Entropy Coding-based Lossless Compression of Asynchronous Event Sequences —Supplementary Material—

Ionut Schiopu and Radu Ciprian Bilcu
Huawei Technologies Oy (Finland) Co. Ltd, Tampere Handset Camera Innovation Lab

Algorithm E1: Modified TTP_e Encoder of N_e

```
Input: $N_e, \hat{N}_e, \mathbf{\Delta}_e = (\delta_1, \delta_2)$;
if $N_e > 0$ then
    Encode $e_n = 0$ using $DTeb_0$; Update $DTeb_0$;
    $\epsilon_e = N_e - \hat{N}_e$; $\Delta_e = \delta_1 + \delta_2$;
    if $|\epsilon_e| < \delta_1$ then                        // R1 Range
        Encode $e_n = 0$ using $DTeb_1$; Update $DTeb_1$;
        Encode $e_n = |\epsilon_e|$ using $Ee\delta_1\ell$; Update $Ee\delta_1\ell$;
    else
        Encode $e_n = 1$ using $DTeb_1$; Update $DTeb_1$;
        if $|\epsilon| < \Delta_e - 1$ then                  // R2 Range
            Encode $|\epsilon_e| - \delta_1$ using $Ee\delta_2\ell$; Update $Ee\delta_2\ell$;
        else                                                  // R6 Range
            Encode $\delta_2 - 1$ using $Ee\delta_2\ell$; Update $Ee\delta_2\ell$;
            Encode $x_\gamma = |\epsilon_e| - \Delta_e - 2$ using Alg. E2;
    Encode $e_n = sgn(|\epsilon|)$ using $DTeb_s$; Update $DTeb_s$;
else
    Encode $e_n = 1$ using $DTeb_0$; Update $DTeb_0$;
```

Algorithm E2: Modified EGC Encoder of x_γ

```
Input: $x_\gamma$
if $x_\gamma = 1$ then
    Encode $x_n = 1$ using $BR4$; Update $BR4$ using (1);
else
    $N = \lfloor log_2(x_\gamma) \rfloor$ ;   // $(x_\gamma)_{(10)} = \overline{1b_{N-1} \dots b_1 b_0}_{(2)}$
    for $i = 1, 2, \dots, N$ do          // Unary rep. of $N$
        Encode $x_n = 0$ using $BR4$; Update $BR4$;
    Encode $x_n = 1$ using $BR4$; Update $BR4$;
    Binarize $(x_\gamma)_{(10)}$ as $\overline{1b_{N-1} \dots b_1 b_0}_{(2)}$;
    for $i = N-1, N-2, \dots, 0$ do  // Last $N$ digits
        Encode $x_n = b_i$ using $BR5$; Update $BR5$;
```

Algorithm E3: Modified TTP_y Encoder of y

```
Input: $y, \hat{y}, [1, W], \mathbf{\Delta} = (\delta_1, \delta_2, \delta_3)$;
$\epsilon_y = y - \hat{y}$; $\Delta_y = \delta_1 + \delta_2 + \delta_3$;
if $|\epsilon_y| < \Delta_y$ then
    Encode $y_n = 0$ using $DTyb_0$; Update $DTyb_0$;
    if $|\epsilon_y| < \delta_1$ then                        // R1 Range
        Encode $y_n = 0$ using $DTyb_1$; Update $DTyb_1$;
        Encode $y_n = |\epsilon_y|$ using $Ey\delta_1\ell$; Update $Ey\delta_1\ell$;
    else
        Encode $y_n = 1$ using $DTyb_1$; Update $DTyb_1$;
        if $|\epsilon_y| < \delta_1 + \delta_2$ then             // R2 Range
            Encode $y_n = 0$ using $DTyb_2$; Update $DTyb_2$;
            Encode $|\epsilon_y| - \delta_1$ using $Ey\delta_2\ell$; Update $Ey\delta_2\ell$;
        else                                                  // R3 Range
            Encode $y_n = 1$ using $DTyb_2$; Update $DTyb_2$;
            Encode $|\epsilon_y|$-$\delta_1$-$\delta_2$ using $Ey\delta_3\ell$; Update $Ey\delta_3\ell$;
else                                                          // R5 Range
    Encode $y_n = 1$ using $DTyb_0$; Update $DTyb_0$;
    $y_2 = \hat{y} + \Delta_y; n_2 = \lceil \log_2 (W - y_2 + 1) \rceil$;
    Binarize $(W - y)_{(10)}$ as $\overline{b_{n_2-1} \dots b_1 b_0}_{(2)}$;
    for $i = 0, 1, \dots, n_2 - 2$ do
        Encode $y_{n+i} = b_i$ using $ByL$; Update $ByL$;
    if $\overline{b_{n_2-1} \dots b_1 b_0} \le W - y_2 + 1 - 2^{n_2-1}$ then
        Encode $x_n = b_{n_2-1}$ using $BR5$; Update $BR5$;
```

Algorithm E4: Modified TTP_x Encoder of x

```
Input: $x, \hat{x}, c_x$ (to signal sorted $x$), $[1,\ H]$, $\mathbf{\Delta} = (\delta_1, \delta_2, \delta_3)$;
$\epsilon_x = x - \hat{x}$; $\Delta_x = \delta_1 + \delta_2 + \delta_3$;
if $|\epsilon_x| < \Delta_x$ then
    Encode $x_n = 0$ using $DTxb_0$; Update $DTxb_0$;
    if $|\epsilon_x| < \delta_1$ then                                  // R1 Range
        Encode $x_n = 0$ using $DTxb_1$; Update $DTxb_1$;
        Encode $x_n = |\epsilon_x|$ using $Ex\delta_1\ell$; Update $Ex\delta_1\ell$;
    else
        Encode $x_n = 1$ using $DTxb_1$; Update $DTxb_1$;
        if $|\epsilon_x| < \delta_1 + \delta_2$ then                         // R2 Range
            Encode $x_n = 0$ using $DTxb_2$; Update $DTxb_2$;
            Encode $|\epsilon_x| - \delta_1$ using $Ex\delta_2\ell$; Update $Ex\delta_2\ell$;
        else                                                      // R3 Range
            Encode $x_n = 1$ using $DTxb_2$; Update $DTxb_2$;
            Encode $|\epsilon_x|$-$\delta_1$-$\delta_2$ using $Ex\delta_3\ell$; Update $Ex\delta_3\ell$;
    if $c_x$ then                                         // Was $x$ sorted?
        Encode $sgn(|\epsilon_x|)$ using $DTxb_s$; Update $DTxb_s$;
else
    Encode $x_n = 1$ using $DTxb_0$; Update $DTxb_0$;
    $x_1 = \hat{x} - \Delta_x; n_1 = \lceil \log_2 x_1 \rceil$;
    $x_2 = \hat{x} + \Delta_x; n_2 = \lceil \log_2 (H - x_2 + 1) \rceil$;
    if $x_1 < 1$ then                 // Deterministic case R5
        Binarize $(H - x)_{(10)}$ as $\overline{b_{n_2-1} \dots b_1 b_0}_{(2)}$;
        for $i = 0, 1, \dots, n_2 - 2$ do
            Encode $x_{n+i} = b_i$ using $BxL$; Update $BxL$;
        if $\overline{b_{n_2-1} \dots b_1 b_0} \leq H - x_2 + 1 - 2^{n_2-1}$ then
            Encode $x_n = b_{n_2-1}$ using $BR5$; Update $BR5$;
    else if $x_2 > H$ then            // Deterministic case R4
        Binarize $(x - 1)_{(10)}$ as $\overline{b_{n_1-1} \dots b_1 b_0}_{(2)}$;
        for $i = 0, 1, \dots, n_1 - 2$ do
            Encode $x_{n+i} = b_i$ using $BxL$; Update $BxL$;
        if $\overline{b_{n_1-1} \dots b_1 b_0} \leq x_1 - 2^{n_1-1}$ then
            Encode $x_n = b_{n_1-1}$ using $BR4$; Update $BR4$;
    else
        if $x \leq \hat{x} - \Delta_x$, then                            // R4 Range
            Encode $x_n = 0$ using $DTxb_1^d$; Update $DTxb_1^d$;
            Binarize $(x - 1)_{(10)}$ as $\overline{b_{n_1-1} \dots b_1 b_0}_{(2)}$;
            for $i = 0, 1, \dots, n_1 - 2$ do
                Encode $b_i$ using $BxL$; Update $BxL$;
            if $\overline{b_{n_1-1} \dots b_1 b_0} \leq x_1 - 2^{n_1-1}$ then
                Encode $b_{n_1-1}$ using $BR4$; Update $BR4$;
        else                                                      // R5 Range
            Encode $x_n = 1$ using $DTxb_1^d$; Update $DTxb_1^d$;
            Binarize $(H - x)_{(10)}$ as $\overline{b_{n_2-1} \dots b_1 b_0}_{(2)}$;
            for $i = 0, 1, \dots, n_2 - 2$ do
                Encode $b_i$ using $BxL$; Update $BxL$;
            if $\overline{b_{n_2-1} \dots b_1 b_0} \leq H - x_2 + 1 - 2^{n_2-1}$ then
                Encode $b_{n_2-1}$ using $BR5$; Update $BR5$;
```

Algorithm D1: Modified TTP_x Decoder of x

```
Input: $\hat{x}, [1,\ H], c_x, \mathbf{\Delta} = (\delta_1, \delta_2, \delta_3)$;
Output: $x$;
$b_0 \leftarrow$ Decode using $DTxb_0$; Update $DTxb_0$;
if $b_0 = 0$ then
    $b_1 \leftarrow$ Decode using $DTxb_1$; Update $DTxb_1$;
    if $b_1 = 0$ then $x \leftarrow$ Decode using $Ex\delta_1\ell$; Update $Ex\delta_1\ell$ ;
    else
        $b_2 \leftarrow$ Decode using $DTxb_2$; Update $DTxb_2$;
        if $b_2 = 0$ then                                  // R2 Range
            $x = \delta_1 +$ Decode using $Ex\delta_2\ell$; Update $Ex\delta_2\ell$;
        else $x = \delta_1 + \delta_2 +$Decode using $Ex\delta_3\ell$; Update $Ex\delta_3\ell$;
    $b_s = 1$;
    if $c_x$ then $b_s \leftarrow$ Decode using $DTxb_s$; Update $DTxb_s$ ;
    $x \leftarrow (b_s = 0)?(\hat{x} - x) : (\hat{x} + x)$;
else
    $x = 0$; $x_1 = \hat{x} - \delta_1 - \delta_2 - \delta_3$; $x_2 = \hat{x} + \delta_1 + \delta_2 + \delta_3$;
    $n_1 = \lceil \log_2 x_1 \rceil$; $n_2 = \lceil \log_2 (H - x_2 + 1) \rceil$;
    if $x_1 < 1$ then             // Deterministic case R5
        for $i = 0, 1, \dots, n_2 - 2$ do
            $b \leftarrow$ Decode using $BxL$; Update $BxL$;
            if $b = 1$ then $x = (1 \ll i)|x$;
        if $x \leq H - x_2 + 1 - 2^{n_2-1}$ then
            $b \leftarrow$ Decode using $BR5$; Update $BR5$;
            if $b = 1$ then $x = (1 \ll (n_2 - 1))|x$ ;
    else if $x_2 > H$ then        // Deterministic case R4
        for $i = 0, 1, \dots, n_1 - 2$ do
            $b \leftarrow$ Decode using $BxL$; Update $BxL$;
            if $b = 1$ then $x = (1 \ll i)|x$ ;
        if $x \leq x_1 - 2^{n_1-1}$ then
            $b \leftarrow$ Decode using $BR4$; Update $BR4$;
            if $b = 1$ then $x = (1 \ll (n_1 - 1))|x$ ;
    else
        $b_1^d \leftarrow$ Decode using $DTxb_1^d$; Update $DTxb_1^d$;
        if $b_1^d = 0$ then                               // R4 Range
            for $i = 0, 1, \dots, n_1 - 2$ do
                $b \leftarrow$ Decode using $BxL$; Update $BxL$;
                if $b = 1$ then $x = (1 \ll i)|x$ ;
            if $x \leq x_1 - 2^{n_1-1}$ then
                $b \leftarrow$ Decode using $BR4$; Update $BR4$;
                if $b = 1$ then $x = (1 \ll (n_1 - 1))|x$ ;
        else                                              // R5 Range
            for $i = 0, 1, \dots, n_2 - 2$ do
                $b \leftarrow$ Decode using $BxL$; Update $BxL$;
                if $b = 1$ then $x = (1 \ll i)|x$ ;
            if $x \leq H - x_2 + 1 - 2^{n_2-1}$ then
                $b \leftarrow$ Decode using $BR5$; Update $BR5$;
                if $b = 1$ then $x = (1 \ll (n_2 - 1))|x$ ;
Return $x$;
```

Algorithm D2: Modified TTP_e Decoder of N_e

```
Input: $\hat{N}_e, \mathbf{\Delta} = (\delta_1, \delta_2)$;
Output: $N_e$;
$b_0$ ←Decode using $DTeb_0$; Update $DTeb_0$;
if $b_0 = 0$ then
    $b_1$ ← Decode using $DTeb_1$; Update $DTeb_1$;
    if $b_1 = 0$ then $e \leftarrow$ Decode using $Ee\delta_1\ell$; Update $Ee\delta_1\ell$ ;
    else
        $e \leftarrow$ Decode using $DTeb_2$; Update $DTeb_2$;
        if $e = \delta_2 - 1$ then
            $e \leftarrow \delta_1 + \delta_2 - 2 +$ Decode using Alg. 3;
        else $e \leftarrow \delta_1 + e$ ;
    $b_s \leftarrow$ Decode using $DTeb_s$; Update $DTeb_s$;
    Return $N_e = (b_s = 0)?(\hat{N}_e - e) : (\hat{N}_e + e)$;
else
    Return $N_e = 0$
```

Algorithm D3: Modified EGC Decoder of x_γ

```
Output: $x_\gamma$;
$x_\gamma$ ←Decode using $BR4$; Update $BR4$;
if $x_\gamma = 0$ then
    $N = 1$; $x_\gamma$ ←Decode using $BR4$; Update $BR4$;
    while $x = 0$ do
        $N = N + 1$; $x_\gamma$ ←Decode using $BR4$; Update $BR4$;
    for $i = N - 1, N - 2, \ldots, 0$ do
        $x_\gamma = (x_\gamma \ll 1)$+Decode using $BR5$; Update $BR5$;
Return $x_\gamma$;
```

Algorithm D4: Modified TTP_y Decoder of y

```
Input: $\hat{y}$, $[1,\, W]$, $\mathbf{\Delta} = (\delta_1, \delta_2, \delta_3)$;
Output: $y$;
$b_0$ ← Decode using $DTyb_0$; Update $DTyb_0$ using (1);
if $b_0 = 0$ then
    $b_1$ ← Decode using $DTyb_1$; Update $DTyb_1$;
    if $b_1 = 0$ then                                  // R1 Range
        $y = \hat{y} +$ Decode using $Ey\delta_1\ell$; Update $Ey\delta_1\ell$;
    else
        $b_2$ ← Decode using $DTyb_2$; Update $DTyb_2$;
        $y \leftarrow \hat{y} + \delta_1$;
        if $b_2 = 0$ then                              // R2 Range
            $y = y$+Decode using $Ey\delta_2\ell$; Update $Ey\delta_2\ell$;
        else                                           // R3 Range
            $y = y$+$\delta_2$+Decode using $Ey\delta_3\ell$; Update $Ey\delta_3\ell$;
else                                                   // R5 Range
    $y = 0; n_2 = \lceil \log_2 (W - \hat{y} - \delta_1 - \delta_2 - \delta_3 + 1) \rceil$;
    for $i = 0, 1, \ldots, n_2 - 2$ do
        $b$ ← Decode using $ByL$; Update $ByL$;
        if $b = 1$ then $y = (1 \ll i)|y$ ;
    if $y \leq W - \hat{y} - \delta_1 - \delta_2 - \delta_3 + 1 - 2^{n_2 - 1}$ then
        $b$ ← Decode using $BR5$; Update $BR5$;
        if $b = 1$ then $y = (1 \ll (n_2 - 1))|y$ ;
Return $y$;
```