Vol.2009-DBS-148 No.23
Vol.2009-FI-95 No.23
2009/7/28

遺伝的プログラミングによる排他制御プログラムの生成について

水野正義[†1,†2] 宝珍輝尚[†1] 野宮浩揮[†1]

任意の特徴をもつトランザクションに対応した排他制御機構を構築するために，遺伝的プログラミング（Genetic programming, GP）によって排他制御プログラムを自動生成するシステムの作成を試みる．システムの入力はトランザクションの特徴設定とし，その特徴をもつトランザクションからスケジュールを生成する．GP により排他制御プログラムを生成し，生成されたスケジュールを制御して適合度を求め，より適合度の高いプログラムを生成する．ここで，GP のノードは，代表的な排他制御法を実現するプログラムを分解・整理することにより求めている．実際に排他制御プログラムが生成できるかを実験により検証した結果，もととなる排他制御プログラムに近いプログラムや，各排他制御法のノードが混合したようなプログラムを生成することができた．

On the Generation of Concurrency Control Program by Using Genetic Programming

Masayoshi Mizuno,[†1,†2] Teruhisa Hochin[†1]
and Hiroki Nomiya[†1]

This paper proposes a concurrency control program generator using genetic programming (GP) to build concurrency control system with the feature of transactions. The schedule is generated from the feature of transactions, which is the input of the generator, and then the programs controlling the schedule are generated by using GP. The fitness of each generated program is calculated in order to generate better program. The nodes used in GP are decided by analyzing and adjusting the programs of typical concurrency control methods. It is experimentally investigated whether the concurrency control program is actually generable. The program similar to a typical concurrency control one, and the program whose nodes are of several typical ones could be generated.

1. はじめに

ユビキタス社会といわれる今日，様々なコンピュータが環境に合わせて発達しており，同時にデータベースの適応分野がますます広がっている．このような分野からは多種多様の要求があり，単一の汎用的なデータベース管理システム (DBMS）では対処し切れない[1),2)]．このことから，分野に対応した機能をもつ DBMS が望まれている[3)–5)]．

ここで，DBMS の性能や信頼性に関わる最も重要な機構のひとつに排他制御機構がある．データベース利用分野に応じて，データを読む操作だけを行ったり，同一のデータを集中的に操作したりというように，異なる特徴をもつ命令の列（トランザクション）が発生し，この特徴に応じた排他制御法を採用することが望まれる．二相ロック法，時刻印順序法や楽観的制御法のような排他制御法のほかにも，ロングトランザクションの考慮，利用者制御のサポート，協調的な協同作業のサポートのための排他制御法が検討されてきている[6)–9)]．また，DBMS のあらゆる機能を拡張可能にする研究でも，多種のトランザクション管理法のサポートに重点をおいて研究が行われている[2),4),5),10)–12)]．

発生するトランザクションの特徴に応じた排他制御機構を用意することができれば，利用分野に最適な DBMS の構築に繋がる．しかしながら，これまでに研究されてきている様々な排他制御法や多種のトランザクション管理法は，特徴のある個々のトランザクションに対して個別に対処するのが主で，様々なトランザクションに対して，統一的な手法でそのトランザクションに合った排他制御法を導出するものではなかった．

一方，人工知能の分野では，生物が環境に適応して進化していく過程を模倣した遺伝的アルゴリズム (Genetic Algorism, GA) や遺伝的プログラミング (Genetic Programming, GP) という手法が開発されている．GA は，進化論的な考え方に基づいてデータを操作することで解の探索や学習を行う手法である[13)]．分散システムの動的負荷均衡に GA を用いた研究が報告されている[14)]．GP は GA の拡張であり，個体が木構造で表現され，プログラムに進化論的な操作を行いやすいようになっている．

そこで本研究では，応用分野独自のトランザクションの特徴に応じた排他制御機構を容易に実現することを目的として，それらのトランザクションに対して最も有効な排他制御プロ

†1 京都工芸繊維大学大学院 工芸科学研究科
Graduate School of Science and Technology, Kyoto Institute of Technology
†2 現在，シャープ株式会社
currently with Sharp Corporation

Vol.2009-DBS-148 No.23
Vol.2009-FI-95 No.23
2009/7/28

グラムを遺伝的プログラミングを使用して自動生成するシステムについて検討する．

以降，**2.** では GP について概説する．**3.** ではシステムの要件と，システムの構成要素について述べる．**4.** では，システムの各関数について述べる．**5.** では，システムの実行例を示す．最後に **6.** でまとめる．

2. 遺伝的プログラミング

遺伝的アルゴリズム（GA）は，進化論的な考え方に基づいてデータを操作することで，解の探索や学習を行う手法である[13)]．個体集団を一定のアルゴリズムに従って進化させることで，環境に適した個体が得られる．

GA のアルゴリズムは以下のようにまとめられる．

(1) 現在の世代の集団をランダムに生成する．
(2) 現在の世代の集団内の各個体に対して適合度をそれぞれ計算する．
(3) 適合度に比例する確率分布を用いて，現在の世代の集団から個体を選択する．
(4) 選択された個体に対して交叉・突然変異等の操作を行い，次世代の集団を生成する．
(5) （2）～（4）の動作を必要な回数分繰り返し，最終的に現在の世代の集団内で最も適合度の高い個体を解とする．

（3）では適合度の低い個体ほど淘汰されやすいようにする．個体の選択方法のひとつにルーレット選択法がある．個体数 n のうちから個体 i を選択する確率を p_i，適合度を f_i とすると，式（1）が成り立つ．

$$p_i = \frac{f_i}{\sum_{k=1}^{n} f_k} \tag{1}$$

（4）で行う操作には以下のようなものがあり，適用頻度は設定により，適用箇所はランダムに決定される．

- 交叉（一点交叉）：交叉点を一箇所選んで，その前後で遺伝子を入れ換える．
- 逆位：決まった範囲の遺伝子を逆順にする．
- 突然変異：遺伝子の値をランダムに変化させる．交叉や逆位はデータを再編成するだけなので，突然変異によって新たなデータを追加し，個体集団の多様性を維持できる．

遺伝的プログラミング (GP) は GA の拡張であり，その個体は木構造で表現される．終端ノード集合 $T = \{T_1, T_2\}$ と，非終端ノード集合 $F = \{F_1, F_2\}$ を組み合わせて表現した木構造を**図 1** に示す．交叉・突然変異などの遺伝的操作は，GP では部分木の入れ替えなどの形で実現される．この木構造は，**図 2** のような括弧を用いた LISP の S 式で表現されることが多い．

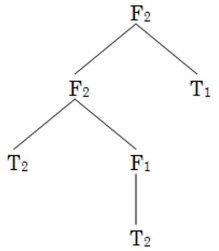

図 1 木構造の例
Fig. 1 An example of a tree structure.

$$(F_2 T_1 \quad (F_2(F_1 T_2) \quad T_2))$$

図 2 S 式で表現した木構造の例
Fig. 2 An example of a tree structure in S-expression.

3. 設　　計

3.1 システムの要件

各排他制御法を実行できるプログラムを細かく関数化したものを GP で操作する記号（ノード）とし，遺伝的操作を行うことで，既存の排他制御法を再構成したり，いくつかを組み合わせた新たな排他制御プログラムを生成したりすることができるようなシステムを考える．生成される個体（排他制御プログラム）の適合度は，どれくらい効率良く排他制御を行えるかで決定する．

本システムの入力となるトランザクションの特徴としては，以下の項目を設定可能とする．

- p1：同時に動作するトランザクションの数
- p2：トランザクションが操作するデータの数
- p3：命令（オペレーション）に書く操作が含まれる確率
- p4：トランザクションが発するオペレーションの最大数
- p5：トランザクションが p4 に達する前に完了する確率

p1 と p3 が大きいほど，また，p2 が小さいほど，複数のトランザクションが一斉に特定のデータに書き込むという競合の発生しやすい状況が増えると考えられる．また，p5 が大きければ，オペレーションの数が p4 で定めた最大数に達することが少なくなり，トランザクションの長さが均一でなくなる．トランザクションの長さの均一さが適合度に及ぼす影響については未知であるが，将来的なロングトランザクションの考慮の必要性なども考え，設

Vol.2009-DBS-148 No.23
Vol.2009-FI-95 No.23
2009/7/28

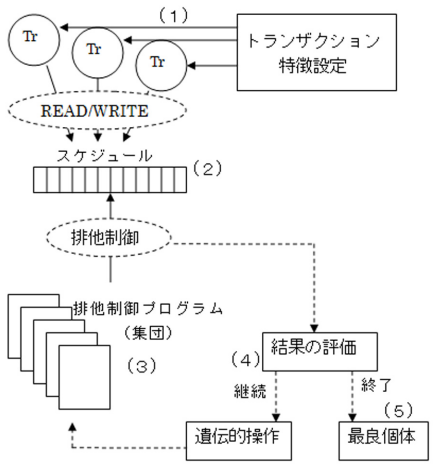

図 3 排他制御プログラム生成システムの概要
Fig. 3 An outline of the concurrency control program generator.

定項目としておくこととする．

3.2 システムの構成

システムの概要を**図 3** に示す．実行順序は以下のとおりである．

(1) トランザクションの特徴を設定し，必要な数のトランザクションを生成する．
(2) 各トランザクションから発せられたオペレーションを生成順にまとめ，スケジュールとして得る．
(3) GP によって排他制御プログラムを生成する．
(4) (2) で得たスケジュールに対して (3) で生成した排他制御プログラムで排他制御を行い，その結果をもとに適合度を計算し，適合度の高い排他制御プログラムをもとに，世代交代を行う．
(5) (3), (4) を必要な回数繰り返し，最終的に最も適合度の高い排他制御プログラムを得る．

3.3 スケジュールの生成

ここでは，プロセスを複数生成し，子プロセスのそれぞれで，ユーザのトランザクションの特徴設定に従ってオペレーションを発する．オペレーションの種類を**表 1** に示す．

オペレーションはメッセージとして親プロセスに届けられる．BEGIN オペレーション以降に発せられた READ および WRITE オペレーションは，親プロセス側で発生順にスケジュールに組み込まれる．END オペレーションを送るとその子プロセスは終了する．

表 1 子プロセスの発するオペレーション
Table 1 Operations in child process.

命令	意味
BEGIN	トランザクションの開始を表す．
READ	データを読む操作を表す．スケジュールに READ を組み込む．
WRITE	データを書く操作を表す．スケジュールに WRITE を組み込む．
END	トランザクションの終了を表す．

オペレーションを発生順に並べるスケジュールを生成すると同時に，トランザクションごとに順番に並べた直列スケジュールも保持しておき，適合度の計算に使用する．

3.4 排他制御プログラムの生成

GP によるプログラム生成のための設定も必要である．以下の設定項目がある．

- s1：最大世代数
- s2：一世代における個体数（集団サイズ）
- s3：初期世代になるプログラムの指定の有無
- s4：プログラムの最大深さ
- s5：プログラムの成長方法
- s6：非終端記号における交叉確率
- s7：終端記号における交叉確率
- s8：コピーのみを行う確率
- s9：評価に用いるスケジュールの数
- s10：小さいプログラムを重視する割合

s3 が初期世代プログラム有の場合，プログラム（S 式表現）を記述した外部ファイルが必要である．プログラムを指定すると，初期世代の集団の半分程度が指定したプログラムとなり，残りはランダムに生成される．プログラムが無の場合は初期世代の全てがランダムに生成される．s5 では，プログラムをランダムに成長させるか，各枝を必ず s4 の最大深さまで成長させるかを設定できる．s6，s7，s8 は遺伝的操作の頻度の設定である．突然変異が起こるのは，交叉もコピーもしない場合である．s9 および s10 は適合度計算に関係する．

3.5 適合度の計算

本システムにおける適合度とは，生成したプログラムがどれだけ設定したトランザクショ

Vol.2009-DBS-148 No.23
Vol.2009-FI-95 No.23
2009/7/28

ンの特徴に合っているかを示す指標である．適合度が高いほうが良いプログラムである．

生成したプログラムを評価するために，まず，3.3 で生成したスケジュールに実際に排他制御を行い，新たな制御済みスケジュールを得る．この制御済みスケジュールの評価結果によって，プログラムの適合度を計算する．なお，評価に用いるのが単一のスケジュールだけでは，特定のスケジュールに対してのみ有効なプログラムが解となってしまう可能性が高いので，スケジュールは複数個用意し，適合度はそれらの評価結果の平均値をとる．生成するスケジュールの数は設定の s9 で変更が可能である．

- **直列化可能性の検証**

制御済みスケジュールの評価においては，まず直列化可能性の検証を行う．本システムでは，最も厳格であり，広く受け入れられている相反直列化可能性についてのみ検証する．排他制御プログラムを実行した結果，直列化可能性が保証されていなければ，それは排他制御プログラムとは言えないものと考え，直列化可能性が保証されない制御を行うプログラムには，常に最低の評価を与える．直列化可能性が保証される場合のみ，以降の適合度計算を行う．

- **同時実行性の検証**

オペレーションは可能な限り同時実行するものとする．ここで，オペレーションを同時に実行して矛盾が生じるような並びになっていれば，直列化可能性が保証されず，適合度計算は行われない．よって，同時実行数が増えるのは基本的に好ましいことであると考えてよい．直列スケジュールの長さを l_0，制御後スケジュールにおける同時実行オペレーションの最大数を最大同時実行数 c_{max}，同時実行を考慮した総ステップ数をスケジュールの長さで割ったものを平均同時実行数 c_{ave} とし，適合度に式（2）で求めた数を加算する．

$$l_0 \times c_{max} \times c_{ave} \tag{2}$$

- **トランザクションの応答時間の比較**

トランザクションの集合 $\{T_1, T_2, \ldots, T_n\}$ があるとき，T_i の直列スケジュールにおける応答時間を ts_i，制御済みスケジュールにおける応答時間を tc_i とする．制御済みスケジュールにおいてトランザクションがアボートされなかった場合，式（3）の計算を行う．

$$\sum_{k=1}^{n} \frac{ts_k}{n \cdot tc_k} \tag{3}$$

結果が 1 より大きければ適合度に乗算する．

- **トランザクションのアボート**

全トランザクションのうちアボートされた割合 p_{abort} を求め，それが 0.5 未満であれば適合度に式（4）で求めた数を乗算する．

$$1 - p_{abort} \tag{4}$$

アボート率が 0.5 を超える場合は，排他制御プログラムとしては性能が著しく悪いと言える．そのため，式（5）で求めた数を乗算する．

$$(1 - p_{abort})^2 \tag{5}$$

- **プログラムサイズによる補正**

同様の制御を行うプログラムならば，サイズが小さいほうが好ましい．プログラムは，非終端記号の数 N_f が大きいほどサイズが大きいと考えられるので，適合度から式（6）で求めた値を減算する．なお，非終端記号の数にかかる定数（式（6）では 0.005）は設定項目の s10 で変更可能である．

$$N_f \times 0.005 \tag{6}$$

4. 実　　装

4.1 二相ロック法のための実装

実際に作成した二相ロック法を実現するプログラムを LISP の S 式で**図 4** に示す．なお，プログラムに使用している非終端記号および終端記号を**表 2** に示す．

図 4 の 1 行目の「ScheLoop」は，スケジュール全体を制御するのに必要不可欠なノード

```
1   (ScheLoop
2       (PROG2
3           (isNoLock
4               (willWRITE
5                   (PROG2
6                       (isDataLocked
7                           LockWait)
8                       doXLock)
9                   (PROG2
10                      (isDataXLocked
11                          LockWait)
12                      doSLock)))
13          (isLastOpe
14              doUnLockAll)))
```

図 4　二相ロック法を実現するプログラム
Fig. 4　An example of two phase locking program.

Vol.2009-DBS-148 No.23
Vol.2009-FI-95 No.23
2009/7/28

表 2 二相ロック法のための記号
Table 2 Nodes for two phase locking.

記号	引数の数	意味
ScheLoop	1	スケジュールの先頭から順に，全てに対して引数を実行する．
PROG2	2	第 1，第 2 引数を順に実行する．
PROG3	3	第 1，第 2，第 3 引数を順に実行する．
isNoLock	1	データにまだロックをかけていなければ引数を実行する．
willWRITE	2	以降に対象が同じ WRITE があれば第 1 引数を， なければ第 2 引数を実行する．
isDataLocked	1	データにロックがかかっていれば引数を実行する．
isDataXLocked	1	データに排他ロックがかかっていれば引数を実行する．
isLastOpe	1	トランザクションの最後のオペレーションなら引数を実行する．
LockWait	終端	データにロックをかけているトランザクションがロックを 解放するまで待つ．
doXLock	終端	データに排他ロックをかける．
doSLock	終端	データに共有ロックをかける．
doUnLock	終端	特定のデータへのロックを解放する．
doUnLockAll	終端	かけているロックを全て解放する．

である．これが存在しなければ，スケジュールの先頭のみを制御するようなプログラムとなる．よって，このノードだけは特別に，ルートに近いほど高い確率で，深い位置ほど低い確率で出現するようにしている．

7 行目と 11 行目の「LockWait」は，自分を，操作対象データをロックしているトランザクションのオペレーションよりも後ろに移動させるという意味をもつ．二相ロック法にはデッドロックが発生するという問題点がある．デッドロックが発生すると「LockWait」部分でスケジュールが永遠に入れ替わり続けることとなるので，ここでデッドロックを検出する必要がある．検出はトランザクションの状態を設定し，それを逐次更新し，参照することで行う．トランザクションの取りうる状態を**表 3** に示す.「LockWait」でロック待ちをすることになれば `TRX_LOCK_WAIT` になり，ロック解放を待つデータが「doUnLockAll」などでアンロックされたら `TRX_ACTIVE` に戻る．デッドロックの検出は以下の手順で行う．

(1) あるトランザクションが `TRX_LOCK_WAIT` になったとき，そのデータにロックをかけているトランザクションを参照する．
(2) (1) のトランザクションのうち，`TRX_LOCK_WAIT` であるものを探す.
(3) (2) が存在すれば,(1) で `TRX_LOCK_WAIT` になったトランザクションをアボート（状態を `TRX_ABORTED` に）する．

表 3 トランザクションのとりうる状態
Table 3 Statuses of transactions.

状態	意味
`TRX_ACTIVE`	トランザクション発生後，実行中である．
`TRX_LOCK_WAIT`	操作するデータのロック解放を待っている． 同時に，ロック解放を待つデータ名を保持する．
`TRX_ABORTED`	トランザクションが異常終了（中止）された．
`TRX_COMMITTED`	トランザクションが正常終了された．

4.2 時刻印順序法のための実装

時刻印順序法を実現するプログラムを**図 5** に示す．プログラムに使用している非終端記号および終端記号を**表 4** に示す．

二相ロック法のプログラムと同様に，1 行目の「ScheLoop」によってスケジュール全体を制御する．オペレーションが READ なら 4〜6 行目の処理を，WRITE なら 8〜10 行目の処理を行う．READ なら Write 時刻印 (WTS) の値を参照し，WRITE なら WTS および Read 時刻印 (RTS) の値を参照してトランザクションをアボートするか否かを判定する．

表 4 にある「isWTSandRTS」は図 5 のプログラムでは使用されていないが,「isWTSorRTS」のように似た意味をもつものがあるので，ノードの多様化を図って導入している．

4.3 楽観的制御法のための実装

楽観的制御法を実現するプログラムを**図 6** に示す．プログラムに使用している非終端記号および終端記号を**表 5** に示す．

まず,「ReadPhase」によってトランザクションの時刻印 Start を発生順に現在時刻で更

```
1    (ScheLoop
2        (PROG2
3            (isREAD
4                (isWTS
5                    trxAbort
6                    updateRTS))
7            (isWRITE
8                (isWTSorRTS
9                    trxAbort
10                   updateWTS))))
```

図 5 時刻印順序法を実現するプログラム
Fig. 5 An example of timestamp ordering program.

Vol.2009-DBS-148 No.23
Vol.2009-FI-95 No.23
2009/7/28

表 4 時刻印順序法のための記号
Table 4 Nodes for timestamp ordering.

記号	引数の数	意味
isREAD	1	READ なら引数を実行する.
isWRITE	1	WRITE なら引数を実行する.
isWTS	2	トランザクションの時刻印がデータの WTS より小さければ第 1 引数を，大きければ第 2 引数を実行する.
isWTSorRTS	2	トランザクションの時刻印がデータの WTS と RTS のどちらかより小さければ第 1 引数を，そうでなければ第 2 引数を実行する.
isWTSandRTS	2	トランザクションの時刻印がデータの WTS と RTS の両方より小さければ第 1 引数を，そうでなければ第 2 引数を実行する.
trxAbort	終端	同じトランザクションのオペレーションを全てスケジュールから除く.
updateRTS	終端	データの RTS を更新する.
updateWTS	終端	データの WTS を更新する.

```
1    (PROG2
2        ReadPhase
3        (ScheLoop
4            (ValidationPhase
5                trxAbort)))
```

図 6 楽観的制御法を実現するプログラム
Fig. 6 An example of optimistic concurrency control program.

表 5 楽観的制御法のための記号
Table 5 Nodes for optimistic concurrency control.

記号	引数の数	意味
ValidationPhase	1	トランザクションに時刻印 Validate を付与し，確認フェーズの条件をパスするかどうかを調べる．パスしなければ引数を実行する.
ReadPhase	終端	トランザクションに，スケジュールに登場する順に時刻印 Start を付与する.

新する．これは読み出しフェーズに該当する．次に,「ScheLoop」でスケジュールを順にたどり，トランザクションの時刻印 Validate を現在時刻で更新しつつ，確認フェーズに該当する条件を満たすかどうかをチェックする．満たしていれば何もせず，満たしていなければ「trxAbort」によってそのトランザクションをアボートする．

5. 実 行 例

5.1 初期世代指定なしでの生成

実際に設定の入力から排他制御プログラム生成を行う例を示す.

トランザクションの特徴設定は，$p1 = 4$, $p2 = 3$, $p3 = 0.6$, $p4 = 5$, $p5 = 0.1$ である．p1 が極端に小さいということはなく，また p3 によりオペレーションの半分以上が WRITE となるため，このトランザクションから生成されるスケジュールは排他制御を行わなければ直列化可能性が保証されないことが多い．実験的に決定した GP のための設定は，$s1 = 800$, $s2 = 40$, $s3 =$ 無, $s4 = 8$, $s5 =$ ランダム, $s6 = 0.1$, $s7 = 0.6$, $s8 = 0.1$, $s9 = 12$, $s10 = 0.005$ である．なお，乱数のシードは 90126 とした.

以上の設定で，578 世代目に**図 7** に示すプログラムが最良個体として生成された．プログラムの適合度はおよそ 18.60 であった.

3 行目の「doXLock」で排他ロックを行うが，データにロックがかかっていると不可能である．その場合，4 行目の「LockWait」でロックの解放を待つ．ロックをかけることができた場合,「LockWait」は自分でかけたロックは待たないので通過する．6 行目の「doXLock」は「LockWait」後に見るので，ここで確実に排他ロックをかけられる．そして，トランザクションの最後のオペレーションであれば，8 行目の「doUnLockAll」でかけているロックを全て解放する．このアルゴリズムをまとめると，非常に単純な二相ロック法であると言える．用いるロックの種類が排他ロックのみであるなど，用意している記号を考慮するとまだ進化の余地はあるが，想定した代表的な排他制御法を再現できている.

```
1     (ScheLoop
2         (PROG3
3             doXLock
4             LockWait
5             (PROG3
6                 doXLock
7                 (isLastOpe
8                     doUnLockAll)
9                 (isLastOpe
10                    doUnLock))))
```

図 7 初期世代設定なしで生成されたプログラム
Fig. 7 A generated program that the first generation is not specified.

Vol.2009-DBS-148 No.23
Vol.2009-FI-95 No.23
2009/7/28

```
1     (PROG2
2         (ScheLoop
3             (PROG2
4                 LockWait
5                 (PROG2
6                     (PROG2
7                         doXLock
8                         (isNoLock
9                             (isWTSorRTS
10                                doUnLock
11                                (isNoLock
12                                    doUnLock))))
13                    (isWTSorRTS
14                        (isWRITE
15                            (isLastOpe
16                                doUnLock))
17                        (isWTS
18                            trxAbort
19                            (PROG3
20                                (willWRITE
21                                    (isNoLock
22                                        doXLock)
23                                    (isLastOpe
24                                        LockWait))
25                                    updateRTS
26                                    (isREAD
27                                        updateRTS)))))))
28        (PROG2
29            LockWait
30            (isWTSorRTS
31                (isWTS
32                    (isWTS
33                        (isWTSorRTS
34                            trxAbort
35                            (isDataXLocked
36                                doUnLockAll))
37                        (isWTS
38                            doSLock
39                            (isLastOpe
40                                (isWTS
41                                    (ValidationPhase
42                                        ReadPhase)
43                                    doSLock))))
44                    ReadPhase)
45                (isDataLocked
46                    (isWRITE
47                        doUnLock)))))
```

図 8 二相ロック法から進化したプログラム
Fig. 8 A generated program that evolved from two phase locking program.

5.2 二相ロック法からの生成

次に，初期世代に二相ロック法のプログラムを指定した例を示す．このために，s3 の指定を「有」とし，外部ファイルに二相ロック法のプログラムを記述して開始する．

この設定で，733 世代目に**図 8** に示すプログラムが最良個体として生成された．なお，このプログラム全体の適合度は約 29.13 で，初期世代用に指定した二相ロック法のプログラムの適合度は約 26.05 であった．

プログラムは,「ScheLoop」によりスケジュール全体に対する制御を行う前半部分と，スケジュールの先頭に対してのみ制御を行う後半部分とに分けられる．

前半部分において，7 行目に「doXLock」が存在するため，8 行目の「isNoLock」の引数（9〜12 行目）が実行されることはない．13〜16 行目では，プログラム全体に時刻印更新系の記号が「updateRTS」しか存在しないことから，データの RTS が自分のトランザクションの時刻印よりも小さく，かつ WRITE で，かつトランザクションの最後のオペレーションならばかけているロックを解放する．17 行目の「isWTS」は前述のとおり「updateWTS」が存在しないので，常に第 2 引数が実行される．その第 2 引数である 19〜27 行目については，まず 21〜22 行目は 7 行目で「doXLock」を通っているのでほとんど意味がない．23〜24 行目は，今後 WRITE を行わないトランザクションの最後のオペレーション（つまり READ）であれば，データのロック解放を待つ．最後に 25 行目の「updateRTS」で RTS を更新する．26〜27 行目は条件判断のついた 25 行目と同じ処理である．

後半部分は「ScheLoop」の引数でないので，前半の処理が終わってからスケジュールの先頭に対して行う処理である．29 行目の「LockWait」でロックの解放を待つので，以降の

Vol.2009-DBS-148 No.23
Vol.2009-FI-95 No.23
2009/7/28

処理はデータのどれかにロックをかけているトランザクションのオペレーションに対して行われる可能性が高い．30〜33 行目と 37 行目に WTS や RTS の条件判断があるが，WTS が更新されることはないので，自分のトランザクションの時刻印がデータの RTS よりも小さい場合に 44 行目の「ReadPhase」，大きい場合に 45 行目以降を見ることになり，32〜43 行目は見られることがない．また，44 行目の「ReadPhase」は単体では特に意味がない．45 行目以降では，データにロックがかかっている場合，自分が WRITE ならデータにかけているロックを解放する．これ以降にロックが解放されたデータを別のトランザクションが参照したりはしないので，大した効果はないと考えられる．

以上より，適合度を上げる効果が期待されるのは，前半のスケジュール全体に対する制御の部分だけであると考えられる．このアルゴリズムをまとめると以下のようである．

(1) まず操作対象データのロック解放を待つ．
(2) データが空いてから排他ロックをかける．
(3) 初めてそのデータを操作対象とするトランザクションは RTS を更新する．
(4) RTS の大小比較により，後発のトランザクションに追い抜かれた WRITE オペレーションがトランザクションの最後であれば，ロックを解放する．

これは，時刻印を利用したロック系のプログラムであると考えられる．今回生成したスケジュールに対してのみ有効である可能性が高いが，想定した 2 つの排他制御法のためのノードが混ざることにより，双方の性質を持った排他制御プログラムが生成されている．

6. おわりに

本研究では，様々な特徴をもつトランザクションに対応した排他制御機構を構築するために，GP による排他制御プログラムを自動生成するシステムの作成を試みた．システムの入力はトランザクションの特徴設定とし，その特徴をもつトランザクションからスケジュールを生成する．GP により排他制御プログラムを生成し，生成されたスケジュールを制御することによって適合度を求め，より適合度の高いプログラムを生成した．ここで，GP のノードは代表的な排他制御法を実現するプログラムを分解・整理することにより求めた．そして，実際に排他制御プログラムが生成できるかを実験により検証した．この結果，もととなる排他制御プログラムに近いプログラムや，各排他制御法のノードが混合したようなプログラムを生成することができ，既存の排他制御法のみならず，それらの性質を併せもった排他制御プログラム生成の可能性を見ることができた．

今後は，GP の終端および非終端記号の種類の充実や，多種多様の排他制御法への適応が課題である．また，生成されたプログラムは人間にとっては分かりやすいプログラムとは言えない．生成されたプログラムを，同じ処理を行うより簡潔なプログラムに変換するツールの作成も望まれる．

参 考 文 献

1) Stonebraker, M. and Cetintemel, U.: One Size Fits All: An Idea Whose Time Has Come and Gone, Proc. of the 21st ICDE, pp. 2–11 (2005).
2) Geppert, A. and Dittrich, K.R.: Constructing the Next 100 Database Management Systems: Like the Handyman or Like the Engineer?, ACM SIGMOD RECORD, Vol. 23, No. 1 (1994).
3) Seltzer, M.: Beyond Relational Databases, Commun. ACM, Vol. 51, No. 7, pp. 52–58 (2008).
4) Dittrich, K. R. : Database Technology Research at the University of Zurich: Using and Engineering Object-Oriented, Active, and Heterogeneous DBMS, 信学技法, Vol. 91, No. 538 (1992).
5) Wells, D. L. *et al.*: Architecture of an Open Object-Oriented Database Management System, COMPUTER, Vol. 25, No. 10, pp.74–82 (1992).
6) Barghouti, N. and Kaiser, G.E.: Concurrency Control in Advanced Database Applications, ACM Computing Surveys, Vol. 23, No. 3, pp.2 69–317 (1991).
7) Garcia-Morina, H.: Using Semantic Knowledge for Transaction Processing in a Distributed Database, ACM TODS, Vol. 8, No. 2, pp. 186–213 (1983).
8) Skarra, A. H. and Zdonik, S. B.: Concurrency Control and Object-Oriented Databases, Object-Oriented Concepts, Databases, and Applications (Kim, W. and Lochovsky, F.H., eds.), ACM, New York, pp. 395–421 (1989).
9) 谷口伸一，西尾章治郎，久保信也：オブジェクト指向データベースにおける競合保存直列可能スケジューラの有効性，信学論 D-I, Vol. J77-D1, No. 7, pp. 514–524 (1994).
10) Chrysanthis, P.K. and Ramamrithan, K.: Synthesis of Extended Transaction Models Using ACTA, ACM TODS, Vol. 19, No. 3, pp. 450–491 (1994).
11) Georgakopoulos, D. *et al.*: Specification and Management of Extended Transactions in a Programmable Transaction Environment, Proc. of the 10th ICDE, pp.462–473 (1994).
12) 大野友寛，宝珍輝尚，都司達夫：排他制御機構の拡張可能化についての一考察，情報処理学会データベースシステム研究会報告, Vol.96, No.11, pp.91–98 (1996).
13) 伊庭斉志：遺伝的プログラミング，東京電機大学出版 (1996).
14) 棟朝雅晴，高井昌彰，佐藤義治：確率学習による適応度評価を導入した遺伝的アルゴリズムに基づく動的負荷均衡，情報処理学会論文誌, Vol. 36, No. 4, pp.868–878 (1995).