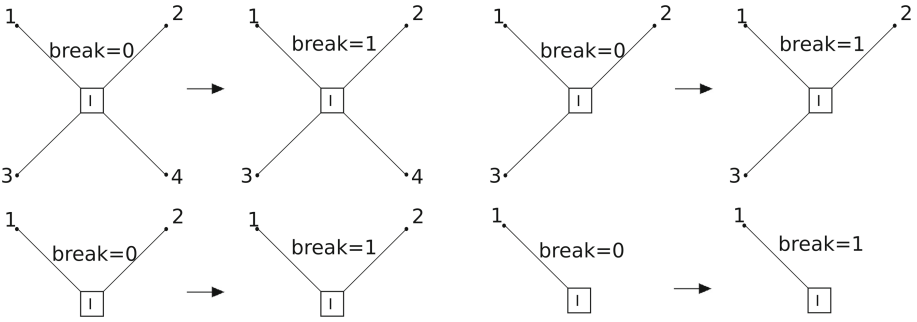

Fig. 9. Productions P5, P5', P5" and P5"' for making virtual adaptation

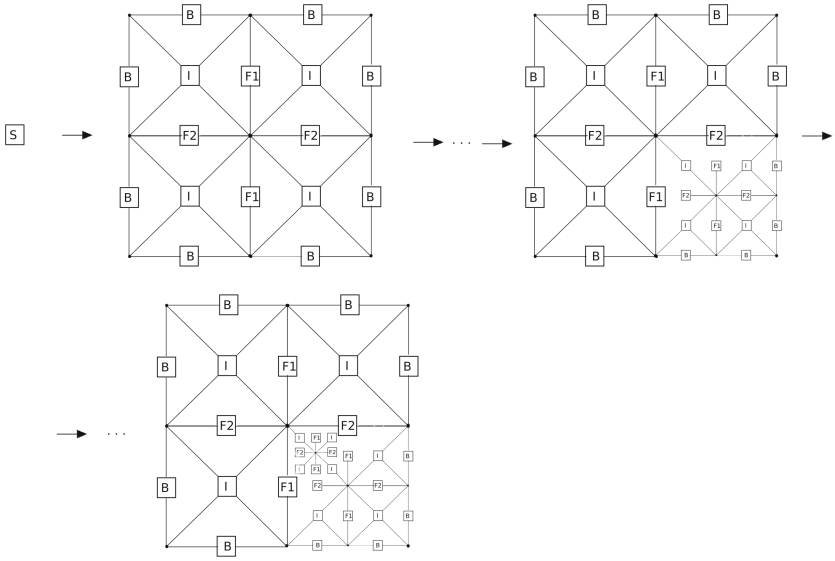

Fig. 10. Exemplary derivation

JPEG in a worst-case scenario, but usually the non-zero quality parameter values leave some room for error and allow some larger areas to stay undivided, resulting in greater compression.

Another observation from examining the graphs more carefully, is that the compression ratios for different test images are quite sparse: depending on the image, for a satisfactory quality, the compression ratios came down to as little as 0.69% (field image for parameter values 15×30) or to only as much as 8.5% (moon image for parameter values 6×60). This shows, that the algorithm clearly works better for certain kind of images than for others. This kind of images are very large images with large relatively uniform areas.

In order to examine the validity of the idea of the presented hypergraph grammar-based Adaptive JPEG algorithm, an effort to compare the performance with the standard JPEG and newer, JPEG successor standards was made: JPEG 2000 and JPEG XR. However, such comparison is not trivial, as the algorithms use different types of parameters to specify the level of compression, directly reflected in the resulting image quality. The parameter types used in JPEG2000 and JPEG XR do not necessarily coincide with the quality parameter type used in JPEG, that is 1–100 scaled quality parameter used to modify the quantization tables. Therefore, some assumptions were made when choosing the parameter values for JPEG2000 and JPEG XR. The JPEG2000 images were generated using the OpenJPEG library, which accepts a quality parameter denoting a target PSNR value in dB. As it was not possible to use the same type of parameter as JPEG accepts, the JPEG2000 images were compressed using PSNR value parameter equal to the PSNR of the corresponding baseline JPEG image. The JPEG XR images were generated using the jxrlib library, which, as a quality parameter, accepts a value ranging between 0.0 and 1.0. In this case, the images were compressed using the parameter value 0.8, to be as consistent with the JPEG compressed test images as possible. The actual impact of the respective parameter values however might differ considerably from JPEG in terms of resulting image quality to file size ratio. The presented Adaptive JPEG seems to work particularly well with large, relatively uniform images - as evident in the case of the field image (Table 1).

Table 1. Comparison of compression ratios for different luminance and chrominance parameters for JPEG and Adaptive JPEG.

Bitmap	JPEG	2×4	5×5	5×10	10×20	15×15	15×30	20×40	25×50	30×30	60×60
Field	3.19%	2.58%	2.26%	2.16 %	1.17%	0.72%	0.69%	0.46%	0.34%	0.28%	0.13%
Lake	5.68%	5.06%	4.53%	3.92%	3.35%	2.70%	2.62%	1.93%	1.55%	1.29%	0.54%
Moon	11.83%	11.24%	11.24%	11.24%	11.24 %	11.24%	11.24%	11.24%	11.24%	11.16%	8.50%

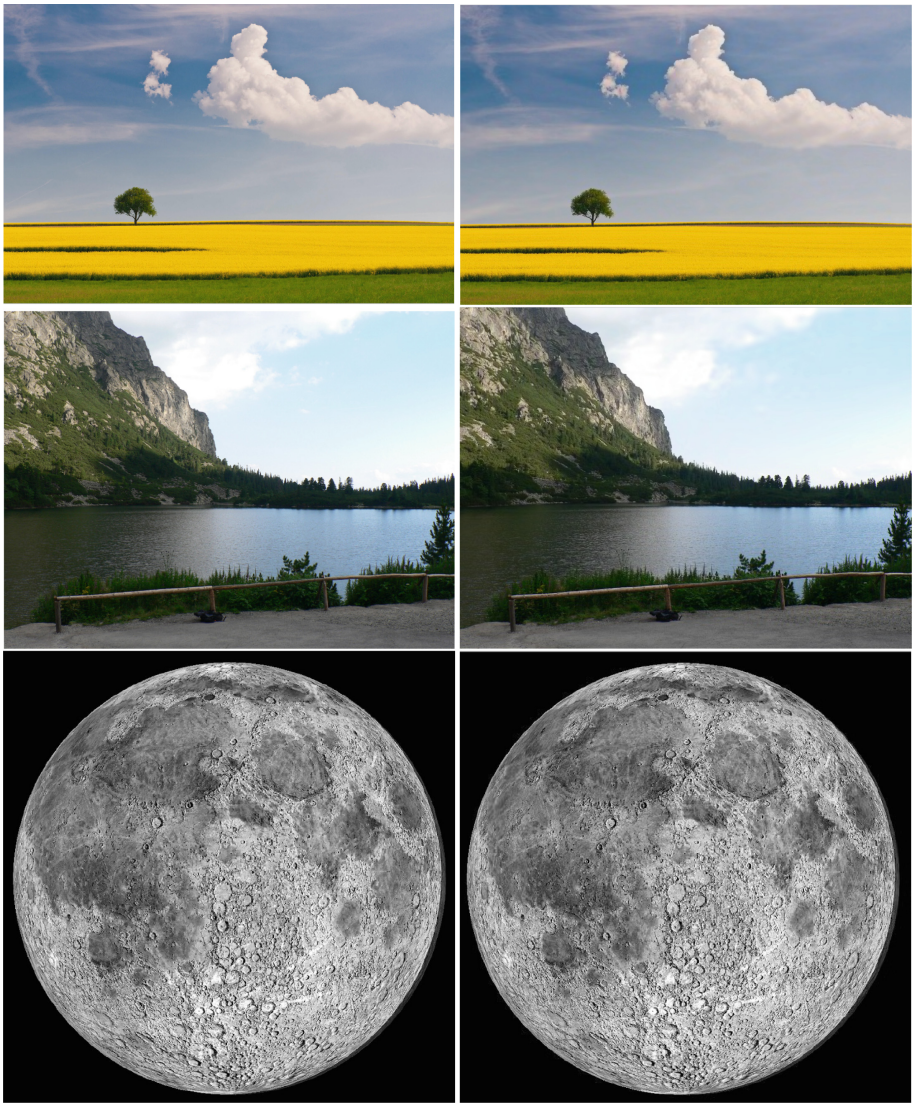

Fig. 11. Bitmap compressions. JPEG compresson on the left, adaptive JPEG on the right with error tolerance parameter values - 15/30.

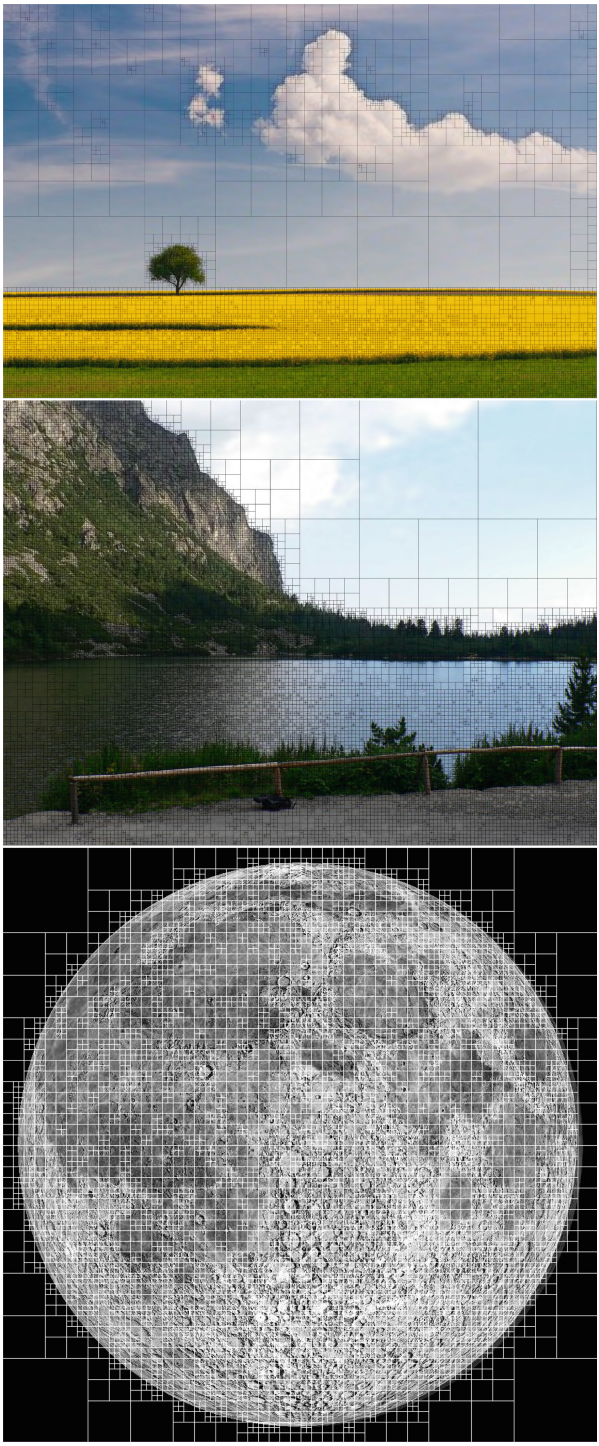

Fig. 12. Compressed bitmaps with only luminance component's MCUs outlined.

Table 2. Comparison of compression ratios for different image standards.

Bitmap	ADAPTIVE JPEG	JPEG-XR	JPEG2000	JPEG
Field	0.72%	3.33%	2.60%	2.29%
Lake	2.70%	6.37%	1.47%	5.68%
Moon	11.24%	16.23%	5.94%	11.83%

5 Conclusions

There seems to be a class of images for which the algorithm presented in this paper achieves higher compression ratios than baseline JPEG, while retaining subjective image quality. Even though this class of images is not clearly defined, some common characteristics can be extracted: large size of the image - images with higher resolution seem to gain more from the adaptive extension introduced by the algorithm, and areas of low frequency - images containing areas that are relatively uniform seem to tolerate larger MCU sizes better, resulting in higher compression ratios. The maximum values of luminance and chrominance quality parameters, consumed by the algorithm, for which the images seemed to retain subjective image quality, were found to be around 15/15 (luminance/chrominance). In some cases, the image quality was still retained for the parameter values of 15/30, proving the less impactful nature of the chrominance component. Future work may include implementing the adaptive algorithm based on an existing improvement over JPEG, e.g. JPEG2000. Both JPEG2000 and JPEG XR, although very different, operate much the same in regard to MCU size, which is globally fixed. If not an implementation, a thorough performance comparison between the presented Adaptive JPEG algorithm and other lossy compression algorithms ought to be conducted, as only JPEG was exhaustively addressed in this work. Finally, we only address the problem of compression, completely leaving out the issue of decompression. Decompression of an image compressed in such non-deterministic way is a very complex problem, deserving a separate research.

Acknowledgement. The research presented in this paper was partially supported by the funds assigned to AGH University of Science and Technology by the Polish Ministry of Science and Higher Education.

References

1. International Organization for Standardization. ISO/IEC 15444–1:2016 - Information technology - JPEG 2000 image coding system: Core coding system (2016)
2. Askelöf, J., Carlander, M.L., Christopoulos, C.: Region of interest coding in jpeg 2000. Signal Process. Image Commun. **17**(1), 105–111 (2002)
3. Ebrahimi, F., Chamik, M., Winkler, S.: Jpeg vs. jpeg2000: an objective comparison of image encoding quality, pp. 55–58 (2004)

4. International Telecommunication Union. ITU-T T.832 - Information technology - JPEG XR image coding system - Image coding specification (2016)
5. Dufaux, F., Sullivan, G.J., Ebrahimi, T.: The jpeg xr image coding standard [standards in a nutshell]. IEEE Signal Process. Mag. **26**(6), 195–204 (2009)
6. Artusi, A., et al.: Jpeg xt: a compression standard for hdr and wcg images [standards in a nutshell]. IEEE Signal Process. Mag. **33**(2), 118–124 (2016)
7. Richter, T., Keinert, J., Descampe, A., Rouvroy, G.: Entropy coding and entropy coding improvements of jpeg xs. In: 2018 Data Compression Conference, pp. 87–96 (2018)
8. Habel, A., Kreowski, H.-J.: May we introduce to you: hyperedge replacement. In: Ehrig, H., Nagl, M., Rozenberg, G., Rosenfeld, A. (eds.) Graph Grammars 1986. LNCS, vol. 291, pp. 15–26. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-18771-5_41
9. Habel, A., Kreowski, H.-J.: Some structural aspects of hypergraph languages generated by hyperedge replacement. In: Brandenburg, F.J., Vidal-Naquet, G., Wirsing, M. (eds.) STACS 1987. LNCS, vol. 247, pp. 207–219. Springer, Heidelberg (1987). https://doi.org/10.1007/BFb0039608
10. Slusarczyk, G., Paszyńska, A.: Hypergraph grammars in hp-adaptive finite element method. Procedia Comput. Sci. **18**, 1545–1554 (2013)
11. Gurgul, P., Jopek, K., Pingali, K., Paszyńska, A.: Applications of a hyper-graph grammar system in adaptive finite-element computations. Int. J. Appl. Math. Comput. Sci. **28**(3), 569–582 (2018)
12. Gurgul, P., Paszyński, M., Paszyńska, A.: Hypergrammar based parallel multi-frontal solver for grids with point singularities. Comput. Sci. **16**(1), 75–102 (2015)
13. Paszyński, M., Jopek, K., Paszyńska, A., Hassan, M.A., Pingali, K.: Hypergraph grammar based multi-thread multi-frontal direct solver with Galois scheduler. Comput. Sci. **18**(2), 27–55 (2019)
14. Paszyńska, A., et al.: Quasi-optimal elimination trees for 2d grids with singularities. Sci. Program. (2015)
15. Paszyńska, A., Paszyński, M., Grabska, E.: Graph Transformations for modeling hp-adaptive finite element method with mixed triangular and rectangular elements. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009. LNCS, vol. 5545, pp. 875–884. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01973-9_97
16. Goik, D., Jopek, K., Paszyński, M., Lenharth, A., Nguyen, D., Pingali, K.: Graph grammar based multi-thread multi-frontal direct solver with Galois scheduler. Procedia Comput. Sci. **29**, 960–969 (2014)
17. Paszyński, M., Paszyńska, A.: Graph transformations for modeling parallel hp-adaptive finite element method. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 1313–1322. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68111-3_139
18. Paszyński, M.: On the parallelization of self-adaptive hp-finite element methods part I. composite programmable graph grammarmodel. Fundam. Inf. **93**(4), 411–434 (2009)
19. Paszyński, M.: On the parallelization of self-adaptive hp-finite element methods part II partitioning communication agglomeration mapping (PCAM) analysis. Fundam. Inf. **93**(4), 435–457 (2009)