可逆 C コンパイラにおける算術符号を用いたメモリ使用量削減

M2019SE009 渡邉将匡
指導教員：横山哲郎

1 はじめに

可逆計算とは計算の履歴をさかのぼることができる計算である．反対に，履歴をさかのぼることができない計算を非可逆計算と呼ぶ．可逆計算は楽観的な並列離散事象シミュレーション (PDES) の高速化 [3] などへの応用が知られている．

PDES は計算途中で問題が発生するとロールバックをする必要がある．PDES ではロールバックを行うために，一定間隔でシミュレーションプログラムの現在の状態をすべてメモリに記憶するスナップショット法が使われていた．スナップショット法はメモリを圧迫し，実行速度に悪影響を与えている．

この問題を解決するために可逆計算を使用した手法が提案された [3]．可逆計算を用いた手法ではスナップショット法よりも使用メモリが少ないので，PDES の実行速度を高速化することに成功した．可逆計算を PDES に用いるためにはプログラムを可逆化する必要があり，プログラムの可逆化には Perumalla の手法である可逆 C コンパイラ [2] を使用した．

可逆 C コンパイラでは，非可逆なプログラムでは破棄されてしまう制御情報と値が破棄される変数の直前の値を記憶し，その情報を使用して制御流を復元することでプログラムを可逆に変換している．この手法では実行のたびに入力にも出力にもならないデータ（ゴミデータ）が記憶される．

しかし，可逆 C コンパイラで生成される平均ゴミデータ量は一般に最適ではない．また，可逆 C コンパイラは再帰と繰返しを統一的に扱うことができない．

本研究では，C プログラムを対象とした可逆 C コンパイラを改良し，プログラムの制御の分岐に確率分布が与えられた場合，情報源符号化手法の 1 つである算術符号を応用して制御情報の平均ゴミデータ量が最適となる可逆化手法を提案する．また，提案した手法を解析し，評価を行う．本研究が達成されると，プログラムのメモリ使用量を減らすことができる．

平均ゴミデータ量が最適となるプログラムの可逆化を行うことで，メモリにたまるゴミデータの量を減らし，実行速度を早める効果があることが期待される．また，可逆計算で生成されるゴミデータを圧縮する知見となることが期待される．

2 関連研究

この章では，可逆計算を応用した関連研究を紹介する．

2.1 並列離散事象シミュレーション

並列離散事象シミュレーション (PDES) とは，離散事象シミュレーション (DES) を並列に実行するシミュレーション技術である．DES は工場生産や計算機ネットワーク，交通など，様々なシミュレーションに使用されている．

PDES では，計算が正しく行われなかった場合，ロールバックを実行して計算が正しかった状態までさかのぼる必要がある．PDES はロールバックのためのメモリ使用量がボトルネックとなる場合があり，メモリ使用量の最適化は高速なシミュレーションのために必要である．

文献 [3] では，ロールバックに可逆計算を用いてメモリ使用量の削減を行った．可逆計算法はスナップショット法に比べてメモリ使用量を大幅に削減することができ，特定のシミュレーションにおいては実行速度を 300% から 500% も向上させた．

可逆計算法では，シミュレーションプログラムが可逆である必要があるので，非可逆なシミュレーションプログラムは可逆なシミュレーションプログラムに変換しなければならない．文献 [3] では，可逆 C コンパイラを用いてシミュレーションプログラムの可逆化を行った．

2.2 可逆乱数ジェネレータ

乱数はシミュレーションを行うためには必須である．しかし，可逆計算法をロールバックに使用する PDES などの可逆シミュレーションでは，乱数を生成するジェネレータも可逆でなければならない．

乱数ジェネレータはふつう，何度も変数の値の破棄を行うので，可逆 C コンパイラのように破棄される情報を記憶するとメモリが圧迫され，実行速度に悪影響を与える．しかし，指数分布やパレート分布などの確率分布は完全な可逆性をもたせることが可能であることが示されている．

文献 [1] では，前述の確率分布の可逆性を達成するために，C 言語の UNU.RAN ライブラリの可逆化を行った．さらに，可逆化された乱数ジェネレータを用いて，一様でない乱数を生成するための手法である TDR メソッドを効率的に可逆化した．

3 準備

この章では，プログラムの可逆化のために必要な用語を定義，説明する．また，平均ゴミデータ量の最適化のために必要な用語を説明する．

3.1 プログラムの可逆性と可逆化

可逆性をもつプログラムとは，状態とそのプログラムのみから実行の履歴をさかのぼることができるプログラムで

ある．したがって，可逆性をもつプログラムは，状態とそのプログラムのみから実行の履歴を逆方向に辿ることができる．

プログラムに可逆性をもたせることが，プログラムの可逆化である．プログラムに可逆性をもたせるためには，プログラムの実行後にプログラムの逆実行が行えなければならない．プログラムの逆実行とは，プログラムの実行で変更された状態を実行前の状態に戻すような実行である．

C 言語のプログラムの可逆化を行うためには，可逆化を行うためには実行時の制御流を復元することができ，破棄された変数の値を復元できるように情報を記憶するコードに変換する必要がある．また，実行時の制御流を復元し，制御流の逆方向に逆実行を行うコードを生成する必要がある．

ここでは，通常の実行と情報の記憶を行うコードを生成することを順方向変換，記憶された情報を使用し逆実行を行うコードを生成することを逆方向変換とする．また，順方向変換されたプログラムの実行を順実行，復元のために記憶されたデータをゴミデータとする．

3.2 情報の記憶

プログラムの可逆化のためには，通常なら破棄されてしまう情報を記憶する必要がある．また，逆実行の際に記憶された情報を読み出す必要がある．本研究では，`SAVE()` マクロによりデータを記憶し，`RESTORE()` マクロによりデータを読み出す．

3.3 情報量とエントロピー

情報量とは，情報という概念の量を示す値である．情報量は確率を用いて次のように定義される．事象 a が起こる確率を $\mathrm{P}(a)$ とする．このとき，事象 a の情報量は $-\log_2 \mathrm{P}(a)$ となる．

エントロピーとは，情報源から得られる情報量の期待値である．エントロピーは次のように定義される．事象の集合 $A = \{a_1, a_2, ..., a_n\}$ があり，事象が起こる確率を $\mathrm{P}(a_i)(1 \leq i \leq n)$ とする．このとき，A のエントロピー H は，$H(A) = -\sum_{i=1}^{n} \mathrm{P}(a_i) \log_2 \mathrm{P}(a_i)$ となる．

3.4 情報源符号化と復号

情報源符号化とは，情報源から発生する記号に別の記号を割り当てて変換することである．このとき，情報源から発生する記号を情報源記号，情報源記号系列に割り当てられた各記号を符号語，符号語の集合を符号，符号語の長さを符号長，情報源記号を符号語に変換することを符号化と呼ぶ．以降，情報源記号系列を単に情報源系列と呼ぶ．

復号とは，情報源符号化によって変換された符号語系列を，元の情報源系列に復元することである．しかし，符号語の割当てによっては復号が行えない場合がある．

復号が可能である符号と情報源のエントロピーには次の関係がある．情報源 A のエントロピーを $H(A)$，平均符号長を L とすると，$H(A) \leq L$ となる．したがって，平均符号長の下限はエントロピーの値になる．

3.5 算術符号

算術符号は情報源符号化手法の 1 つである．算術符号は情報源記号の発生確率と，発生確率を用いて分割された 0 から 1 の区間幅を用いて情報源符号化を行う．算術符号では 1 つの記号列が 1 つの符号語に変換される．

算術符号の符号語は，情報源符号化の途中で，1 つの情報源記号を符号語に変換し終わったとき中間表現となる．この中間表現は算術符号の計算途中の値である．同様に，本研究では提案手法での計算途中のゴミデータの値のことをゴミデータの中間表現と呼ぶ．

4 提案手法

この章では提案手法の説明をする．本研究で扱うゴミデータは制御情報のゴミデータのみを対象としている．ここで，平均ゴミデータ量が最適とは，制御の分岐の確率分布に対して平均ゴミデータ量がエントロピー H 以上，$H + 1$ 未満であることをいう．

4.1 提案手法の形式化

提案手法では，情報源系列は各分岐でどの分岐に制御が流れたかを表す記号の列となる．また，時点は各分岐でいずれかの分岐に制御が流れた瞬間となる．情報源系列は，写像 f により符号化される．ただし写像 f は，情報源系列を入力に取り，0 以上 1 未満の無限精度の数を出力する．算術符号の辞書には，各分岐でそれぞれの分岐に制御が流れる確率を使用する．また，算術符号では復号を終了させるために特殊な記号が必要となるが，提案手法では復号の終了はプログラムの実行情報から読み取ることができるので，復号の終了を示す特殊な記号を使用する必要はない．

分岐が n 個ある場合，分岐 $m(1 \leq m \leq n)$ に制御が流れる確率を $p_m(p_m \neq 0)$ とする．ただし，$\sum_{i=0}^{n} p_i = 1(p_0 = 0)$ とする．分岐 m に制御が流れた場合，区間の下限値 b は分岐 $m-1$ までの確率を用いて $b_{next} = b_{prev} + \sum_{i=0}^{m-1} p_i$ と更新される．また，区間幅 w は p_m を用いて $w_{next} = w_{prev} \times p_m$ と更新される．

分岐 m に制御が流れた場合の更新された下限値を b_m とすると，各下限値の間には次の関係が成り立つ．$0 = b_1 < b_2 < \cdots < b_n < 1$ このとき，b_m と他の更新された下限値を区別するためには，b_m が b_{m-1}, b_{m+1} と区別できればよい．また，$b_m = b_{m-1} + p_{m-1}$ である．さらに，b は計算機上のデータでは 2 進数で表されるので，b_m と b_{m-1} を区別するためには，p_{m-1} を 2 進数で表した場合に最初に 1 が現れる桁数まで見れば十分である．したがって，$2^{-d} \leq p_{m-1} < 2^{-d-1}$ となる d に対し，小数点以下 d 桁まで見れば b_m と b_{m-1} を区別することができる．このような d は $d = -\log_2 p_{m-1}$ で与えられる．

同様に，b_m と b_{m+1} を区別するためには小数点以下 $d' = -\log_2$ 桁まで見ればよい．したがって，b_m と他の更新された下限値を区別するためには，$d_m = \max(d, d')$ とな

る d_m に対し，小数点以下 d_m 桁まであれば十分である．このとき，この分岐の符号の平均符号長 L は $L = \sum_{i=0}^{n} p_i d_i$ となる．また，p_i と p_{i-1} の差はふつうあまり大きくない．したがって，$d_i \fallingdotseq -\log_2 p_i$ と近似できる．これを平均符号長の式に代入すると $L = \sum_{i=0}^{n} -p_i \log_2 p_i$ となり，右辺の式がエントロピーの式と一致する．したがって，提案手法の平均符号長はエントロピー H に漸近する．ここで，平均符号長は平均ゴミデータ量となるので，平均ゴミデータ量 G は $H \leq G < H + 1$ となる．

提案手法では，符号化に無限精度の区間分割を用いる場合を考えている．算術符号では，情報源系列の情報源記号 1 個から，符号の中間表現が計算されるごとに区間幅が縮小される．したがって，提案手法では制御の分岐が増えるごとに有効桁数を増やせばよい．現在の区間幅が w, 現在符号化しようとしている制御の分岐が n 分岐である場合，有効桁数は小数点以下 $-\log_2(w \max(D))(D = d_1, d_2, \cdots, d_n)$ 桁となる．

4.2 提案手法

提案手法は，制御の分岐の確率分布が与えられたら，確率分布によって制御の分岐のエントロピーが変化することをアイデアとしている．例えば，if 文の副文 1 が実行される確率を x, 副文 2 が実行される確率を $1 - x$ とした場合のエントロピーは図 1 のようになる．このとき，エントロピーが最大値である 1bit となるような x は 0.5 である．したがって，if 文の副文 1 が実行される確率と副文 2 が実行される確率はどちらも 0.5 となり，if 文の分岐のエントロピーが一様分布で最大値となることがわかる．

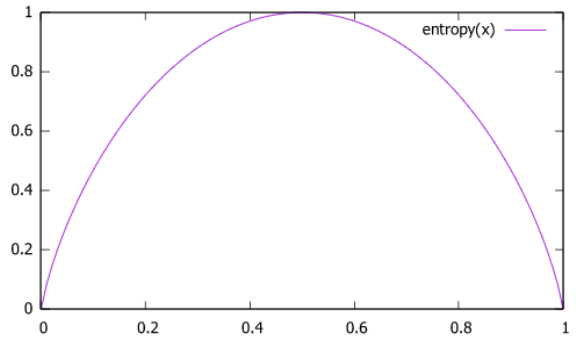

図 1 if 文のエントロピー

提案手法では，制御の分岐が n 分岐し，分岐 m に制御が流れる確率を p_m（ただし $p_0 = 0, m \geq 1$）とする場合，図 2 のように符号化と復号を行う．符号化では，区間幅 w と p_0 から p_{m-1} までの確率を用いて，対応する分岐 m の区間の下限値を求め，区間内の値 b に足し合わせる．また，w と b の初期値はそれぞれ 1 と 0 である．その後，p_m を用いて区間幅を縮小する．復号では，b が分岐 m の区間内にあるのかを判定する．b が区間内にあった場合，b から分岐 m の下限値を引き，p_m で割ることによって b から分岐 m の情報を取り除く．

```
{
    s
    b = b + w * q;
    w = w * p_m;
}
```

```
if (b < (q + p_m)) {
    b = (b - q) / p_m;
    s
}
```

図 2 提案手法の符号化と復号（$q = \sum_{i=0}^{m-1} p_i$）

```
w = 1.0;
b = 0.0;
if (e1) {
  s1
  b = b + w * 0;
  w = w * 1 / 4.0;
} else if (e2) {
  s2
  b = b + w * (0 + 1 /
      4.0);
  w = w * 1 / 2.0 ;
} else {
  s3
  b = b + w*(0 + 1/4.0 +
      1/2.0);
  w = w * 1 / 4.0;
}
SAVE(b);
```

図 3 提案手法を用いた順実行プログラム

```
RESTORE(b);
if (b < 1 / 4.0) {
  rs1
  b = b / (1 / 4.0);
} else if (b < 3 / 4.0)
    {
  rs2
  b = (b - 1 / 4.0) / (1
      / 2.0);
} else if (b < 1.0) {
  rs3
  b = (b - 3 / 4.0) / (1
      / 4.0);
}
```

図 4 提案手法を用いた逆実行プログラム

ただし，復号で取り出せる選択文の制御情報の順番と逆実行が必要とする制御情報の順番は逆であり，繰返し文と再帰関数では順番が一致する．したがって，復号で取り出せる制御情報の順番と逆実行が必要とする制御情報の順番をすべて一致させる必要がある．これは，符号化か復号のどちらかにおいて行えばよい．

4.3 提案手法の適用例

副文 1 が $\frac{1}{4}$, 副文 2 が $\frac{1}{2}$, 副文 3 が $\frac{1}{4}$ の確率で実行されるような if 文を用いた 3 分岐のプログラムを考える．このとき，提案手法によってプログラムを可逆プログラムに変換したものが図 3 と図 4 のプログラムである．

順実行プログラムは制御流の情報を区間内の値として持つ．また，区間内の値の更新のために，現在の区間幅の値をもつ．区間内の値の初期値は 0，区間幅の値の初期値は 1 である．プログラムが実行されるたびに区間内の値を，区間幅と直前までの確率の合計値を掛けた値で足して更新する．したがって，区間内の値は該当する区間の下限値となる．その後，区間幅にそれぞれの文が実行される確率を掛けて更新する．逆実行プログラムでは，区間内の値がどの区間にあるのかで制御流を復元する．区間幅の情報は制御流の情報を含まないので，メモリに記憶する必要はない．

また，提案手法ではプログラムの制御の分岐の確率を用いてゴミデータを表すので，繰返し文と再帰関数を統一的に扱うことができる．

5 比較

この章では，可逆 C コンパイラと提案手法をプログラムに適用し比較を行う．また，この章で扱うすべてのプログラムは，前提条件として制御式が副作用を含んでいない．

5.1 可逆 C コンパイラ

Perumalla のプログラム可逆化手法である可逆 C コンパイラは，PDES のロールバックのために使用されるメモリ量を削減するために提案された．可逆 C コンパイラでは，元のプログラムから順実行プログラムと逆実行プログラムを生成することで元のプログラムを可逆化している．

順実行プログラムは，通常の実行を行い，実行中に破棄される制御情報と，破棄される変数の値が記憶される．

逆実行プログラムは，通常の実行の制御流とは逆方向に実行を行う．逆実行プログラムが逆方向に実行を行えるように，順実行プログラムの文の順序とは逆になるように文を配置する．順実行の制御の合流地点では，記憶された制御情報を用いて制御をどこに分岐させるかを決める．

5.2 可逆 C コンパイラの短所

可逆 C コンパイラはそれぞれの制御の分岐の確率に関して考慮されておらず，確率分布が与えられると平均ゴミデータ量が最適とならない．例えば，副文 1 に 100% 分岐する if 文を可逆化することを考える．可逆 C コンパイラでは if 文の制御情報の記憶にはゴミデータが 1bit 必要である．しかし，この if 文のエントロピーは 0bit なので，可逆 C コンパイラの平均ゴミデータ量は最適でない．

可逆 C コンパイラでは小数以下の情報量をあつめて表現することができない．2 分岐の if 文で，副文 1 が実行される確率が $\log_2 \frac{1}{\sqrt{2}}$ である文を考える．この文が 2 つ連続して配置されるとき，副文 1 が連続して実行される確率は $\frac{1}{2}$ となり，情報量が $\log_2 \frac{1}{2} = 1$ となるので，+1bit で表現可能である．しかし，可逆 C コンパイラでは複数の if 文の制御流をまとめて扱うことができない．したがって，if 文 1 つで 1bit のゴミデータが発生し，どのような制御流であっても if 文 2 つでは 2bit のゴミデータが発生する．また，繰返し文，再帰関数を統一的に扱うことができない．

6 評価

提案手法は制御の分岐の数が十分に多い，またはプログラムの実行回数が十分に多ければ平均ゴミデータ量はエントロピー H 以上，$H+1$ 未満に漸近する．

また，提案手法では実数の変数を用いて，四則演算と比較演算のみを使用することで符号化と復号を行う．符号化で制御情報 1 つから中間表現を生成する際，繰返しと再帰的呼出しが起こらない．したがって，提案手法は符号化と復号が定数時間で終了する．

また，提案手法ではプログラムの制御の分岐の確率を用いてゴミデータを表現するので，繰返し文と選択文，再帰関数を統一的に扱うことができる．

本研究では，平均ゴミデータ量の削減を目的としているので，最悪の場合のゴミデータ量は最適でない場合がある．例えば，if 文で一方の副文が実行される確率が極端に低い場合を考える．if 文の副文 1 が実行される確率が $\frac{1}{10^{100}}$，副文 2 が実行される確率を $1-\frac{1}{10^{100}}$ とする．このとき，if 文の副文の実行のエントロピーは $-\frac{1}{10^{100}}\log_2\frac{1}{10^{100}} - (1-\frac{1}{10^{100}})\log_2(1-\frac{1}{10^{100}}) \fallingdotseq 0$bit となる．したがって，この if 文に提案手法を用いると，平均ゴミデータ量はほぼ 0 となる．しかし，副文 1 が実行された場合，副文 1 の情報量である $-\log_2\frac{1}{10^{100}}$bit のゴミデータが発生してしまう．

提案手法では，ゴミデータの表現に浮動小数点数を用いている．浮動小数点数の演算は定数時間で終わるが，低速である．本研究でメモリ使用量の削減を行う目的は実行速度の高速化であるので，低速である浮動小数点数の演算が悪影響を与える可能性がある．

提案手法は，事前に制御の分岐の確率分布が既知でなければ適用できない．提案手法では静的な算術符号を用いている．静的な算術符号は事前に情報源記号の発生確率が未知の場合は適用できない．したがって，静的な算術符号を応用している本手法も制御の分岐の確率が未知であるプログラムに適用することができない．

7 おわりに

本研究では，プログラムの制御の分岐に確率分布が与えられた場合，情報源符号化手法である算術符号を用いて制御情報の平均ゴミデータ量がエントロピーに漸近する可逆化手法を提案した．また，提案手法の解析を行い，評価を行った．提案手法のゴミデータの中間表現は浮動小数点数を使用していて，繰返しが起こらないので，定数時間で得ることができる．また，ゴミデータの復号も同様に定数時間で終了する．提案手法はプログラムの制御の分岐の確率を用いて変換を行うので，繰返し文と再帰関数を統一的に扱うことができる．

今後の課題として，浮動小数点数の代わりに整数を使って実行速度を上げること，逆実行と復号の流れを一致させること，確率分布が未知である場合に適用可能な手法を提案することが挙げられる．

参考文献

[1] Yoginath, S.B. and Perumalla, K.S.: Efficient Reversible Uniform and Non-Uniform Random Number Generation in UNU.RAN, *Proc. the Annual Simulation Symposium* (*ANSS '18*), Frydenlund, E. Shafagh, J. and Kavak, H. (Eds.), pp.1–10, Society for Computer Simulation International (2018).

[2] Perumalla, K.S.: Introduction to Reversible Computing, pp.147–176 (2013).

[3] Carothers, C.D., Perumalla, K.S. and Fujimoto, R.M.: Efficient Optimistic Parallel Simulations Using Reverse Computation, ACM Trans. Model. Comput. Simul., Vol.9, No.3, pp.224–253 (1999).