Robust implementation of LT Codes.

by
Anurag Paul
20EC01045

Under the supervision of
Dr. Siddhartha S. Borkotoky

INTRODUCTION

1. Luby Transform (LT) codes are first realization of a class of erasure codes.
2. Analyse the run time of the encoder and decoder in terms of symbol operations.
3. Symbol operation = $S_i \oplus S_j$.
4. Rate–less.
5. LT codes are near optimal with respect to any erasure channel.
6. Encoding/decoding times are efficient as a function of the data length.

Input:

```
ad = '/MATLAB Drive/O/L out/b.jpg';
% file path of the file
r = 2;          % the wanted redundancy or rep
SYC = 1;        % SYSTEMATIC LT Codes, T/F
VSE = 1;        % VERBOSE, increase output verbosity
P = 2^16;       % PACKET_SIZE
ro = 0.01;      % ROBUST_FAILURE_PROBABILITY
EPN = 1e-4; % EPSILON = 0.0001
```

Encoding:

```
symbol = Symbol.empty(0,dq);
for i = 1:dq % i = symbol_index
    % Get the random selection, generated precedently
    % (for performance)

    % selection_indexes, deg = generate_indexes(i, r[i], n)
    [si, d] = geni(i, pr(i), nb, SYC); %

    % Xor each selected array within each other gives the
    % drop (or just take one block if there is only one
    % selected)
    drop = fb(si(1), :); % si(1)+ pf = f blocks
    for n = 2: d % bitwise_xor
        drop = bitxor(drop, fb(si(n),:)); % pf
        % drop = drop ^ blocks[selection_indexes[n]]
    end

    % Create symbol, then log the process
    symbol(i) = Symbol(i, d, drop); %i, d, drop
    symbol(i).log(nb, SYC);

    logo("Encoding", i, dq, EPN, P, dq) % , start_time
    %yield symbol
end
```


```
symbol 1, degree = 1     1
-- Encoding: 1/2 - 50.00% symbols at 1.63 MB/s
~0.08s
symbol 2, degree = 1     1
```

```
[~] = toc;
fprintf("\n---- Correctly dropped %d symbols " + ...
    "(packet size=%d)", dq, P);
```

```
---- Correctly dropped 2 symbols (packet size=65536)
```

Decoding

```
sbc = 0; % solved_blocks_count
isc = 0; % teration_solved_count
tic     % start_time
while isc > 0 || sbc == 0

    isc = 0;

    % Search for solvable symbols
    while 0<length(symbol) % symbol in enumerate(symbols)

        % Check the current degree. If it's 1 then we can
        % recover data
        if symbol(1).deg == 1 % i

            isc = isc + 1;
            bi = symbol(1).nes; % i block_index = next(iter
            syl = symbol(1);    % i
            symbol(1) = [];     % symbols.pop(i)
```

```
            for os = 1:length(symbol) % other_symbol
                if symbol(os).deg > 1 && ~isempty(find( ...
                        symbol(os).nes==bi, 1))

                    % XOR the data and remove the index from
                    % the neighbors
                    symbol(os).data = bitxor(bls(bi), ...
                        symbol(os).data)
                    symbol(os).nes(bi) = []; % .remove

                    symbol(os).deg = symbol(os).deg - 1

                    if VSE
                        fprintf("XOR block_%d with " + ...
                            "symbol_%d : %d", bi, ...
                            symbol(os).ind, symbol(os).nes);
                        % list( .keys()
                    end
                end
            end
        else
            symbol(1) = [];
        end % break here while testing
    end
end
```

Writing down the recovered blocks in a copy.

```
b = bls';   b = b(:);
b = typecast(b(1:1:ceil(f/8)),'uint8');

fid = fopen(fcy, 'w','n',oen);
fwrite(fid,b(1:f)); % shrinked_data
fclose('all');

fprintf("Wrote %d bytes in %s", dir(fcy).bytes, fcy)
```

```
Wrote 181586 bytes in /MATLAB Drive/b-copy.jpg
```

References

1) M. Luby, "LT codes," *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings*., Vancouver, BC, Canada, 2002, pp. 271-280, doi: 10.1109/SFCS.2002.1181950.
2) *Efficient Python Implementation of LT Codes*. url: https://github.com/Spriteware/ lt-codes-python.
3) *Fountain Code: Matlab Implementation of LT Codes*. url: https://github.com/ AnuragPaul0/LT-Codes.

THANK YOU