

(19)日本国特許庁(JP)

(12)公開特許公報(A)

(11)公開番号

特開2023-45347

(P2023-45347A)

(43)公開日 令和5年4月3日(2023.4.3)

(51)国際特許分類

G 0 6 F 8/30 (2018.01)

F I

G 0 6 F 8/30

テーマコード(参考)

5 B 3 7 6

審査請求 未請求 請求項の数 8 O L (全35頁)

(21)出願番号	特願2021-153696(P2021-153696)	(71)出願人	000005223 富士通株式会社 神奈川県川崎市中原区上小田中4丁目1番1号
(22)出願日	令和3年9月22日(2021.9.22)	(74)代理人	110002918 弁理士法人扶桑国際特許事務所
		(72)発明者	新井 正樹 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		Fターム(参考)	5B376 BC32 FA13

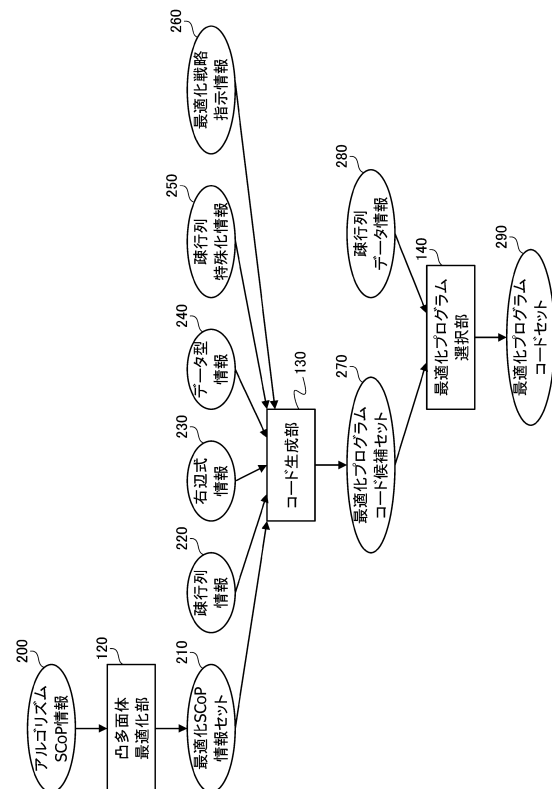
(54)【発明の名称】 プログラムおよび情報処理方法

(57)【要約】

【課題】疎行列処理に対して最適化されたソースコードを効率的に得る。

【解決手段】処理部は、行列に対するループ処理が静的制御部形式で記述された第1コードを凸多面体モデルにより最適化することで複数の第2コードを取得する。処理部は、疎行列の非ゼロの要素を表す変数を示す疎行列情報と、第2コードに含まれる関数に対応する演算式を示す式情報と、変数に対して使用する型を示すデータ型情報とに基づいて、複数の第2コードを複数のソースコード候補に変換する。処理部は、複数のソースコード候補それぞれを用いた場合の疎行列に対する処理性能の評価に応じて複数のソースコード候補の中からソースコードを選択する。

【選択図】図5



**【特許請求の範囲】****【請求項 1】**

疎行列に対する処理を示すソースコードを生成するプログラムであって、  
行列に対するループ処理が静的制御部形式で記述された第 1 コードを凸多面体モデルにより最適化することで複数の第 2 コードを取得し、  
前記疎行列の非ゼロの要素を表す変数を示す疎行列情報と、第 2 コードに含まれる関数に対応する演算式を示す式情報と、変数に対して使用する型を示すデータ型情報とに基づいて、前記複数の第 2 コードを複数のソースコード候補に変換し、  
前記複数のソースコード候補それぞれを用いた場合の前記疎行列に対する処理性能の評価に応じて前記複数のソースコード候補の中から前記ソースコードを選択する、  
処理をコンピュータに実行させるプログラム。

10

**【請求項 2】**

前記疎行列情報は、前記疎行列の表現形式に応じて前記ソースコードで使用する複数の変数であって、前記ループ処理を制御するインデックスを示す第 1 変数と前記疎行列の行番号を示す第 2 変数と前記疎行列の列番号を示す第 3 変数とを含む前記複数の変数それぞれの間の依存関係を示す情報を含み、  
前記複数の第 2 コードから前記複数のソースコード候補への変換では、前記疎行列情報に基づいて、前記複数の第 2 コードに含まれる前記ループ処理の記述を、前記複数の変数を用いたコードに変換する、  
処理を前記コンピュータに実行させる請求項 1 記載のプログラム。

20

**【請求項 3】**

前記複数の変数それぞれの間の前記依存関係と前記複数の第 2 コードそれぞれに含まれるループ構造とに基づいて、前記疎行列情報に対して前記複数の第 2 コードそれぞれを利用可能であるか否かを判定し、利用可能であると判定された前記第 2 コードをソースコード候補に変換する、  
処理を前記コンピュータに実行させる請求項 2 記載のプログラム。

**【請求項 4】**

前記複数の変数それぞれの値域を示す情報に基づいて、前記複数のソースコード候補それぞれにおける変数に対する型の特殊化および前記疎行列の要素の値の特殊化の少なくとも何れかを行う、  
処理を前記コンピュータに実行させる請求項 2 または 3 記載のプログラム。

30

**【請求項 5】**

前記第 1 コードおよび前記複数の第 2 コードでは、変数の型が省略されるとともに代入文における右辺の前記演算式が前記関数により省略されて記述されており、  
前記式情報は、前記関数に対応する前記演算式の情報を有し、  
前記複数の第 2 コードから前記複数のソースコード候補への変換では、前記式情報および前記データ型情報に基づいて、前記複数の第 2 コードに含まれる前記関数を前記演算式のコードに変換する、  
処理を前記コンピュータに実行させる請求項 1 ~ 4 の何れか一項に記載のプログラム。

**【請求項 6】**

前記複数の第 2 コードは、並列化対象の前記ループ処理を示す記述を含み、  
前記複数の第 2 コードから前記複数のソースコード候補への変換では、前記複数のソースコード候補における、前記並列化対象の前記ループ処理に対応するループに対して、並列化指示文を挿入する、  
処理を前記コンピュータに実行させる請求項 1 ~ 5 の何れか一項に記載のプログラム。

40

**【請求項 7】**

前記ソースコードの選択では、前記複数のソースコード候補のうち、前記処理性能を示す指標が基準値よりも良いソースコード候補を、前記ソースコードとして選択する、  
処理を前記コンピュータに実行させる請求項 1 ~ 6 の何れか一項に記載のプログラム。

**【請求項 8】**

50

疎行列に対する処理を示すソースコードを生成する情報処理方法であって、  
コンピュータが、  
行列に対するループ処理が静的制御部形式で記述された第1コードを凸多面体モデルにより最適化することで複数の第2コードを取得し、  
前記疎行列の非ゼロの要素を表す変数を示す疎行列情報と、第2コードに含まれる関数に対応する演算式を示す式情報と、変数に対して使用する型を示すデータ型情報とに基づいて、前記複数の第2コードを複数のソースコード候補に変換し、  
前記複数のソースコード候補それぞれを用いた場合の前記疎行列に対する処理性能の評価に応じて前記複数のソースコード候補の中から前記ソースコードを選択する、  
情報処理方法。

10

【発明の詳細な説明】

【技術分野】

【0001】

本発明はプログラムおよび情報処理方法に関する。

【背景技術】

【0002】

HPC (High Performance Computing) アプリケーションではプログラムのホットスポットに限られる傾向がある。例えば、プログラムの特徴を捉えるためにプロファイルデータを取る場合でも、幾つかのループ(カーネルループ)のみを調査すれば良いことが多い。HPCアプリケーションのカーネルループは一般に大量のデータにアクセスする。カーネルループを高速に実行するために、CPU (Central Processing Unit) のキャッシュの有効利用が図られる。

20

【0003】

大量のデータにアクセスし得るループ処理として行列演算がある。行列演算では疎行列が処理対象になることがある。疎行列は、行列やベクトルの要素にゼロが多い場合に利用されるデータ構造である。疎行列は、ゼロを明示的に保持せず、非ゼロのデータと、どの位置に非ゼロが存在するかの情報を保持する。疎行列を用いることで、メモリとCPU間のデータ転送量を減らしてキャッシュを有効利用し、プログラムの実行を高速化し得る。

【0004】

ここで、プログラムの実行を効率化するために、ソースコードを実行可能コードに変換する過程でコンパイラによる最適化が行われたり、ソースコードレベルでの最適化が行われたりすることがある。

30

【0005】

例えば、ソースプログラムに対して疎行列に関する不要な演算の実行を省略可能にする命令文を挿入することで、当該不要な演算を実行しないで済むようにするコンパイル処理装置の提案がある。

【0006】

また、アプリケーションプログラムのソースコードから、並列演算が可能なホットスポットを抽出して、アクセラレータデバイス用コードを自動的に生成する技術も提案されている。提案の計算機は、ソースコードから生成した中間コードの中から、凸多面体モデルを利用して、静的制御部(SCoP: Static Control Parts)と呼ばれるループ構造を抽出する。計算機は、検出したループ構造をCPU処理部とGPU (Graphics Processing Unit) 処理部とに分割し、CPU処理部内の中間コード上にチェックポイント処理を配置する。計算機は、中間コードの各関数からCPUマシンコードまたはGPUアセンブリコードを生成する。

40

【0007】

また、コンピュータシステム命令から有向非巡回グラフを作成し、有向非巡回グラフの凸多面体表現を決定し、凸多面体表現を使用し、コンピュータシステム命令の実行スケジュールに適用する最適化を決定するコンピュータシステムの提案もある。提案のコンピュータシステムは、当該実行スケジュールおよびプロセッサアーキテクチャに基づいて、コ

50

ンピュータシステム命令の実行可能コードを生成する。

【0008】

また、コンパイル中にソースコードを最適化するために凸多面体ループ変換を利用するコンパイラ技術の提案もある。

更に、凸多面体モデルによりSCoP形式のプログラムを最適化する方法の提案もある。提案の方法を実行する装置は、ベースとなるSCoP形式のコードに、凸多面体モデルによる各種のループ最適化を適用し、最適化されたSCoP形式のコードを複数生成できる。

【先行技術文献】

【特許文献】

【0009】

【特許文献1】特開2007-66128号公報

【特許文献2】国際公開第2017/216858号

【特許文献3】米国特許出願公開2019/0278593号明細書

【特許文献4】米国特許出願公開2009/0307673号明細書

【非特許文献】

【0010】

【非特許文献1】Cohen、外2名、“A Polyhedral Approach to Ease the Composition of Program Transformations”、2004年、European Conference on Parallel Processing、EuroPar 2004: EuroPar 2004 Parallel Processing、p

10

20

p.292-303

【発明の概要】

【発明が解決しようとする課題】

【0011】

疎行列処理のソースコードには、ポインタの使用やデータ間接参照が含まれるため、コンパイラで最適化することは難しい。そこで、ソースコードに記述され得る各種疎行列処理のアルゴリズムに対して、最適化済のライブラリを事前に用意しておき、当該ライブラリを利用することがある。

【0012】

しかし、疎行列処理では、当該疎行列処理のアルゴリズムに対して、各種疎行列フォーマット、データ型および非ゼロの分布に応じた疎行列の性質などのパラメータが性能に大きく影響するため、事前に用意されたライブラリでは、不十分な性能になることがある。特に、疎行列の性質を利用するために、実行時に最適化済コードを作成し、コンパイルして、実行するという、実行時コンパイル手法が用いられる場合があるが、事前に用意されたライブラリでは、実行時コンパイル手法に対処できない。また、事前に用意されたライブラリが、プログラム記述者により利用される疎行列処理のデータ構造やアルゴリズムに合わない場合、当該ライブラリを適用することはできない。

30

【0013】

1つの側面では、本発明は、疎行列処理に対して最適化されたソースコードを効率的に得ることができるプログラムおよび情報処理方法を提供することを目的とする。

40

【課題を解決するための手段】

【0014】

1つの態様では、疎行列に対する処理を示すソースコードを生成するプログラムが提供される。提案のプログラムは、行列に対するループ処理が静的制御部形式で記述された第1コードを凸多面体モデルにより最適化することで複数の第2コードを取得し、疎行列の非ゼロの要素を表す変数を示す疎行列情報と、第2コードに含まれる関数に対応する演算式を示す式情報と、変数に対して使用する型を示すデータ型情報とに基づいて、複数の第2コードを複数のソースコード候補に変換し、複数のソースコード候補それぞれを用いた場合の疎行列に対する処理性能の評価に応じて複数のソースコード候補の中からソースコードを選択する、処理をコンピュータに実行させる。

50

## 【 0 0 1 5 】

また、1つの態様では、情報処理方法が提供される。

## 【 発明の効果 】

## 【 0 0 1 6 】

1つの側面では、疎行列処理に対して最適化されたソースコードを効率的に得ることができる。

## 【 図面の簡単な説明 】

## 【 0 0 1 7 】

【 図 1 】 第 1 の実施の形態の情報処理装置を説明する図である。

【 図 2 】 第 2 の実施の形態の情報処理装置のハードウェア例を示す図である。

10

【 図 3 】 行列演算のソースコード例を示す図である。

【 図 4 】 情報処理装置の機能例を示す図である。

【 図 5 】 情報処理装置で処理されるデータの例を示す図である。

【 図 6 】 情報処理装置の処理例を示すフローチャートである。

【 図 7 】 行列ベクトル積プログラムの例を示す図である。

【 図 8 】 疎行列ベクトル積プログラムの例を示す図である。

【 図 9 】 アルゴリズム S C o P 情報の例を示す図である。

【 図 1 0 】 最適化 S C o P 情報の第 1 の例を示す図である。

【 図 1 1 】 最適化 S C o P 情報の第 2 の例を示す図である。

【 図 1 2 】 最適化 S C o P 情報の第 3 の例を示す図である。

20

【 図 1 3 】 利用可能性判定例を示すフローチャートである。

【 図 1 4 】 疎行列情報 ( C S R ) の例を示す図である。

【 図 1 5 】 疎行列情報 ( C S C ) の例を示す図である。

【 図 1 6 】 疎行列情報 ( C O O ) の例を示す図である。

【 図 1 7 】 最適化プログラムコード候補セット生成例を示すフローチャートである。

【 図 1 8 】 右辺式情報の例を示す図である。

【 図 1 9 】 データ型情報の例を示す図である。

【 図 2 0 】 最適化プログラムコード候補の第 1 の例を示す図である。

【 図 2 1 】 最適化プログラムコード候補の第 2 の例を示す図である。

【 図 2 2 】 最適化プログラムコード候補の第 3 の例を示す図である。

30

【 図 2 3 】 最適化プログラムコード候補の第 4 の例を示す図である。

【 図 2 4 】 疎行列特殊化情報の例を示す図である。

【 図 2 5 】 最適化プログラムコード候補の第 6 の例を示す図である。

【 図 2 6 】 最適化プログラムコード候補の第 7 の例を示す図である。

【 図 2 7 】 最適化戦略指示情報の例を示す図である。

## 【 発明を実施するための形態 】

## 【 0 0 1 8 】

以下、本実施の形態について図面を参照して説明する。

## [ 第 1 の実施の形態 ]

第 1 の実施の形態を説明する。

40

## 【 0 0 1 9 】

図 1 は、第 1 の実施の形態の情報処理装置を説明する図である。

情報処理装置 1 0 は、疎行列処理のためのソースコードの生成を支援する。情報処理装置 1 0 は、記憶部 1 1 および処理部 1 2 を有する。記憶部 1 1 は、R A M ( R a n d o m A c c e s s M e m o r y ) などの揮発性記憶装置でもよいし、H D D ( H a r d D i s k D r i v e ) やフラッシュメモリなどの不揮発性記憶装置でもよい。処理部 1 2 は、C P U、D S P ( D i g i t a l S i g n a l P r o c e s s o r )、A S I C ( A p p l i c a t i o n S p e c i f i c I n t e g r a t e d C i r c u i t )、F P G A ( F i e l d P r o g r a m m a b l e G a t e A r r a y ) などを含み得る。処理部 1 2 はプログラムを実行するプロセッサでもよい。「プロセッサ」は、複数のプロセッサの集合 ( マルチプロセッサ ) を含み得る。

50

## 【 0 0 2 0 】

まず、処理部 1 2 は、情報処理装置 1 0 に入力された S C o P コード 2 0 を取得する。S C o P コード 2 0 は、行列に対するループ処理が静的制御部形式、すなわち、S C o P 形式で記述されたコードである。S C o P コード 2 0 は、ある行列ベクトル積のアルゴリズムを抽象化した表現である。ここで、S C o P は、配列インデックスやループ条件文が全てアフィン式で表されるループ記述である。アフィン式は、ループ変数の線形結合と定数項との加算である式である。ただし、S C o P コード 2 0 では、ソースコードに通常存在するデータ型情報や代入文の右辺の具体的な計算方法は、抽象化されて省略されている。例えば、代入文の右辺は、関数を表す「f 0」や「f 1」の記述のみとして抽象化されている。S C o P コード 2 0 の「do」の記述はループを表す。

10

## 【 0 0 2 1 】

S C o P コード 2 0 は、ユーザが必要とする疎行列処理に応じて予め作成される。一例として、S C o P コード 2 0 は、行列 A および列ベクトル x , y に対する  $y = A * x$  の行列ベクトル積の演算を示す。S C o P コード 2 0 では、行列 A は二次元配列 M で表される。列ベクトル x は一次元配列 v で表される。列ベクトル y は一次元配列 r v で表される。行列 A の行数は N R であり、列数は N C である。行列 A の行を示すインデックスは r であり、列を示すインデックスは c である。例えば、「do ( r 0 ( - N R 1 ) )」の記述は、インデックス r を 0 から N R まで 1 ずつインクリメントしながらループを実行することを示す。なお、S C o P コード 2 0 は、第 1 コードと表記されてもよい。

## 【 0 0 2 2 】

処理部 1 2 は、S C o P コード 2 0 を凸多面体モデルにより最適化することで、複数の最適化 S C o P コードを取得する (ステップ S 1 )。凸多面体モデルによる最適化は、凸多面体最適化と表記される。複数の最適化 S C o P コードそれぞれは、S C o P コード 2 0 に対する最適化済の S C o P コードである。S C o P コードに対して凸多面体最適化を行うツールとして、例えば P o l l y、P L U T E および G r a p h i t e などがある。

20

## 【 0 0 2 3 】

処理部 1 2 は、凸多面体モデルにより、S C o P コード 2 0 に含まれるループ構造を線形代数的に解析してモデル化し、データの依存関係や境界条件を計算することで、並列性の抽出、および、ループ分割やループ交換などのループ最適化の適用を行う。処理部 1 2 は、凸多面体最適化により、S C o P コード 2 0 に対して、複数パターンの最適化 S C o P コードを生成する。例えば、複数の最適化 S C o P コードは、最適化 S C o P コード 3 0 , 3 1 , 3 2 を含む。なお、最適化 S C o P コード 3 0 , 3 1 , 3 2 それぞれは、第 2 コードと表記されてもよい。

30

## 【 0 0 2 4 】

例えば、処理部 1 2 は、S C o P コード 2 0 の外側ループを並列化した最適化 S C o P コード 3 0 を生成する。この場合、処理部 1 2 は、S C o P コード 2 0 の「do ( r 0 ( - N R 1 ) )」の箇所を、例えば「do - parallel ( r 0 ( - N R 1 ) )」に置換することで、最適化 S C o P コード 3 0 を生成する。「do - parallel」の記述はループの並列化を表す。

## 【 0 0 2 5 】

また、例えば、処理部 1 2 は、S C o P コード 2 0 の文 s 1 を別のループに分離するループ分割の最適化を適用し、それぞれ外側ループで並列化した最適化 S C o P コード 3 1 を生成する。この場合、処理部 1 2 は、S C o P コード 2 0 の「do ( r 0 ( - N R 1 ) )」の箇所を、「do - parallel ( r 0 ( - N R 1 ) )」に置換する。更に、処理部 1 2 は、「do ( c 0 ( - N C 1 ) )」の直前の行に、「do - parallel ( r 0 ( - N R 1 ) )」を挿入することで、最適化 S C o P コード 3 1 を生成する。

40

## 【 0 0 2 6 】

また、処理部 1 2 は、例えば、ループ分割とループ交換との両方を用いた最適化を行うなどの他のループ最適化を S C o P コード 2 0 に適用した最適化 S C o P コード 3 2 を生

50

成し得る。このように、処理部 12 は、複数パターンの最適化 S C o P コード 3 0 , 3 1 , 3 2 を生成する。

【 0 0 2 7 】

処理部 12 は、疎行列情報 4 0 と、式情報 4 1 と、データ型情報 4 2 とに基づいて、複数の最適化 S C o P コードを複数のソースコード候補に変換する（ステップ S 2）。疎行列情報 4 0 は、処理対象の疎行列の非ゼロの要素を表す変数を示す情報である。当該変数は、目的のソースコードで使用される変数である。処理部 12 は、当該変数の変数名には、各最適化 S C o P コードに記述される変数名と同じものを用いることができる。また、後述されるように、処理部 12 は、疎行列情報 4 0 に基づいて各最適化 S C o P コードには含まれない変数を追加し得る。

10

【 0 0 2 8 】

式情報 4 1 は、最適化 S C o P コードに含まれる関数に対応する演算式を示す情報である。例えば、式情報 4 1 は、最適化 S C o P コード 3 0 , 3 1 , 3 2 それぞれに含まれる代入文の右辺の関数  $f_0$  ,  $f_1$  に係る式を示す。本例では、式情報 4 1 は、関数  $f_0$  を 0 に対応付ける情報を含む。また、式情報 4 1 は、関数  $f_1$  を、 $f_1$  の第 1 引数に、第 2 引数と第 3 引数との積を加算する式に対応付ける情報を含む。ここで、S C o P コード 2 0 および最適化 S C o P コード 3 0 , 3 1 , 3 2 における関数  $f_1$  の第 1 引数は  $r v$  であり、第 2 引数は  $M$  であり、第 3 引数は  $v$  である。ステップ S 2 の変換の際、処理部 12 は、二次元配列  $M$  を、疎行列に対応する一次元配列（例えば、配列  $S M$ ）に変換する。データ型情報 4 2 は、最適化 S C o P コードに含まれる配列やインデックスなどの、目的のソースコードにおける変数に対応する型を示す情報である。疎行列情報 4 0、式情報 4 1 およびデータ型情報 4 2 は、ユーザが必要とする疎行列処理に応じて予め作成され、記憶部 11 に格納される。

20

【 0 0 2 9 】

ステップ S 1 で生成される最適化 S C o P コード 3 0 , 3 1 , 3 2 は、S C o P 形式で記述されており、ユーザが利用するプログラミング言語での疎行列処理に対応する記述ではない。そこで、処理部 12 は、疎行列情報 4 0、式情報 4 1 およびデータ型情報 4 2 に基づいて、最適化 S C o P コードを該当のプログラミング言語の記述に変換することで、利用する疎行列の表現が反映されたソースコードの候補を得る。本例では、プログラミング言語として C を例示する。利用するプログラミング言語の情報は、S C o P コード 2 0 に含まれる。例えば、S C o P コード 2 0 の「( language c)」の記述がプログラミング言語の情報に相当する。したがって最適化 S C o P コード 3 0 , 3 1 , 3 2 も当該プログラミング言語の情報を含む。

30

【 0 0 3 0 】

ここで、非ゼロの値とともにゼロの値を明示的に保持するデータ構造は密行列と言われる。上記の配列  $M$  は、密行列のデータ構造であると言える。疎行列は、密行列からゼロの要素を削除したデータ構造となる。疎行列では、非ゼロの要素が元の密行列のどの位置に存在したかを表す付加情報が必要となる。付加情報による疎行列の表現方法は様々であり、当該表現方法の違いが疎行列のフォーマットの違いとなる。使用するフォーマットによって、ソースコードの記述内容は大きく変わる。

40

【 0 0 3 1 】

例えば、二次元の行列に対する疎行列表現のフォーマットには、C S R (Compressed Sparse Row) フォーマット、C S C (Compressed Sparse Column) フォーマットおよび C O O (Coordinate) フォーマットなどがある。C S R フォーマットは、非ゼロの要素を行方向に圧縮して保持するとともに、各要素の列情報を保持する形式である。C S C フォーマットは、非ゼロの要素を列方向に圧縮して保持するとともに、各要素の行情報を保持する形式である。C O O フォーマットは、非ゼロの要素に対して、行情報および列情報を保持する形式である。疎行列情報 4 0 は、こうした疎行列の非ゼロの要素を表す何れかのフォーマットに対応して使用される変数および変数間の依存関係を示す。疎行列情報 4 0 は、使用するフォーマットに応じて予め作成される。

50

## 【 0 0 3 2 】

例えば、処理部 1 2 は、ステップ S 2 の変換により、最適化 S C o P コード 3 0 に対して C S R フォーマットの疎行列情報 4 0 が適用されたソースコード候補 5 0 を得る。ソースコード候補 5 0 では、行列 A に対して用いられていた二次元配列 M は、一次元配列 S M に変換されている。C S R フォーマットの場合、疎行列情報 4 0 には、行番号の変数 r、r に対しループの開始「0」および終了「NR」を指定可能であること、および、行番号 r の行に含まれる非ゼロの要素の配列 S M における位置を示す変数（例えば、index）が定められる。また、疎行列情報 4 0 には、非ゼロの要素の列番号を保持する配列（例えば、col\_index[index]）の値を代入する変数 c が定められる。更に、疎行列情報 4 0 には、S M における行 r の先頭位置を保持する配列（例えば、row\_ptr[r]）により表される上記 index のループの開始と終了の値が定められる。なお、行 r は、行番号 r の行を示す。ソースコード候補 5 0 に含まれる変数 r、c、index や配列 r v、S M、v、row\_ptr、col\_index などに対して用いる型は、データ型情報 4 2 に予め登録されている。なお、ソースコード候補 5 0 において、配列の定義部分の図示は省略されている。

10

## 【 0 0 3 3 】

更に、処理部 1 2 は、最適化 S C o P コード 3 0 における「do-parallel(r 0 (- NR 1))」の記述に基づいて、当該記述箇所に対応する位置に、並列化指示文を挿入する。ソースコード候補 5 0 は、OpenMP (Open Multi-Processing、登録商標) ディレクティブ「#pragma omp parallel for」が挿入される例を示す。

20

## 【 0 0 3 4 】

同様に、処理部 1 2 は、ステップ S 2 の変換により、最適化 S C o P コード 3 1 に対してソースコード候補 5 1 を得る。更に、処理部 1 2 は、ステップ S 2 の変換により、最適化 S C o P コード 3 2 に対してソースコード候補 5 2 を得る。

## 【 0 0 3 5 】

処理部 1 2 は、複数のソースコード候補それぞれを用いた場合の疎行列に対する処理の性能を評価する。処理部 1 2 は、当該性能の評価に応じて複数のソースコード候補の中からソースコード 6 0 を選択する（ステップ S 3）。ソースコード 6 0 は、情報処理装置 1 0 により最終的に出力される最適化済のソースコードである。選択されるソースコード 6 0 の数は、1 つでもよいし、複数でもよい。

30

## 【 0 0 3 6 】

例えば、処理部 1 2 は、各ソースコード候補をコンパイルして実行可能コードを生成し、当該実行可能コードにより疎行列を処理し、処理時間を計測することで、性能を評価する。この場合、処理部 1 2 は、処理時間が短い実行可能コードに対応するソースコード候補を優先して選択する。

## 【 0 0 3 7 】

あるいは、処理部 1 2 は、ソースコード候補と疎行列との特徴に対して処理時間などの性能評価結果の指標を出力する機械学習モデルを用いて、各ソースコード候補および実際の疎行列に対する性能評価を行ってもよい。この場合、処理部 1 2 は、性能評価結果の指標が良い実行可能コードに対応するソースコード候補を優先して選択する。

40

## 【 0 0 3 8 】

情報処理装置 1 0 によれば、行列に対するループ処理が静的制御部形式で記述された第 1 コードを凸多面体モデルにより最適化することで複数の第 2 コードが取得される。疎行列の非ゼロの要素を表す変数を示す疎行列情報と、第 2 コードに含まれる関数に対応する演算式を示す式情報と、変数に対して使用する型を示すデータ型情報とに基づいて、複数の第 2 コードが複数のソースコード候補に変換される。複数のソースコード候補それぞれを用いた場合の疎行列に対する処理性能の評価に応じて複数のソースコード候補の中からソースコードが選択される。

## 【 0 0 3 9 】

50



これにより、情報処理装置 10 は、疎行列処理に対して最適化されたソースコードを効率的に得ることができる。

ここで、疎行列処理のソースコードには、ポインタの使用やデータ間接参照が含まれるため、コンパイラで最適化することは難しい。そこで、ソースコードに記述され得る各種疎行列処理のアルゴリズムに対して、最適化済のライブラリを事前に用意しておき、当該ライブラリを利用することがある。しかし、事前に用意されたライブラリでは、実行時コンパイル手法に対処できない。また、事前に用意されたライブラリが、プログラム記述者により利用される疎行列処理のデータ構造やアルゴリズムに合わない場合、当該ライブラリによる最適化を適用することはできない。また、このような場合に追従して、事前に用意されたライブラリを更新していくことも考えられるが、ライブラリの更新の手間もかかる。

#### 【0040】

そこで、情報処理装置 10 は、疎行列処理に対して最適化済のソースコード 60 を自動生成する。特に、情報処理装置 10 は、ソースコードを直接、最適化コードに変換するのではなく、疎行列処理のアルゴリズムを記述した S C O P コード 20 を最適化し、複数の最適化 S C O P コードを得る。これにより、情報処理装置 10 は、疎行列処理を記述したソースコードには適用することができなかつた、凸多面体モデルを利用したループ最適化を利用できるようになる。情報処理装置 10 は、凸多面体最適化を行う既存のツールを活用することで、凸多面体モデルによるループ最適化を容易に利用できる。

#### 【0041】

また、情報処理装置 10 は、疎行列フォーマット、データ型および疎行列の性質に応じた疎行列情報 40、式情報 41 およびデータ型情報 42 に基づいて、複数の最適化 S C O P コードを所定のプログラミング言語で記述された複数のソースコード候補に変換する。そして、情報処理装置 10 は、各ソースコード候補を用いた場合の実際の疎行列に対する処理性能の評価に応じて、ソースコード候補の中から何れかをソースコード 60 として選択する。

#### 【0042】

これにより、情報処理装置 10 は、ユーザの環境に合った最適なソースコード 60 を、効率的に得ることができる。例えば、S C O P コード 20 において、データ型や右辺式の具体的な情報を除くことで、疎行列で利用するデータ型や代入文の右辺の形式が変わった場合でも、凸多面体モデルによるループの最適化効果を容易に得ることができる。更に、情報処理装置 10 は、最適化済のソースコード 60 をコンパイルすることで得られた実行可能コードを実行して疎行列処理を行うことで、R A M などのメモリと C P U 間のデータ転送量を減らし、当該疎行列処理の高速化を図れる。

#### 【0043】

以下では、より具体的な例を示して情報処理装置 10 の機能を更に詳細に説明する。

#### [ 第 2 の実施の形態 ]

次に、第 2 の実施の形態を説明する。

#### 【0044】

図 2 は、第 2 の実施の形態の情報処理装置のハードウェア例を示す図である。

情報処理装置 100 は、疎行列処理に対する最適なソースコードの生成を行う。一例として、プログラミング言語は C であるものとする。ただし、プログラミング言語は C 以外の他のプログラミング言語でもよい。

#### 【0045】

情報処理装置 100 は、C P U 101、R A M 102、H D D 103、G P U 104、入力インタフェース 105、媒体リーダー 106 および N I C (Network Interface Card) 107 を有する。なお、C P U 101 は、第 1 の実施の形態の処理部 12 の一例である。R A M 102 または H D D 103 は、第 1 の実施の形態の記憶部 11 の一例である。

#### 【0046】

10

20

30

40

50

C P U 1 0 1 は、プログラムの命令を実行するプロセッサである。C P U 1 0 1 は、H D D 1 0 3 に記憶されたプログラムやデータの少なくとも一部を R A M 1 0 2 にロードし、プログラムを実行する。なお、C P U 1 0 1 は複数のプロセッサコアを含んでもよい。また、情報処理装置 1 0 0 は複数のプロセッサを有してもよい。以下で説明する処理は複数のプロセッサまたはプロセッサコアを用いて並列に実行されてもよい。また、複数のプロセッサの集合を「マルチプロセッサ」または単に「プロセッサ」と言うことがある。

【 0 0 4 7 】

R A M 1 0 2 は、C P U 1 0 1 が実行するプログラムや C P U 1 0 1 が演算に用いるデータを一時的に記憶する揮発性の半導体メモリである。なお、情報処理装置 1 0 0 は、R A M 以外の種類のメモリを備えてもよく、複数個のメモリを備えてもよい。

10

【 0 0 4 8 】

H D D 1 0 3 は、O S ( Operating System ) やミドルウェアやアプリケーションソフトウェアなどのソフトウェアのプログラム、および、データを記憶する不揮発性の記憶装置である。なお、情報処理装置 1 0 0 は、フラッシュメモリや S S D ( Solid State Drive ) などの他の種類の記憶装置を備えてもよく、複数の不揮発性の記憶装置を備えてもよい。

【 0 0 4 9 】

G P U 1 0 4 は、C P U 1 0 1 からの命令に従って、情報処理装置 1 0 0 に接続されたディスプレイ 7 1 に画像を出力する。ディスプレイ 7 1 としては、C R T ( Cathode Ray Tube ) ディスプレイ、液晶ディスプレイ ( L C D : Liquid Crystal Display ) 、

20

【 0 0 5 0 】

入力インタフェース 1 0 5 は、情報処理装置 1 0 0 に接続された入力デバイス 7 2 から入力信号を取得し、C P U 1 0 1 に出力する。入力デバイス 7 2 としては、マウス、タッチパネル、タッチパッド、トラックボールなどのポインティングデバイス、キーボード、リモートコントローラ、ボタンスイッチなどを用いることができる。また、情報処理装置 1 0 0 に、複数の種類の入力デバイスが接続されていてもよい。

【 0 0 5 1 】

媒体リーダ 1 0 6 は、記録媒体 7 3 に記録されたプログラムやデータを読み取る読み取り装置である。記録媒体 7 3 として、例えば、磁気ディスク、光ディスク、光磁気ディスク ( M O : Magneto-Optical disk ) 、半導体メモリなどを使用できる。磁気ディスクには、フレキシブルディスク ( F D : Flexible Disk ) や H D D が含まれる。光ディスクには、C D ( Compact Disc ) や D V D ( Digital Versatile Disc ) が含まれる。

30

【 0 0 5 2 】

媒体リーダ 1 0 6 は、例えば、記録媒体 7 3 から読み取ったプログラムやデータを、R A M 1 0 2 や H D D 1 0 3 などの他の記録媒体にコピーする。読み取られたプログラムは、例えば、C P U 1 0 1 によって実行される。なお、記録媒体 7 3 は可搬型記録媒体であってもよく、プログラムやデータの配布に用いられることがある。また、記録媒体 7 3 や H D D 1 0 3 を、コンピュータ読み取り可能な記録媒体と言うことがある。

40

【 0 0 5 3 】

N I C 1 0 7 は、ネットワーク 7 4 に接続され、ネットワーク 7 4 を介して他のコンピュータと通信を行うインタフェースである。N I C 1 0 7 は、例えば、スイッチやルータなどの通信装置とケーブルで接続される。N I C 1 0 7 は、無線通信を行うインタフェースでもよい。

【 0 0 5 4 】

図 3 は、行列演算のソースコード例を示す図である。

ソースコード P 1 1 , P 1 2 は、行列ベクトル積  $y = A * x$  の記述例を示す。A は N R 行 N C 列の行列である。N R および N C は何れも 2 以上の整数である。x は N C 行の列ベクトルである。y は N R 行の列ベクトルである。図 3 の例では、N R = 4 であり、N C =

50

6である。ソースコードP11は、密行列に対する記述例である。ソースコードP12は、疎行列に対する記述例である。ソースコードP12は、CSRフォーマットに対応するソースコードである。なお、疎行列に対するベクトル積は、疎行列ベクトル積 (Sparse Matrix-Vector multiplication) と呼ばれる。

#### 【0055】

行列Aは、ゼロを比較的多く含む。図3において、行列Aの空欄で示される箇所は、行列Aのゼロの要素を示す。行列Aの非ゼロの値が記載された箇所は、行列Aの非ゼロの要素を示す。CSRフォーマットの例では、行列Aの非ゼロの要素が行方向に圧縮されて、すなわち、ゼロを省略して、一次元の配列 (例えば、val) に保持される。行列Aの非ゼロの要素数は7なので、配列valの要素数は7となる。また、行列Aの各行における先頭の要素に対応する、配列valのインデックスが一次元の配列 (例えば、rowptr) に保持される。rowptrのインデックスは、行列Aの行番号iとなる。更に、配列valに保持される要素に対して、行列Aにおける当該要素の列番号が、一次元の配列 (例えば、col) に保持される。配列colのインデックスは、配列valのインデックスと同じである。また、rowptrは、最後の要素として非ゼロ要素の総数を保持する。rowptr[i+1]で最後の行を超えると、最後の行の終わりの位置を認識するためである。

#### 【0056】

ソースコードP12では、疎行列のデータ構造を用いることで、ソースコードP11で用いられる密行列よりも、配列に保持するデータ量を削減でき、RAM102とCPU101との間のデータ転送量を削減し得る。しかし、疎行列処理のソースコードには、ポインタの使用やデータ間接参照が含まれるため、コンパイラで最適化することは難しい。例えば、疎行列のゼロまたは非ゼロの値の分布により、各ループで処理される要素の数が異なると、コンパイラによる最適化では十分に処理性能の向上を図れないこともある。そこで、情報処理装置100は、疎行列処理に対して最適化されたソースコードを自動的に生成する機能を提供する。

#### 【0057】

図4は、情報処理装置の機能例を示す図である。

情報処理装置100は、記憶部110、凸多面体最適化部120、コード生成部130および最適化プログラム選択部140を有する。記憶部110には、RAM102やHDD103の記憶領域が用いられる。また、CPU101は、RAM102に記憶されたプログラムを実行することで、凸多面体最適化部120、コード生成部130および最適化プログラム選択部140の機能を発揮する。

#### 【0058】

記憶部110は、凸多面体最適化部120、コード生成部130および最適化プログラム選択部140の処理に用いられる各種のデータを記憶する。記憶部110は、アルゴリズムSCOP情報200、最適化SCOP情報セット210、疎行列情報220、右辺式情報230、データ型情報240、疎行列特殊化情報250、最適化戦略指示情報260、最適化プログラムコード候補セット270、疎行列データ情報280および最適化プログラムコードセット290を含む。

#### 【0059】

アルゴリズムSCOP情報200は、行列ベクトル積のアルゴリズムがSCOP形式で記述された情報である。アルゴリズムSCOP情報200では、行列ベクトル積のアルゴリズムは、密行列に対するループ処理によって記述される。アルゴリズムSCOP情報200では、データ型情報が省略され、また、代入文の右辺の具体的な計算方法が、関数名だけ記述することで抽象化されて省略されている。アルゴリズムSCOP情報200は、第1の実施の形態のSCOPコード20、すなわち、第1コードの一例である。

#### 【0060】

最適化SCOP情報セット210は、アルゴリズムSCOP情報200が凸多面体最適化部120により最適化された結果である最適化SCOP情報の集合である。最適化SC

10

20

30

40

50

○ P 情報は、第 1 の実施の形態の最適化 S C o P コード 3 0 , 3 1 , 3 2 の一例である。

【 0 0 6 1 】

疎行列情報 2 2 0 は、疎行列の非ゼロの要素を表す変数を示す。より具体的には、疎行列情報は、疎行列のフォーマットに応じた、行列 A の非ゼロの要素の表現に用いられる複数の変数や変数間の依存関係を示す情報である。

【 0 0 6 2 】

右辺式情報 2 3 0 は、最適化 S C o P 情報で省略された代入文の右辺式を、目的のソースコードにおける記述に変換するための情報である。右辺式情報 2 3 0 は、第 1 の実施の形態の式情報 4 1 の一例である。

【 0 0 6 3 】

データ型情報 2 4 0 は、目的のソースコードにおける変数の型を示す情報である。目的のソースコードにおける変数名としては、最適化 S C o P 情報に含まれる変数名が使用される。ただし、最適化 S C o P 情報における密行列の配列は、疎行列の配列に置換される。

【 0 0 6 4 】

疎行列特殊化情報 2 5 0 は、疎行列に含まれる非ゼロの要素の値域（取り得る値）や非ゼロの要素の数などに応じて、変数の型を特殊化するための情報である。例えば、非ゼロの要素が特定の値のみである場合に疎行列の要素を当該特定の値のみで表す、非ゼロの要素の数が比較的少ない場合に配列のインデックスの型をサイズの小さい型にするなどのデータ特殊化が行われ得る。

【 0 0 6 5 】

最適化戦略指示情報 2 6 0 は、並列化やデータ特殊化などの使用する最適化手法をコード生成部 1 3 0 に指示するための情報である。

最適化プログラムコード候補セット 2 7 0 は、最適化 S C o P 情報セット 2 1 0 の各要素、すなわち、最適化 S C o P 情報がコード生成部 1 3 0 により変換されて得られた最適化プログラムコード候補の集合である。最適化プログラムコード候補は、第 1 の実施の形態のソースコード候補 5 0 , 5 1 , 5 2 の一例である。

【 0 0 6 6 】

疎行列データ情報 2 8 0 は、実際に使用される疎行列を示す情報である。

最適化プログラムコードセット 2 9 0 は、最適化プログラム選択部 1 4 0 により最適化プログラムコード候補セット 2 7 0 の中から選択された最適化プログラムコードの集合である。最適化プログラムコードは、第 1 の実施の形態のソースコード 6 0 の一例である。

【 0 0 6 7 】

凸多面体最適化部 1 2 0 は、凸多面体モデルを利用してアルゴリズム S C o P 情報 2 0 0 に種々のループ最適化を適用することで、最適化 S C o P 情報セット 2 1 0 を生成する。凸多面体最適化部 1 2 0 は、凸多面体モデルによる最適化を行うツールである P o l l y、P L U T E および G r a p h i t e などを利用して、最適化 S C o P 情報セット 2 1 0 を生成してもよい。

【 0 0 6 8 】

コード生成部 1 3 0 は、疎行列情報 2 2 0、右辺式情報 2 3 0 およびデータ型情報 2 4 0 に基づいて最適化 S C o P 情報セット 2 1 0 の各要素を、最適化プログラムコード候補に変換することで、最適化プログラムコード候補セット 2 7 0 を生成する。最適化プログラムコード候補は、目的のプログラミング言語で記述されたソースコードの候補である。本例では、前述のように当該プログラミング言語を C とする。コード生成部 1 3 0 は、疎行列特殊化情報 2 5 0 や最適化戦略指示情報 2 6 0 に基づいて、最適化 S C o P 情報セット 2 1 0 の各要素を、最適化プログラムコード候補に変換してもよい。

【 0 0 6 9 】

最適化プログラム選択部 1 4 0 は、最適化プログラムコード候補セット 2 7 0 の各要素、すなわち、最適化プログラムコード候補を用いた場合の、疎行列データ情報 2 8 0 に対する処理性能の評価を行う。最適化プログラム選択部 1 4 0 は、当該処理性能の評価に

10

20

30

40

50

じて、最適化プログラムコード候補セット 270の中から、最適化プログラムコードセット 290を選択する。

【0070】

例えば、最適化プログラム選択部 140は、各最適化プログラムコード候補をコンパイルして実行可能コードを生成し、当該実行可能コードにより疎行列データ情報 280を処理し、処理時間を計測することで、処理性能を評価する。この場合、例えば、最適化プログラム選択部 140は、処理時間が短い実行可能コードに対応する最適化プログラムコード候補を優先して所定数選択する。

【0071】

あるいは、最適化プログラム選択部 140は、機械学習モデルを用いて、実際の疎行列および各最適化プログラムコード候補に対する性能評価を行ってもよい。最適化プログラム選択部 140は、機械学習モデルとして、最適化プログラムコード候補と疎行列データ情報 280との特徴に対して処理時間などの性能評価結果の指標を出力するモデルを使用する。この場合、例えば、最適化プログラム選択部 140は、性能評価結果の指標が良い実行可能コードに対応する最適化プログラムコード候補を優先して所定数選択する。

10

【0072】

図 5は、情報処理装置で処理されるデータの例を示す図である。

前述のように、アルゴリズム S C o P 情報 200は、凸多面体最適化部 120の入力である。最適化 S C o P 情報セット 210は、凸多面体最適化部 120の出力である。

【0073】

最適化 S C o P 情報セット 210、疎行列情報 220、右辺式情報 230、データ型情報 240、疎行列特殊化情報 250および最適化戦略指示情報 260は、コード生成部 130の入力である。最適化プログラムコード候補セット 270は、コード生成部 130の出力である。

20

【0074】

最適化プログラムコード候補セット 270および疎行列データ情報 280は最適化プログラム選択部 140の入力である。最適化プログラムコードセット 290は、最適化プログラム選択部 140の出力である。

【0075】

アルゴリズム S C o P 情報 200は、ユーザが必要とする疎行列処理に応じて予め作成され、情報処理装置 100に入力される。疎行列情報 220、右辺式情報 230、データ型情報 240、疎行列特殊化情報 250、最適化戦略指示情報 260および疎行列データ情報 280は、記憶部 110に予め格納される。

30

【0076】

次に、情報処理装置 100の処理手順を説明する。

図 6は、情報処理装置の処理例を示すフローチャートである。

(S10)凸多面体最適化部 120は、アルゴリズム S C o P 情報 200の入力を受け付ける。

【0077】

(S11)凸多面体最適化部 120は、アルゴリズム S C o P 情報 200に対して凸多面体最適化を行い、凸多面体最適化の結果として、最適化 S C o P 情報セット 210を取得する。

40

【0078】

(S12)コード生成部 130は、最適化 S C o P 情報セットの各要素の利用可能性判定を行う。コード生成部 130は、利用可能性判定を行うことで、疎行列情報 220に対して利用できないことが明らかな要素を最適化 S C o P 情報セット 210の中から予め除外する。具体的には、コード生成部 130は、最適化 S C o P 情報セット 210の各要素 Xについて、疎行列情報 220に合わない場合は、利用不可能として要素 Xを破棄する。コード生成部 130は、利用可能である場合は、要素 Xを集合 S E Tに追加する。利用可能性判定の詳細は後述される。

50

## 【 0 0 7 9 】

( S 1 3 ) コード生成部 1 3 0 は、最適化プログラムコード候補セット生成を行う。コード生成部 1 3 0 は、集合 S E T の各要素 Y を、疎行列情報 2 2 0、右辺式情報 2 3 0 およびデータ型情報 2 4 0 に基づいて、最適化プログラムコード候補に変換することで、最適化プログラムコード候補セット 2 7 0 を生成する。最適化プログラムコード候補セット生成の詳細は後述される。

## 【 0 0 8 0 】

( S 1 4 ) 最適化プログラム選択部 1 4 0 は、最適化プログラムコード候補セット 2 7 0 の各要素 Z を用いた場合の、疎行列データ情報 2 8 0 に対する処理性能の評価を行う。最適化プログラム選択部 1 4 0 は、当該処理性能の評価に応じて、最適化プログラムコード候補セット 2 7 0 の中から、最適化プログラムコードセット 2 9 0 を選択する。

10

## 【 0 0 8 1 】

例えば、最適化プログラム選択部 1 4 0 は、疎行列データ情報 2 8 0 を利用して、実際に各要素 Z をコンパイルして実行し、実行時間が閾値よりも短い要素を残し、それ以外の要素を破棄することで、最適化プログラムコードセット 2 9 0 を取得してもよい。

## 【 0 0 8 2 】

また、最適化プログラム選択部 1 4 0 は、各要素 Z の記述内容および疎行列データ情報 2 8 0 の特徴を利用した機械学習モデルによって、各要素 Z の性能を予測してもよい。この場合、最適化プログラム選択部 1 4 0 は、機械学習モデルが出力する各要素の性能評価結果を示す指標に基づいて、見込みの高い要素を残し、それ以外の要素を破棄することで、最適化プログラムコードセット 2 9 0 を取得してもよい。

20

## 【 0 0 8 3 】

( S 1 5 ) 最適化プログラム選択部 1 4 0 は、最適化プログラムコードセット 2 9 0 を出力する。例えば、最適化プログラム選択部 1 4 0 は、最適化プログラムコードセット 2 9 0 をディスプレイ 7 1 に表示させてもよい。最適化プログラム選択部 1 4 0 は、ネットワーク 7 4 を介して、他のコンピュータに最適化プログラムコードセット 2 9 0 を送信してもよい。

## 【 0 0 8 4 】

次に、ステップ S 1 0 , S 1 1 における具体的な入出力例を説明する。まず、アルゴリズム S C O P 情報 2 0 0 を説明する。アルゴリズム S C O P 情報 2 0 0 は、密行列に対する行列ベクトル積プログラムのアルゴリズムに応じて予め作成される。一例として、行列ベクトル積  $y = A * x$  を示す。

30

## 【 0 0 8 5 】

図 7 は、行列ベクトル積プログラムの例を示す図である。

行列ベクトル積プログラム 3 0 1 は、密行列に対する行列ベクトル積の記述例である。例えば、行列 A の要素は二次元配列 M に保持される。列ベクトル x の要素は、配列 v に保持される。列ベクトル y の要素は、配列 r v に保持される。行列ベクトル積プログラム 3 0 1 の記述は比較的短いが、実際の H P C アプリケーションでは、この部分が実行時間の多く（例えば 8 0 % 以上など）を占める場合がある。

## 【 0 0 8 6 】

図 8 は、疎行列ベクトル積プログラムの例を示す図である。

疎行列ベクトル積プログラム 3 0 2 は、C S R フォーマットの疎行列に対する疎行列ベクトル積の記述例である。例えば、疎行列の非ゼロの値は、一次元配列 S M に保持される。当該配列 S M や配列 v のインデックスは、配列 r o w \_ p t r や配列 c o l \_ i n d e x によって間接的に表されている。

40

## 【 0 0 8 7 】

例えば、ユーザは、情報処理装置 1 0 0 にソースコードを自動生成させる際、疎行列ベクトル積プログラム 3 0 2 を予め記述しなくてよい。その代わりに、ユーザは、C S R フォーマットで利用する変数 r , c , i n d e x の情報や配列 r , r v , S M などの型を疎行列情報 2 2 0 やデータ型情報 2 4 0 として情報処理装置 1 0 0 に入力すればよい。

50

## 【 0 0 8 8 】

また、ユーザは、行列ベクトル積プログラム 3 0 1 のアルゴリズムを抽象化したアルゴリズム S C o P 情報 2 0 0 を作成して、情報処理装置 1 0 0 に入力する。

図 9 は、アルゴリズム S C o P 情報の例を示す図である。

## 【 0 0 8 9 】

アルゴリズム S C o P 情報 2 0 0 では、行列ベクトル積プログラム 3 0 1 におけるデータ型情報や代入文の右辺の具体的な計算方法は、抽象化されて省略されている。具体的には、省略された計算方法の箇所には、「f 0」や「f 1」といった関数名と関数の引数のみが記述されている。

## 【 0 0 9 0 】

例えば、アルゴリズム S C o P 情報 2 0 0 の 1 行目は、使用するプログラミング言語（本例では C）を指定する文である。アルゴリズム S C o P 情報 2 0 0 の 2 ~ 3 行目は、行列 A の行数を示す変数 N R および列数を示す変数 N C の定義である。アルゴリズム S C o P 情報 2 0 0 の 4 ~ 6 行目は、配列 r v , M , v および当該配列のインデックスの定義である。「array」は配列を示す。例えば、「array ( r v N R )」の記述は、要素数が N R 個の一次元配列 r v の定義である。「array ( M N R N C )」の記述は、要素数が N R \* N C 個の二次元配列 M の定義である。アルゴリズム S C o P 情報 2 0 0 の 7 ~ 1 0 行目は、抽象化されたアルゴリズムの記述である。「do」は、ループを示す。「s 1」や「s 2」は代入文の識別子である。

## 【 0 0 9 1 】

凸多面体最適化部 1 2 0 は、アルゴリズム S C o P 情報 2 0 0 に対して凸多面体最適化を行うことで、例えば、次のような複数パターンのお最適化 S C o P 情報を生成する。

図 1 0 は、最適化 S C o P 情報の第 1 の例を示す図である。

## 【 0 0 9 2 】

最適化 S C o P 情報 2 1 1 は、アルゴリズム S C o P 情報 2 0 0 に比べて、入力データやループの形式は変化していないが、凸多面体最適化部 1 2 0 により外側のループが並列化可能と判定された結果である。7 行目の記述「do - parallel」は、ループの並列化を示す。

## 【 0 0 9 3 】

図 1 1 は、最適化 S C o P 情報の第 2 の例を示す図である。

最適化 S C o P 情報 2 1 2 は、文 s 1 を別のループに分離するループ分割の最適化が適用され、トップレベルの 2 つのループが凸多面体最適化部 1 2 0 により並列化可能と判定された結果である。

## 【 0 0 9 4 】

図 1 2 は、最適化 S C o P 情報の第 3 の例を示す図である。

最適化 S C o P 情報 2 1 3 は、凸多面体最適化部 1 2 0 により最適化 S C o P 情報 2 1 2 と同様のループ分割の最適化が適用され、更に文 s 2 を含むループに対して、ループ交換の最適化が適用された結果である。加えて、トップレベルの最初のループと 2 つ目のループの内側が凸多面体最適化部 1 2 0 により並列化可能と判定されている。

## 【 0 0 9 5 】

最適化 S C o P 情報 2 1 1 , 2 1 2 , 2 1 3 は、最適化 S C o P 情報セット 2 1 0 の要素である。次に、最適化 S C o P 情報セット 2 1 0 の各要素に対する利用可能性判定の手順を説明する。

## 【 0 0 9 6 】

図 1 3 は、利用可能性判定例を示すフローチャートである。

利用可能性判定は、ステップ S 1 2 に相当する。

( S 2 0 ) コード生成部 1 3 0 は、疎行列情報 2 2 0 の各項目の変数の依存関係を検出し、変数の利用可能順序を決定する。

## 【 0 0 9 7 】

( S 2 1 ) コード生成部 1 3 0 は、最適化 S C o P 情報セット 2 1 0 の中から処理対象

10

20

30

40

50

とする最適化 S C o P 情報 X を選択する。

( S 2 2 ) コード生成部 1 3 0 は、最適化 S C o P 情報 X 内から疎行列の元になる密行列を参照している文 S を含むループ L の構造を検出する。

【 0 0 9 8 】

( S 2 3 ) コード生成部 1 3 0 は、ステップ S 2 0 で決定した利用可能順序と、ステップ S 2 2 で検出したループ L の構造とが一致するか否かを判定する。一致する場合、コード生成部 1 3 0 は、ステップ S 2 4 に処理を進める。一致しない場合、コード生成部 1 3 0 は、ステップ S 2 7 に処理を進める。なお、コード生成部 1 3 0 は、ステップ S 2 0 で決定した利用可能順序が特別ループの作成を意味する場合は、ステップ S 2 3 で一致すると判定し、ステップ S 2 4 に処理を進める。変数の利用可能順序が特別ループの作成を意味する場合の具体例は後述される。

10

【 0 0 9 9 】

( S 2 4 ) コード生成部 1 3 0 は、ループ L に文 S 以外の文があるか否かを判定する。ループ L に文 S 以外の文がある場合、コード生成部 1 3 0 は、ステップ S 2 5 に処理を進める。ループ L に文 S 以外の文がない場合、コード生成部 1 3 0 は、ステップ S 2 6 に処理を進める。

【 0 1 0 0 】

( S 2 5 ) コード生成部 1 3 0 は、ループ L の文 S 以外の文のそれぞれについて、使用する変数を利用可能であるか否かを判定する。ループ L の文 S 以外の文のそれぞれについて使用する変数を利用可能である場合、コード生成部 1 3 0 は、ステップ S 2 6 に処理を進める。ループ L の文 S 以外の何れかの文で使用する変数を利用可能でない場合、すなわち、利用できない変数が存在する場合、コード生成部 1 3 0 は、ステップ S 2 7 に処理を進める。

20

【 0 1 0 1 】

( S 2 6 ) コード生成部 1 3 0 は、最適化 S C o P 情報 X を利用可能と判定し、集合 S E T に追加する。そして、コード生成部 1 3 0 は、ステップ S 2 8 に処理を進める。

( S 2 7 ) コード生成部 1 3 0 は、最適化 S C o P 情報 X を利用不可能と判定し、破棄する。そして、コード生成部 1 3 0 は、ステップ S 2 8 に処理を進める。

【 0 1 0 2 】

( S 2 8 ) コード生成部 1 3 0 は、最適化 S C o P 情報セット 2 1 0 の全要素を処理済であるか否かを判定する。最適化 S C o P 情報セット 2 1 0 の全要素を処理済の場合、コード生成部 1 3 0 は、利用可能性判定を終了する。最適化 S C o P 情報セット 2 1 0 の全要素を処理済でない場合、コード生成部 1 3 0 は、ステップ S 2 1 に処理を進める。

30

【 0 1 0 3 】

次に、利用可能性判定の具体例を説明する。まず、疎行列情報 2 2 0 の例を説明する。疎行列情報 2 2 0 としては、C S R フォーマット、C S C フォーマットおよび C O O フォーマットなどの使用するフォーマットに応じた情報が記憶部 1 1 0 に予め格納される。一例として、C S R フォーマットを用いる場合の疎行列情報 2 2 0 を例示する。

【 0 1 0 4 】

図 1 4 は、疎行列情報 ( C S R ) の例を示す図である。

40

疎行列情報 2 2 0 は、疎行列の表現に C S R フォーマットを用いる場合のインデックス番号と、疎行列の行番号と列番号との関係を示す。当該インデックス番号は、行番号から疎行列の非ゼロの要素および当該要素の列番号の取得に用いられる変数である。インデックス番号は、ループを制御するループ変数としても用いられ得る。

【 0 1 0 5 】

疎行列情報 2 2 0 は、項目、変数、開始、終了および取得方法の項目を含む。項目の項目には、変数により表される内容が登録される。変数の項目には、該当の内容に対してソースコードで用いる変数名が登録される。開始の項目には、該当の変数に対応するループの開始の値が登録される。終了の項目には、該当の変数に対応するループの終了の値が登録される。ただし、値の範囲を明示的に定められない場合、開始および終了の項目は設定

50



なしとなる。図中、設定なしを、ハイフン「-」で示す。取得方法の項目には、開始および終了の項目が設定なしの場合、該当の変数の値の取得方法が設定される。具体的には、取得方法の項目には、該当の変数に代入する値を表す他の変数が設定される。開始および終了の項目に設定がある場合、取得方法の項目は設定なしとなる。

【0106】

例えば、疎行列情報220は、項目「インデックス番号」、変数「`index`」、開始「`row_ptr[r]`」、終了「`row_ptr[r+1]`」、取得方法「-」のレコードを有する。当該レコードは、目的のソースコードにおいて、インデックス番号を表す変数名に「`index`」を使用し、`index`の開始が「`row_ptr[r]`」、終了が「`row_ptr[r+1]`」であることを示す。

10

【0107】

また、疎行列情報220は、項目「行番号」、変数「`r`」、開始「0」、終了「NR」、取得方法「-」のレコードを有する。当該レコードは、目的のソースコードにおいて、疎行列の行番号を表す変数名に「`r`」を使用し、`r`の開始が「0」、終了が「NR」であることを示す。

【0108】

更に、疎行列情報220は、項目「列番号」、変数「`c`」、開始「-」、終了「-」、取得方法「`col_index[index]`」のレコードを有する。当該レコードは、目的のソースコードにおいて、疎行列の列番号を表す変数名に「`c`」を使用し、`c`の取得方法が「`col_index[index]`」であることを示す。

20

【0109】

例えば、コード生成部130は、疎行列情報220を基に、次の依存関係を得る。コード生成部130は、行番号を示す変数`r`が、疎行列情報220における変数`index`、`c`の何れにも依存せず、変数`r`に値が直接代入され、変数`r`に対してループの開始の値と終了の値とを指定可能であることを検出する。また、コード生成部130は、インデックス番号を示す変数`index`が、変数`r`に依存し、変数`r`に対応して値が間接的に代入され、変数`index`に対してループの開始の値と終了の値とを指定可能であることを検出する。更に、コード生成部130は、列番号を示す変数`c`が、疎行列情報220における変数`index`に依存し、変数`index`に対して値が間接的に代入され、変数`c`に対してループの開始の値と終了の値とを指定可能でないことを検出する。このように、疎行列情報220は、目的のソースコードにおける、疎行列を表す変数間の依存関係を示している。当該依存関係は、コード生成部130による下記の利用可能性判定や、後述される最適化プログラムコード候補生成に用いられる。

30

【0110】

ここで、例として、疎行列情報220を用いる場合の、最適化SCoP情報211, 212, 213に対する利用可能性判定を説明する。

まず、コード生成部130は、ステップS20で、疎行列情報220から変数の利用可能順序を決定する。疎行列情報220の変数は、`index`, `r`, `c`の3つである。変数`index`は、開始と終了で変数`r`を使用しているため、変数`r`に依存する。また、変数`c`は、取得方法で変数`index`を使用しているため、変数`index`に依存する。変数`r`は、他の変数に依存していない。したがって、コード生成部130は、変数の利用可能順序を(`r`, `index`, `c`)と決定する。なお、(`r`, `index`, `c`)は、左から右へ向かう順に利用可能であることを示す。

40

【0111】

次に、コード生成部130は、最適化SCoP情報211を処理対象の最適化SCoP情報Xとして選択する。なお、最適化SCoP情報Xの選択順序は任意でよい。最適化SCoP情報211では、疎行列の元となる密行列、すなわち、二次元配列Mを参照する文はs2である。したがって、コード生成部130は、ステップS22で特定される文s2を含むループ構造として(`r`, `c`)を検出する。ループ構造で示される変数の順序は、どちらも外側ループとして`r`が最初に、その内側ループとして`c`が次に来る。したがって、

50

コード生成部 130 は、ループ構造と疎行列情報 220 の変数の利用可能順序とが合っている、すなわち、一致していると判定し、ステップ S23 YES とする。

【0112】

次に、コード生成部 130 は、ステップ S24 で、文 s2 を含むループ内の文 s1 を検出する。コード生成部 130 は、ステップ S25 で、文 s1 で利用される変数 r の利用可能性について判定する。疎行列情報 220 では、変数 r は開始と終了がそれぞれ定義されている。したがって、コード生成部 130 は、文 s1 で利用される変数 r が利用可能であると判定する。そして、コード生成部 130 は、ステップ S26 で、疎行列情報 220 に対して最適化 SCOP 情報 211 を利用可能であると判定し、最適化 SCOP 情報 211 を集合 SET に追加する。

10

【0113】

また、コード生成部 130 は、最適化 SCOP 情報 212 に対して疎行列情報 220 を用いる場合については、文 s1 は文 s2 と異なるループ内に存在するため、ステップ S24 No となり、最適化 SCOP 情報 212 を利用可能と判定することになる。

【0114】

更に、コード生成部 130 は、最適化 SCOP 情報 213 に対して疎行列情報 220 を用いる場合については、最適化 SCOP 情報 213 を利用不可能と判定する。最適化 SCOP 情報 213 に対して、ステップ S22 で検出される文 s2 を含むループ構造は (c, r) となり、変数の利用可能順序 (r, index, c) と比較すると、変数 r, c の順序が合わないためである。

20

【0115】

上記の例では、CSR フォーマットの疎行列情報 220 を例示したが、CSC フォーマットや COO フォーマットなどでもよい。そこで、次に、CSC フォーマットおよび COO フォーマットを用いる場合を例示する。まず、CSC フォーマットを用いる場合を説明する。

【0116】

図 15 は、疎行列情報 (CSC) の例を示す図である。

疎行列情報 220 a は、疎行列の表現に CSC フォーマットを用いる場合のインデックス番号と、疎行列の行番号と列番号との関係を示す。当該インデックス番号は、列番号から疎行列の非ゼロの要素および当該要素の行番号の取得に用いられる変数である。疎行列情報 220 a は、疎行列情報 220 と同様の項目を有する。

30

【0117】

例えば、疎行列情報 220 a は、項目「インデックス番号」、変数「index」、開始「col\_ptr[c]」、終了「col\_ptr[c+1]」、取得方法「-」のレコードを有する。当該レコードは、目的のソースコードにおいて、インデックス番号を表す変数名に「index」を使用し、index の開始が「col\_ptr[c]」、終了が「col\_ptr[c+1]」であることを示す。

【0118】

また、疎行列情報 220 a は、項目「行番号」、変数「r」、開始「-」、終了「-」、取得方法「row\_index[index]」のレコードを有する。当該レコードは、目的のソースコードにおいて、疎行列の行番号を表す変数名に「r」を使用し、r の取得方法が「row\_index[index]」であることを示す。

40

【0119】

更に、疎行列情報 220 a は、項目「列番号」、変数「c」、開始「0」、終了「NC」、取得方法「-」のレコードを有する。当該レコードは、目的のソースコードにおいて、疎行列の列番号を表す変数名に「c」を使用し、c の開始が「0」、終了が「NC」であることを示す。

【0120】

例として、疎行列情報 220 a を用いる場合の、最適化 SCOP 情報 211, 212, 213 に対する利用可能性判定を説明する。

50

コード生成部 130 は、最適化 S C o P 情報 2 1 1 に対して、疎行列情報 2 2 0 a から決定される変数の利用可能順序 ( c , i n d e x , r ) と、最適化 S C o P 情報 2 1 1 のループ構造 ( r , c ) とが合わないため、利用不可能と判定する。

【 0 1 2 1 】

コード生成部 130 は、最適化 S C o P 情報 2 1 2 に対して、疎行列情報 2 2 0 a から決定される変数の利用可能順序 ( c , i n d e x , r ) と、最適化 S C o P 情報 2 1 2 のループ構造 ( r , c ) とが合わないため、利用不可能と判定する。

【 0 1 2 2 】

コード生成部 130 は、最適化 S C o P 情報 2 1 3 に対して、疎行列情報 2 2 0 a から決定される変数の利用可能順序 ( c , i n d e x , r ) と、最適化 S C o P 情報 2 1 3 のループ構造 ( c , r ) とが合っていると判定する。更に、最適化 S C o P 情報 2 1 3 において文 s 1 と文 s 2 とが異なるループに分割されている。したがって、コード生成部 130 は、疎行列情報 2 2 0 a に対して、最適化 S C o P 情報 2 1 3 を利用可能と判定する。

10

【 0 1 2 3 】

次に、C O O フォーマットを用いる場合を説明する。

図 1 6 は、疎行列情報 ( C O O ) の例を示す図である。

疎行列情報 2 2 0 b は、疎行列の表現に C O O フォーマットを用いる場合のインデックス番号と、疎行列の行番号と列番号との関係を示す。当該インデックス番号は、疎行列の非ゼロの要素の行番号および列番号の取得に用いられる変数である。疎行列情報 2 2 0 b は、疎行列情報 2 2 0 と同様の項目を有する。

20

【 0 1 2 4 】

例えば、疎行列情報 2 2 0 b は、項目「インデックス番号」、変数「 i n d e x 」、開始「 0 」、終了「 N N Z 」、取得方法「 - 」のレコードを有する。当該レコードは、目的のソースコードにおいて、インデックス番号を表す変数名に「 i n d e x 」を使用し、 i n d e x の開始が「 0 」、終了が「 N N Z 」であることを示す。

【 0 1 2 5 】

また、疎行列情報 2 2 0 b は、項目「行番号」、変数「 r 」、開始「 - 」、終了「 - 」、取得方法「 r o w [ i n d e x ] 」のレコードを有する。当該レコードは、目的のソースコードにおいて、疎行列の行番号を表す変数名に「 r 」を使用し、 r の取得方法が「 r o w [ i n d e x ] 」であることを示す。

30

【 0 1 2 6 】

更に、疎行列情報 2 2 0 b は、項目「列番号」、変数「 c 」、開始「 - 」、終了「 - 」、取得方法「 c o l u m n [ i n d e x ] 」のレコードを有する。当該レコードは、目的のソースコードにおいて、疎行列の列番号を表す変数名に「 c 」を使用し、 c の取得方法が「 c o l u m n [ i n d e x ] 」であることを示す。

【 0 1 2 7 】

例として、疎行列情報 2 2 0 b を用いる場合の、最適化 S C o P 情報 2 1 1 , 2 1 2 , 2 1 3 に対する利用可能性判定を説明する。疎行列情報 2 2 0 b の場合、変数の利用可能順序は ( i n d e x , ( r | c ) ) となる。当該利用可能順は、変数 i n d e x から変数 r , c の両方が同時に生成されることを示し、すなわち、変数 r , c によるループを構成しない特別ループの作成を意味する。したがって、コード生成部 130 は、最適化 S C o P 情報 2 1 1 , 2 1 2 , 2 1 3 の何れに対しても、ステップ S 2 3 N o と判定することはない。

40

【 0 1 2 8 】

しかし、最適化 S C o P 情報 2 1 1 では、文 s 1 用の変数 r のループを作成することができない。すなわち、疎行列情報 2 2 0 b の変数 r の開始 / 終了が設定されておらず、文 s 1 用のループ制御に変数 r を利用可能でない。よって、コード生成部 130 は、ステップ S 2 5 N o と判定し、最適化 S C o P 情報 2 1 1 を利用不可能と判定する。一方、最適化 S C o P 情報 2 1 2 , 2 1 3 については、コード生成部 130 は、利用可能と判定する。ただし、変数 r と変数 c のループに対して、変数 r と変数 c の順序に依存しない特別

50

ループを作成するため、最適化 S C o P 情報 2 1 2 , 2 1 3 の差はなくなり、コード変換により、どちらも結果として同じ最適化プログラムコード候補になる。このとき、変数 r のループは存在しないため、並列化は適用できなくなる。なお、疎行列情報 2 2 0 b を用いる場合に、最適化 S C o P 情報 2 1 2 , 2 1 3 に対して生成される最適化プログラムコード候補の例は、後述される。

#### 【 0 1 2 9 】

次に、最適化プログラムコード候補セット生成の手順を説明する。

図 1 7 は、最適化プログラムコード候補セット生成例を示すフローチャートである。

最適化プログラムコード候補セット生成は、ステップ S 1 3 に相当する。

#### 【 0 1 3 0 】

( S 3 0 ) コード生成部 1 3 0 は、図 1 3 の手順で疎行列情報 2 2 0 に対して利用可能と判定された最適化 S C o P 情報を 1 つ選択する。すなわち、コード生成部 1 3 0 は、集合 S E T の中から、処理対象とする最適化 S C o P 情報 Y を 1 つ選択する。

#### 【 0 1 3 1 】

( S 3 1 ) コード生成部 1 3 0 は、対象の最適化 S C o P 情報 Y の d o ループ構造を上から順に、外側から内側に向けて辿り順番にコードを参照する。参照するコードには、d o ループや文 s 1 , s 2 などの代入文が含まれる。コード生成部 1 3 0 は、コードを順番に辿る過程で、下記ステップ S 3 2 , S 3 3 を実行する。

#### 【 0 1 3 2 】

( S 3 2 ) コード生成部 1 3 0 は、d o ループに対して疎行列情報 2 2 0 に基づく f o r 文への変換およびデータ型情報 2 4 0 に基づく変数定義の生成を実行する。このとき、コード生成部 1 3 0 は、疎行列特殊化情報 2 5 0 に基づいて、データ特殊化を適用してもよい。

#### 【 0 1 3 3 】

( S 3 3 ) コード生成部 1 3 0 は、各代入文に対して右辺式情報 2 3 0 に従ったソースコードへの変換を実行する。このとき、コード生成部 1 3 0 は、疎行列の元になる密行列の変数（例えば、配列 M ）を、疎行列の変数（例えば、配列 S M ）に変換する。また、コード生成部 1 3 0 は、疎行列特殊化情報 2 5 0 に基づいて、データ特殊化を適用してもよい。コード生成部 1 3 0 は、疎行列情報 2 2 0 や疎行列特殊化情報 2 5 0 に含まれないデータについては、最適化 S C o P 情報 Y のデータを用いてそのままソースコードに変換する。

#### 【 0 1 3 4 】

( S 3 4 ) コード生成部 1 3 0 は、対象の最適化 S C o P 情報 Y の d o ループ構造を全て処理済であるか否かを判定する。最適化 S C o P 情報 Y の d o ループ構造を全て処理済の場合、コード生成部 1 3 0 は、生成した最適化プログラムコード候補を最適化プログラムコード候補セット 2 7 0 に追加して、ステップ S 3 5 に処理を進める。最適化 S C o P 情報 Y に未処理の d o ループ構造がある場合、コード生成部 1 3 0 は、ステップ S 3 1 に処理を進める。その後、コード生成部 1 3 0 は、ステップ S 3 1 で次の d o ループ構造を選択して手順を進める。

#### 【 0 1 3 5 】

( S 3 5 ) コード生成部 1 3 0 は、疎行列情報 2 2 0 に対して利用可能な最適化 S C o P 情報、すなわち、集合 S E T に含まれる最適化 S C o P 情報を全て処理済であるか否かを判定する。利用可能な最適化 S C o P 情報を全て処理済の場合、コード生成部 1 3 0 は、最適化プログラムコード候補セット生成を終了する。利用可能な最適化 S C o P 情報を全て処理済でない場合、コード生成部 1 3 0 は、ステップ S 3 0 に処理を進める。

#### 【 0 1 3 6 】

次に、コード生成部 1 3 0 による最適化プログラムコード候補セット生成の具体例を説明する。

図 1 8 は、右辺式情報の例を示す図である。

#### 【 0 1 3 7 】

10

20

30

40

50

右辺式情報 230 は、関数および式の項目を含む。関数の項目には、最適化 S C o P 情報に記述される関数名が登録される。関数の項目には、関数名とともに引数が登録されることもある。式の項目には、関数名に対応する式が登録される。式は定数を含む。

【0138】

例えば、右辺式情報 230 は、関数「( f 0 )」、式「0」のレコードを有する。当該レコードは、最適化 S C o P 情報に含まれる関数「( f 0 )」を、0 に変換することを示す。

【0139】

また、右辺式情報 230 は、関数「( f 1 @ 1 @ 2 @ 3 )」、式「@ 1 + @ 2 \* @ 3」のレコードを有する。当該レコードは、最適化 S C o P 情報に含まれる関数「( f 1 @ 1 @ 2 @ 3 )」を、「@ 1 + @ 2 \* @ 3」の式に変換することを示す。ここで、「@ 1」、「@ 2」、「@ 3」は、関数の引数を表す。例えば、最適化 S C o P 情報 211 では、10 行目に ( f 1 ( r v r ) ( M r c ) ( v c ) ) の記述がある。この場合、( r v r ) は、引数「@ 1」に対応する。( M r c ) は、引数「@ 2」に対応する。( v c ) は、引数「@ 3」に対応する。

【0140】

図 19 は、データ型情報の例を示す図である。

データ型情報 240 は、CSR フォーマット用であり、CSC フォーマットなど他のフォーマットを用いる場合には、該当のフォーマットに適合したデータ型情報が記憶部 110 に予め格納される。

【0141】

データ型情報 240 は、変数および型の項目を含む。変数の項目には、目的のソースコードで使用される変数が登録される。型の項目には、変数の型が登録される。例えば、データ型情報 240 は、変数「index」、型「int」のレコードを有する。当該レコードは、変数「index」を int 型とすることを示す。データ型情報 240 は、他の変数の型を示すレコードも保持する。

【0142】

図 20 は、最適化プログラムコード候補の第 1 の例を示す図である。

最適化プログラムコード候補 271 は、最適化プログラムコード候補セット 270 の要素である。最適化プログラムコード候補 271 は、疎行列情報 220 および最適化 S C o P 情報 211 に対して生成される。具体的には、コード生成部 130 は、最適化 S C o P 情報 211 を次のように最適化プログラムコード候補 271 に変換する。

【0143】

コード生成部 130 は、最適化 S C o P 情報 211 の 7 ~ 10 行目における下記の最初の d o ループ ( 1 ) を処理する。

```
( d o - p a r a l l e l ( r 0 ( - N R 1 ) )
...
) . . . ( 1 )
```

当該 d o ループは、並列化ループである。このため、コード生成部 130 は、疎行列情報 220 を利用して、下記の形式のループ ( 1 a ) に変換する。

【0144】

```
# p r a g m a o m p p a r a l l e l f o r
f o r ( i n t r = 0 ; r < N R ; r + + ) {
...
} . . . ( 1 a )
```

次に、コード生成部 130 は、最適化 S C o P 情報 211 の 8 行目における下記の代入文 ( 2 ) を処理する。

【0145】

```
( s 1 ( r v r ) = ( f 0 ) ) . . . ( 2 )
```

左辺はベクトルの参照である。右辺は右辺式情報 230 の値 0 の式である。よって、コ 50

ード生成部 130 は、当該代入文を下記のコード (2a) に変換する。

【0146】

```
rv[r] = 0;          . . . (2a)
```

次に、コード生成部 130 は、最適化 S C o P 情報 211 の 9 ~ 10 行目における下記の内側の d o ループ (3) を処理する。

【0147】

```
(do (c 0 (- NC 1))
```

...

)

. . . (3)

当該内側の d o ループは、変数 c のループである。しかし、疎行列情報 220 では、c の開始と終了は設定なしであり、直接ループにすることができない。したがって、コード生成部 130 は、上記の内側の d o ループに対して、c の代わりに i n d e x を用いるループを生成し、i n d e x から c を取り出す下記の形式のループ (3a) に変換する。

【0148】

```
int start = row_ptr[r];
```

```
int end = row_ptr[r+1];
```

```
for (int index = start; index < end; index++) {
```

```
    int c = col_index[index];
```

...

}

. . . (3a)

このとき、コード生成部 130 は変数の型についてはデータ型情報 240 を利用する。なお、コード生成部 130 は、最適化プログラムコード候補 271 に示すように、上記の変数 s t a r t や変数 e n d を用いない記述としてもよい。

【0149】

最後に、コード生成部 130 は、最適化 S C o P 情報 211 の 10 行目における、残る代入文 (4) を処理する。

```
(s2 (rv r) = (f1 (rv r) (M r c) (v c))) ... (4)
```

右辺は、右辺式情報 230 によればデータの乗算と加算である。また、コード生成部 130 は、M が疎行列の元の密行列を示すことを検出し、疎行列情報 220 を基に疎行列に対応する配列 S M に変換することで、下記のコード (4a) を得る。

【0150】

```
rv[r] = rv[r] + SM[index] * v[c]; ... (4a)
```

コード生成部 130 は、以上のコード変換の結果として、最適化プログラムコード候補 271 を生成する。なお、最適化プログラムコード候補 271 では、データ型情報 240 に含まれる一部の変数に対し、ループ外に記述される定義文の図示を省略している。以下の説明でも同様である。

【0151】

コード生成部 130 は、疎行列情報 220 に対して利用可能と判定された最適化 S C o P 情報 212 についても、同様のコード変換により、最適化プログラムコード候補を生成する。

【0152】

図 21 は、最適化プログラムコード候補の第 2 の例を示す図である。

最適化プログラムコード候補 272 は、疎行列情報 220 に対して利用可能と判定される最適化 S C o P 情報 212 に基づいて、コード生成部 130 によるコード変換により生成される。

【0153】

疎行列情報 220 に対して、最適化 S C o P 情報 213 は、利用不可能と判定されるため、コード生成部 130 は、最適化 S C o P 情報 213 に対するコード変換を行わない。したがって、この場合、最適化プログラムコード候補セット 270 の要素は、最適化プログラムコード候補 271, 272 の 2 つとなる。

10

20

30

40

50

## 【 0 1 5 4 】

そして、最適化プログラム選択部 1 4 0 は、最適化プログラムコード候補セット 2 7 0 の各要素の評価を行う。例えば、最適化プログラム選択部 1 4 0 は、最適化プログラムコード候補 2 7 1 , 2 7 2 それぞれをコンパイルして実行した結果、最適化プログラムコード候補 2 7 1 の方が良い性能であると評価する。すると、最適化プログラム選択部 1 4 0 は、最適化プログラムコードセット 2 9 0 に、最適化プログラムコード候補 2 7 1 を追加する。この場合、最適化プログラムコード候補 2 7 1 は、最終的に選択された最適化済のソースコードである。

## 【 0 1 5 5 】

なお、最適化プログラムコード候補 2 7 1 , 2 7 2 による処理性能が同じ程度であれば、他の疎行列データの場合には差が出る可能性がある。このため、最適化プログラム選択部 1 4 0 は、最適化プログラムコード候補 2 7 1 , 2 7 2 の両方を、最適化プログラムコードセット 2 9 0 に追加する。この場合、最適化プログラムコード候補 2 7 1 , 2 7 2 の両方は、最終的に選択された最適化済のソースコードである。そして、最適化プログラム選択部 1 4 0 は、最適化プログラムコードセット 2 9 0 を出力する。

## 【 0 1 5 6 】

なお、上記説明では、CSRフォーマットの疎行列情報 2 2 0 を使用する場合を主に説明したが、前述のように CSC フォーマットの疎行列情報 2 2 0 a や COO フォーマットの疎行列情報 2 2 0 b など、他のフォーマットの疎行列情報が用いられることもある。そこで、疎行列情報 2 2 0 a , 2 2 0 b が用いられる場合に生成される最適化プログラムコード候補を例示する。

## 【 0 1 5 7 】

図 2 2 は、最適化プログラムコード候補の第 3 の例を示す図である。

最適化プログラムコード候補 2 7 3 は、疎行列情報 2 2 0 a に対して利用可能と判定される最適化 SCOP 情報 2 1 3 に基づいて、コード生成部 1 3 0 によるコード変換により生成される。

## 【 0 1 5 8 】

なお、前述のように、最適化 SCOP 情報 2 1 1 , 2 1 2 , 2 1 3 のうち、疎行列情報 2 2 0 a に対して利用可能と判定されるものは最適化 SCOP 情報 2 1 3 のみである。この場合、最適化プログラム選択部 1 4 0 は、最適化 SCOP 情報 2 1 3 を基に生成した最適化プログラムコード候補 2 7 3 を、性能評価をスキップして、最適化プログラムコードセット 2 9 0 に追加してもよい。あるいは、最適化プログラム選択部 1 4 0 は、最適化プログラムコード候補 2 7 3 に対する性能評価を行い、当該評価結果が最低限満たすべき性能を満たしている場合に最適化プログラムコードセット 2 9 0 に追加してもよい。例えば、最適化プログラム選択部 1 4 0 は、最適化プログラムコードセット 2 9 0 が空集合となる場合、凸多面体最適化部 1 2 0 による最適化オプションの変更などをユーザに通知し、凸多面体最適化からやり直すように促してもよい。

## 【 0 1 5 9 】

図 2 3 は、最適化プログラムコード候補の第 4 の例を示す図である。

最適化プログラムコード候補 2 7 4 は、疎行列情報 2 2 0 b に対して利用可能と判定される最適化 SCOP 情報 2 1 3 に基づいて、コード生成部 1 3 0 によるコード変換により生成される。

## 【 0 1 6 0 】

ここで、最適化 SCOP 情報 2 1 3 は、9 ~ 1 1 行目の下記 d o ループ ( 5 ) を含む。

```
( d o ( c 0 ( - N C 1 ) )
    ( d o - p a r a l l e l ( r 0 ( - N R 1 ) )
...
) )
... ( 5 )
```

疎行列情報 2 2 0 b では、変数 c , r の何れも開始および終了が設定されていない。したがって、コード生成部 1 5 0 は、d o ループ ( 5 ) を下記の形式のループ ( 5 a ) に変

10

20

30

40

50

換する。

【0161】

```
for (int index = 0; index < NNZ; index++) {
    int r = row[index];
    int c = column[index];
    ...
}
```

・・・(5a)

コード生成部130は、以上の変換を含むコード変換の結果として、最適化プログラムコード候補274を生成する。

【0162】

ここで、ステップS32, S33で説明したように、コード生成部130は、最適化プログラムコード候補生成の過程で、疎行列特殊化情報250に基づくデータ特殊化を行ってもよい。そこで、次に、データ特殊化を説明する。

【0163】

図24は、疎行列特殊化情報の例を示す図である。

疎行列特殊化情報250は、項目、minおよびmaxの項目を含む。項目の項目には、特殊化対象のデータを示す識別名が登録される。minの項目には、該当データの最小値が登録される。maxの項目には、該当データの最大値が登録される。最小値と最大値とが同じ値の場合、該当のデータは定数であることを示す。

【0164】

例えば、疎行列特殊化情報250は、項目「1次元インデックス」、min「0」、max「10000」のレコードを有する。当該レコードは、1次元インデックス、すなわち、行番号を示すインデックスの範囲が0以上10000未満であることを示す。

【0165】

また、疎行列特殊化情報250は、項目「2次元インデックス」、min「0」、max「127」のレコードを有する。当該レコードは、2次元インデックス、すなわち、列番号に対応するインデックスの範囲が0以上127未満であることを示す。

【0166】

更に、疎行列特殊化情報250は、項目「データ値」、min「1.0」、max「1.0」のレコードを有する。当該レコードは、疎行列の要素のデータ値が全て1.0であることを示す。

【0167】

図25は、最適化プログラムコード候補の第6の例を示す図である。

最適化プログラムコード候補276は、最適化SCoP情報211、疎行列情報220、右辺式情報230、データ型情報240および疎行列特殊化情報250に基づいて、コード生成部130により生成される。コード生成部130は、最適化プログラムコード候補271の代わりに、最適化プログラムコード候補276を生成する。

【0168】

具体的には、コード生成部130は、疎行列特殊化情報250に基づいて、2次元インデックスが1バイトに収まる値の範囲であることを利用する。すなわち、コード生成部130は、最適化プログラムコード候補271の5行目の「int c = col\_index[index];」の代わりに、「char c = col\_index[index];」を生成する。

【0169】

また、コード生成部130は、疎行列特殊化情報250に基づいて、疎行列のデータの値が1.0だけであることを利用する。すなわち、コード生成部130は、最適化プログラムコード候補271の6行目の「rv[r] = rv[r] + SM[index] \* v[c];」の代わりに、「rv[r] = rv[r] + 1.0 \* v[c];」を生成する。

【0170】

その結果、コード生成部130は、最適化プログラムコード候補276を得る。コード

10

20

30

40

50



生成部 130 は、最適化プログラムコード候補 276 を生成することで、最適化プログラムコード候補 271 を用いるよりも、疎行列処理の高速化を図れる。

【0171】

図 26 は、最適化プログラムコード候補の第 7 の例を示す図である。

最適化プログラムコード候補 277 は、最適化 S C o P 情報 212、疎行列情報 220 a、右辺式情報 230、データ型情報および疎行列特殊化情報 250 に基づいて、コード生成部 130 により生成される最適化プログラムコード候補の例である。データ型情報としては、C S R フォーマット用のデータ型情報 240 の代わりに、C S C フォーマット用のデータ型情報が用いられる。

【0172】

また、コード生成部 130 は、最適化 S C o P 情報セット 210、疎行列情報 220、右辺式情報 230、データ型情報 240 よび疎行列特殊化情報 250 に加え、最適化戦略指示情報 260 に基づいて最適化プログラムコード候補セット 270 を生成してもよい。

【0173】

図 27 は、最適化戦略指示情報の例を示す図である。

最適化戦略指示情報 260 は、項目およびパラメータの項目を含む。項目の項目には、最適化の対象項目が登録される。パラメータの項目には、対象項目に対する最適化を行うか否かを示す情報が登録される。

【0174】

例えば、最適化戦略指示情報 260 は、項目「並列化」、パラメータ「O N」のレコードを有する。当該レコードは、並列化を利用することを示す。

また、最適化戦略指示情報 260 は、項目「ベクトル化」、パラメータ「O F F」のレコードを有する。当該レコードは、ベクトル化を利用しないことを示す。同様に、最適化戦略指示情報 260 は、ループ展開を利用しないことを示すレコードを有する。

【0175】

また、最適化戦略指示情報 260 は、項目「データ特殊化」、パラメータ「O N」のレコードを有する。当該レコードは、疎行列特殊化情報 250 に基づくデータ特殊化を利用することを示す。

【0176】

更に、最適化戦略指示情報 260 は、項目「アーキテクチャ」、パラメータ「x 8 6 \_ 6 4」のレコードを有する。当該レコードは、目的のソースコードに基づく処理を実行するコンピュータの命令セットアーキテクチャが「x 8 6 \_ 6 4」であり、当該命令セットアーキテクチャに応じた最適化を利用することを示す。

【0177】

このように、情報処理装置 100 は、最適化戦略指示情報 260 によるユーザの最適化戦略指示の入力を許容する。例えば、ユーザは、最適化戦略指示情報 260 において、データ特殊化を O N または O F F にすることでデータ特殊化を利用するか否かを情報処理装置 100 に指示できる。これにより、情報処理装置 100 は、ユーザの環境に合った最適化プログラムコードを、より効率的に生成可能になる。

【0178】

ここで、疎行列を用いて R A M 102 と C P U 101 の間のデータ転送量を減らすことでプログラム実行を高速化できる可能性がある。一方で、ソースコードは複雑になるためコンパイラによる最適化を適用することが難しくなる。また、疎行列において 0 がどのように分布しているかによってプログラムの実行時間が大幅に変わる可能性がある。特にキャッシュの効率的な利用が難しく、プログラムのチューニングが困難になる問題もある。

【0179】

疎行列処理のソースコードには、ポインタの使用やデータ間接参照が含まれるため、コンパイラで最適化することは難しい。そこで、ソースコードに記述され得る各種疎行列処理のアルゴリズムに対して、最適化済のライブラリを事前に用意しておき、当該ライブラリを利用することがある。しかし、事前に用意されたライブラリでは、実行時コンパイル

10

20

30

40

50

手法に対処できない。また、事前に用意されたライブラリが、プログラム記述者により利用される疎行列処理のデータ構造やアルゴリズムに合わない場合、当該ライブラリによる最適化を適用することはできない。また、疎行列処理の実行性能は、ターゲットアーキテクチャに大きく依存するので、新しい世代の計算機が出るたびに、事前に用意するライブラリの更新をしなければならず、任意の時点で最高の性能を得ることは難しい。

#### 【0180】

そこで、第2の実施の形態で例示したように、情報処理装置100は、最適化済のソースコードの集合である最適化プログラムコードセット290を自動生成する。特に、情報処理装置100は、ソースコードを直接、最適化コードに変換するのではなく、疎行列処理のアルゴリズムを記述したアルゴリズムSCOP情報200を最適化し、最適化SCOP情報セット210を得る。これにより、情報処理装置100は、疎行列処理を記述したソースコードには適用することができなかつた、凸多面体モデルを利用したループ最適化を利用できるようになる。情報処理装置100は、凸多面体最適化を行う既存のツールを活用することで、凸多面体最適化によるループ最適化を容易に利用できる。

10

#### 【0181】

また、情報処理装置100は、疎行列情報220、右辺式情報230およびデータ型情報240に基づいて、最適化SCOP情報セット210を所定のプログラミング言語で記述された最適化プログラムコード候補セット270に変換する。そして、情報処理装置100は、最適化プログラムコード候補を用いた場合の実際の疎行列に対する処理性能の評価に応じて、最適化プログラムコード候補セット270の中から最適化プログラムコードセット290を選択する。

20

#### 【0182】

これにより、情報処理装置100は、ユーザの環境に合った最適化プログラムコードを、効率的に得ることができる。例えば、アルゴリズムSCOP情報200において、データ型や右辺式の具体的な情報を除くことで、疎行列で利用するデータ型や代入文の右辺の形式が変わった場合でも、凸多面体モデルによるループの最適化効果を容易に得ることができる。更に、情報処理装置100は、最適化プログラムコードをコンパイルすることで得られた実行可能コードを実行して疎行列処理を行うことで、RAMなどのメモリとCPU間のデータ転送量を減らし、当該疎行列処理を高速に行える。

#### 【0183】

このように、情報処理装置100は、疎行列処理のアルゴリズム、疎行列フォーマット、データ型、疎行列の性質、ターゲットアーキテクチャ、最適化手法の組み合わせをパラメータとした幅広い最適化に対応できる。また、情報処理装置100は、疎行列アルゴリズムに対して、凸多面体モデルの最適化を適用することができる。更に、情報処理装置100は、ターゲットアーキテクチャに合わせた最適なソースコードを得ることができる。

30

#### 【0184】

情報処理装置100は、次の処理を実行するということもできる。

凸多面体最適化部120は、行列に対するループ処理が静的制御部形式で記述された第1コードを凸多面体モデルにより最適化することで複数の第2コードを取得する。コード生成部130は、疎行列の非ゼロの要素を表す変数を示す疎行列情報と、第2コードに含まれる関数に対応する演算式を示す式情報と、変数に対して使用する型を示すデータ型情報とに基づいて、複数の第2コードを複数のソースコード候補に変換する。最適化プログラム選択部140は、複数のソースコード候補それぞれを用いた場合の疎行列に対する処理性能の評価に応じて複数のソースコード候補の中からソースコードを選択する。

40

#### 【0185】

これにより、情報処理装置100は疎行列処理に対して最適化されたソースコードを効率的に得ることができる。ここで、アルゴリズムSCOP情報200は、第1コードの一例である。最適化SCOP情報セット210の各要素、すなわち、最適化SCOP情報は、第2コードの一例である。最適化プログラムコード候補セット270の各要素、すなわち、最適化プログラムコード候補は、ソースコード候補の一例である。最適化プログラム

50

コードセット 290 の各要素、すなわち、最適化プログラムコードは、複数のソースコード候補の中から選択されるソースコードの一例である。

【0186】

例えば、疎行列情報は、疎行列の表現形式に応じて目的のソースコードで使用する複数の変数であって、ループ処理を制御するインデックスを示す第1変数と疎行列の行番号を示す第2変数と疎行列の列番号を示す第3変数とを含む複数の変数それぞれの間の依存関係を示す情報を含む。コード生成部130は、複数の第2コードから複数のソースコード候補への変換では、疎行列情報に基づいて、複数の第2コードに含まれるループ処理の記述を、当該複数の変数を用いたコードに変換する。

【0187】

これにより、情報処理装置100は、CSRフォーマットやCSCフォーマットなどの疎行列の表現形式に適合したソースコード候補を効率的に生成できる。例えば、前述のように、コード生成部130は、疎行列情報に基づいて、目的のソースコード候補で利用される、疎行列を表す各変数について、当該変数に対するループの開始および終了の値の指定可否を含む変数間の依存関係を取得する。コード生成部130は、当該依存関係によりソースコード候補におけるループの記述を決定可能となり、当該記述を用いてソースコード候補を適切に生成することができる。

【0188】

また、コード生成部130は、複数の変数それぞれの間の依存関係と複数の第2コードそれぞれに含まれるループ構造とに基づいて、疎行列情報に対して複数の第2コードそれぞれを利用可能であるか否かを判定してもよい。そして、コード生成部130は、利用可能であると判定された第2コードをソースコード候補に変換してもよい。

【0189】

このように、情報処理装置100は、利用可能な第2コードを絞り込むことで、利用できないことが明らかな第2コードに対するソースコード候補の生成を省略でき、ソースコード候補の生成を効率化できる。すなわち、情報処理装置100は、余計な第2コードに対してソースコード候補生成以降の余計な処理が発生することを抑制できる。

【0190】

また、コード生成部130は、複数の変数それぞれの値域を示す情報に基づいて、複数のソースコード候補それぞれにおける変数に対する型の特殊化および疎行列の要素の値の特殊化の少なくとも何れかを行ってもよい。

【0191】

これにより、情報処理装置100は、最終的に得られるソースコードを用いた疎行列処理を高速化できる可能性を高められる。疎行列特殊化情報250は、複数の変数それぞれの値域を示す情報の一例である。

【0192】

また、第1コードおよび複数の第2コードでは、変数の型が省略されるとともに代入文における右辺の演算式が関数により省略されて記述される。更に、式情報は、関数に対応する当該演算式の情報を含む。コード生成部130は、複数の第2コードから複数のソースコード候補への変換では、当該式情報およびデータ型情報に基づいて、複数の第2コードにおける関数を演算式に変換する。

【0193】

このように、第1コードおよび第1コードを基に得られる第2コードにおいてデータ型や右辺式の具体的な情報が省かれることで、情報処理装置100によるソースコード生成の汎用性を高めることができる。すなわち、情報処理装置100は、疎行列で利用するデータ型や代入文の右辺の形式が変わった場合でも、式情報やデータ型情報を当該形式に合わせて用意することで、凸多面体モデルによるループの最適化効果を容易に得ることができる。

【0194】

また、複数の第2コードは、並列化対象のループ処理を示す記述を含む。コード生成部

10

20

30

40

50

130は、複数の第2コードから複数のソースコード候補への変換では、複数のソースコード候補における、並列化対象のループ処理に対応するループに対して、並列化指示文を挿入する。

【0195】

これにより、情報処理装置100は、凸多面体最適化により並列化可能と判定されたループ箇所を、コンパイラに対して適切に指示できる。

更に、最適化プログラム選択部140は、複数のソースコード候補のうち、処理性能を示す指標が基準値よりも良いソースコード候補を、最終的なソースコードとして選択し、選択したソースコードを出力する。

【0196】

これにより、情報処理装置100は、実際に処理対象となる疎行列に対して、処理性能の向上が見込める可能性の高いソースコード候補に絞り込める。処理性能の指標としては、例えば、疎行列に対する処理の実行時間が挙げられる。例えば、最適化プログラム選択部140は、実行時間が基準値(閾値)よりも短いソースコード候補を、最終的に出力するソースコードとして選択してもよい。

【0197】

なお、第1の実施の形態の情報処理は、処理部12にプログラムを実行させることで実現できる。また、第2の実施の形態の情報処理は、CPU101にプログラムを実行させることで実現できる。プログラムは、コンピュータ読み取り可能な記録媒体73に記録できる。

【0198】

例えば、プログラムを記録した記録媒体73を配布することで、プログラムを流通させることができる。また、プログラムを他のコンピュータに格納しておき、ネットワーク経由でプログラムを配布してもよい。コンピュータは、例えば、記録媒体73に記録されたプログラムまたは他のコンピュータから受信したプログラムを、RAM102やHDD103などの記憶装置に格納し(インストールし)、当該記憶装置からプログラムを読み込んで実行してもよい。

【符号の説明】

【0199】

- 10 情報処理装置
- 11 記憶部
- 12 処理部
- 20 SCOPコード
- 30, 31, 32 最適化SCOPコード
- 40 疎行列情報
- 41 式情報
- 42 データ型情報
- 50, 51, 52 ソースコード候補
- 60 ソースコード
- S1, S2, S3 ステップ

10

20

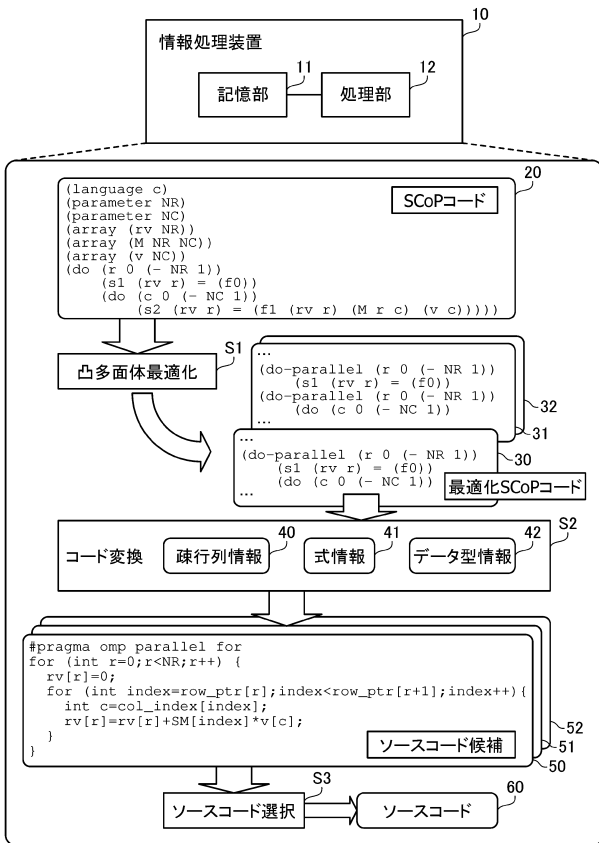
30

40

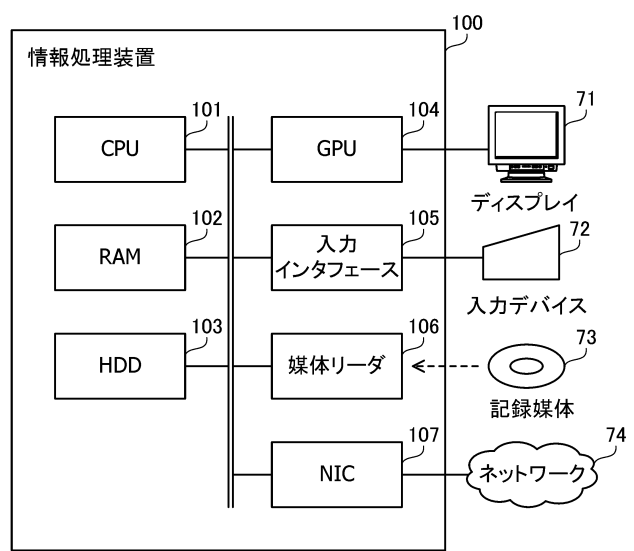
50

【 図 面 】

【 図 1 】



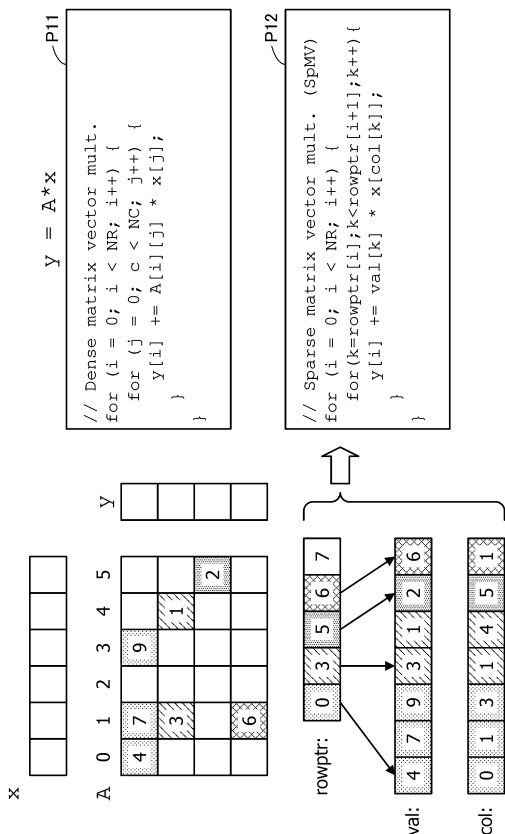
【 図 2 】



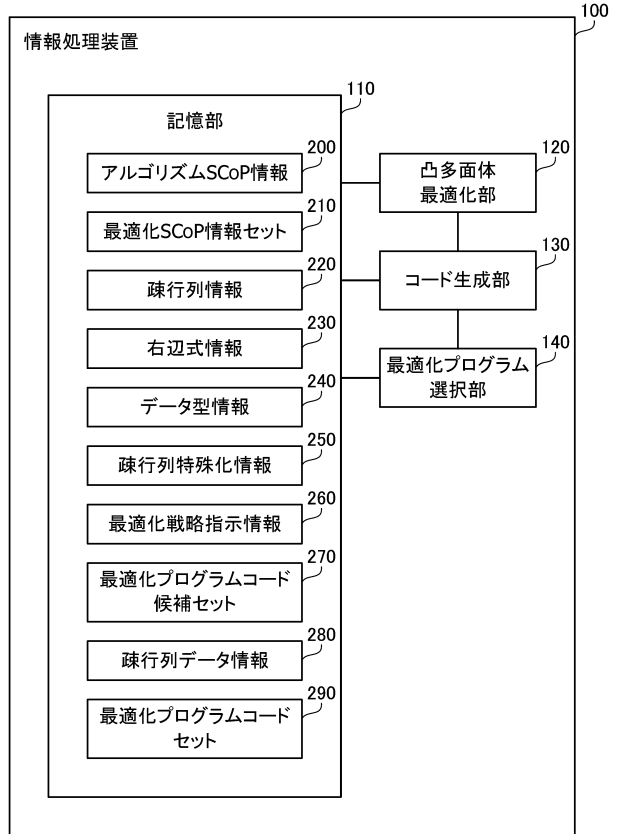
10

20

【 図 3 】



【 図 4 】

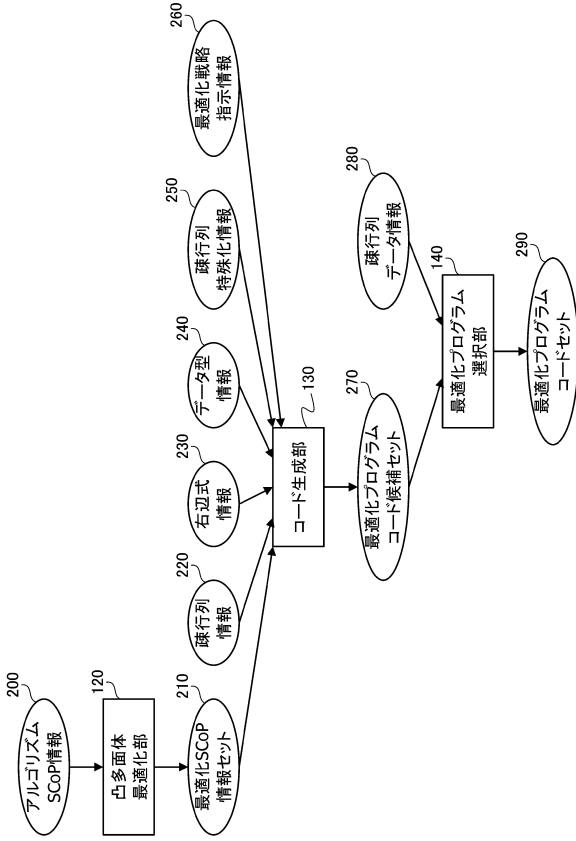


30

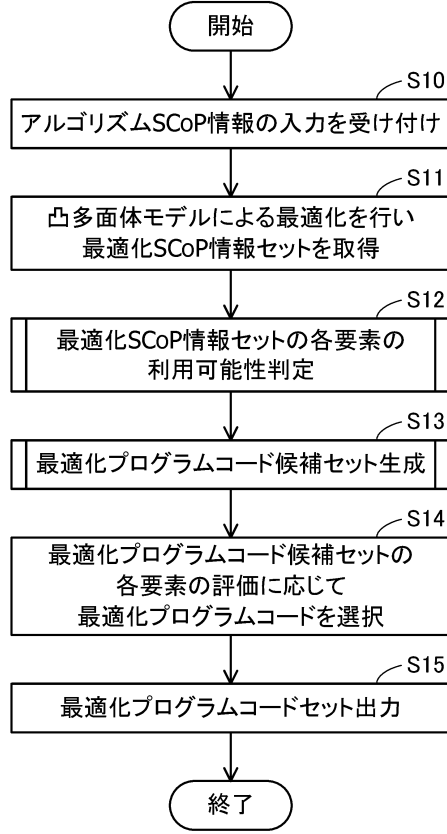
40

50

【 図 5 】



【 図 6 】



10

20

【 図 7 】

行列ベクトル積プログラムの例

```

1 for (int r = 0; r < NR; r++) {
2   rv[r] = 0;
3   for (int c = 0; c < NC; c++) {
4     rv[r] = rv[r] + M[r][c] * v[c];
5   }
6 }

```

【 図 8 】

疎行列ベクトル積プログラムの例 (CSRフォーマット)

```

1 for (int r = 0; r < NR; r++) {
2   rv[r] = 0;
3   int start = row_ptr[r];
4   int end = row_ptr[r + 1];
5   for (int index = start; index < end; index++) {
6     rv[r] = rv[r] + SM[index] * v[col_index[index]];
7   }
8 }

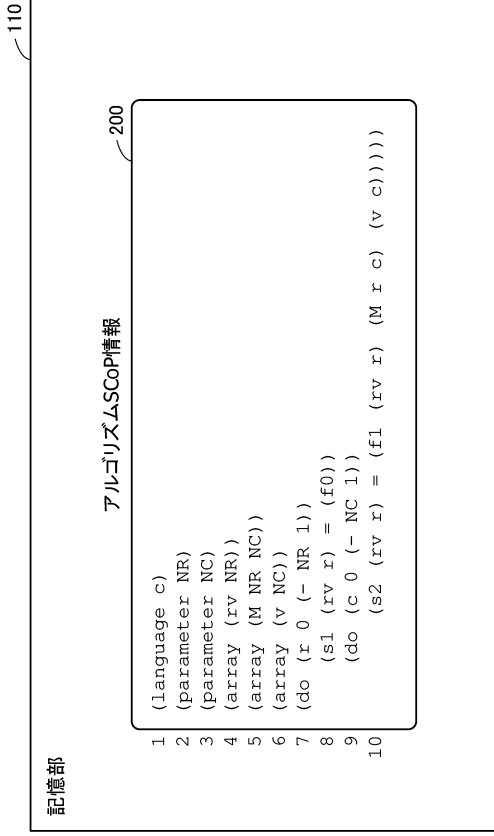
```

30

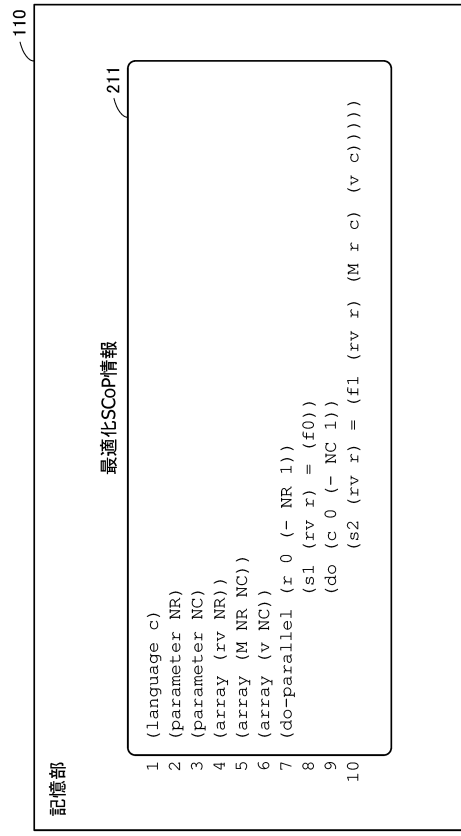
40

50

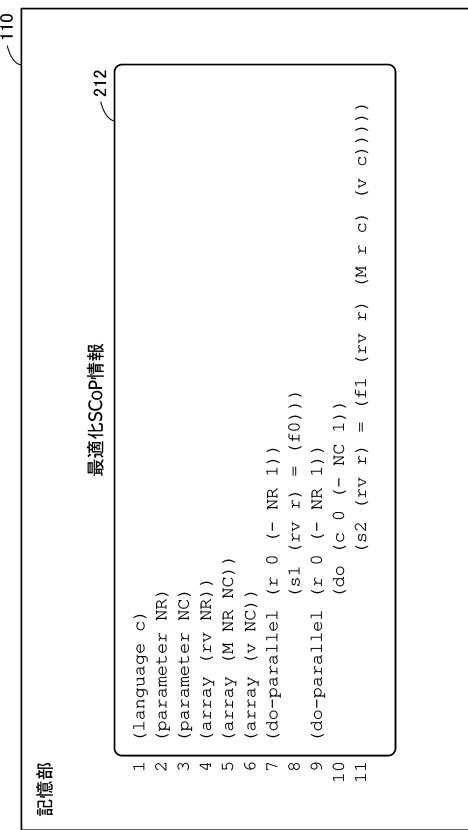
【 図 9 】



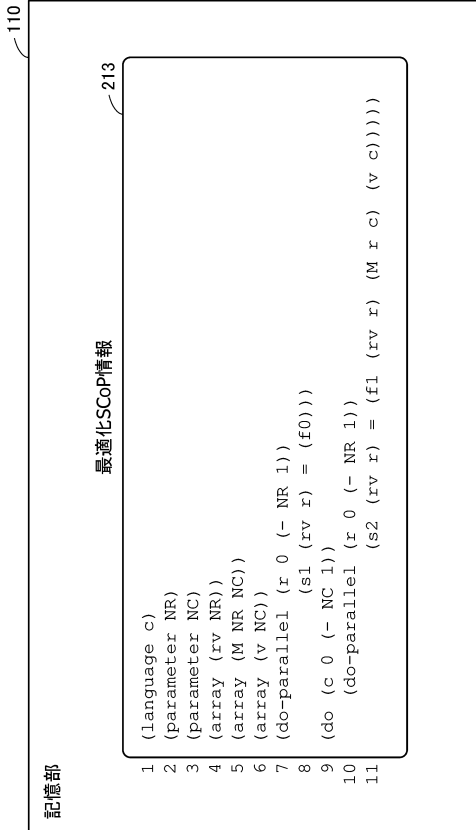
【 図 10 】



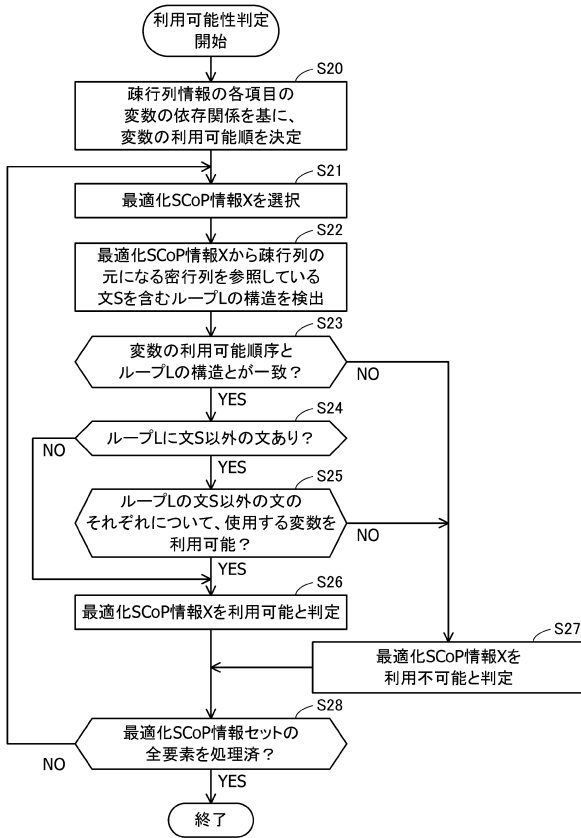
【 図 11 】



【 図 12 】



【 図 1 3 】



【 図 1 4 】

記憶部 110

疎行列情報 (CSR) 220

項目	変数	開始	終了	取得方法
インデックス番号	index	row_ptr[r]	row_ptr[r+1]	-
行番号	r	0	NR	-
列番号	c	-	-	col_index[index]

10

20

【 図 1 5 】

記憶部 111

疎行列情報 (CSC) 220a

項目	変数	開始	終了	取得方法
インデックス番号	index	col_ptr[c]	col_ptr[c+1]	-
行番号	r	-	-	row_index[index]
列番号	c	0	NC	-

【 図 1 6 】

記憶部 110

疎行列情報 (COO) 220b

項目	変数	開始	終了	取得方法
インデックス番号	index	0	NNZ	-
行番号	r	-	-	row[index]
列番号	c	-	-	column[index]

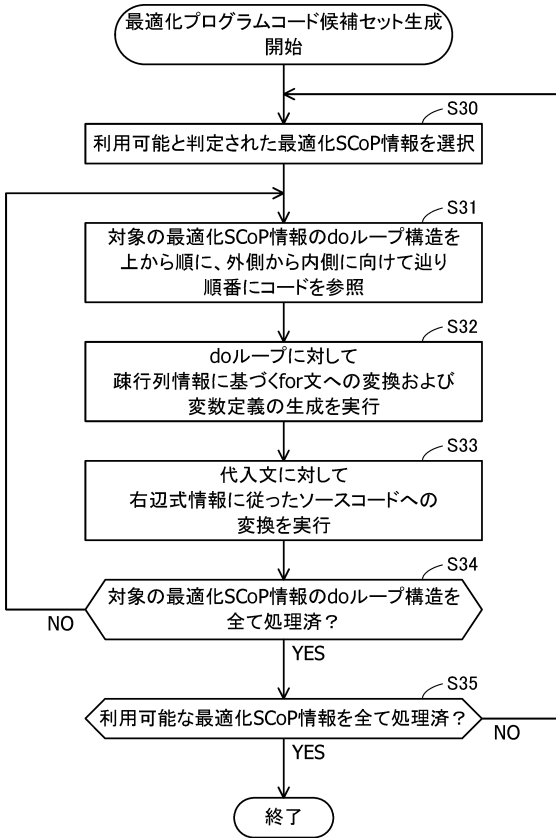
30

40

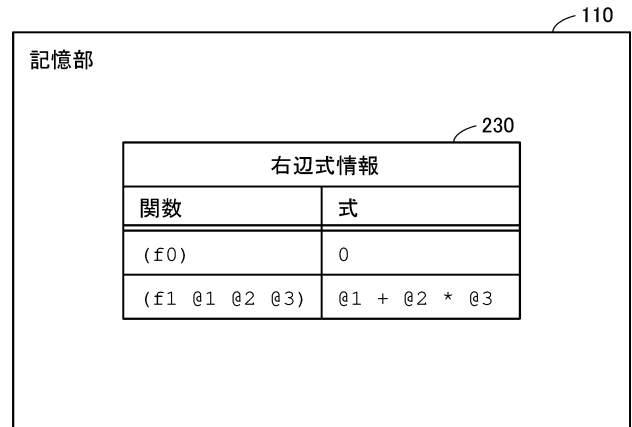
50



【図 17】



【図 18】



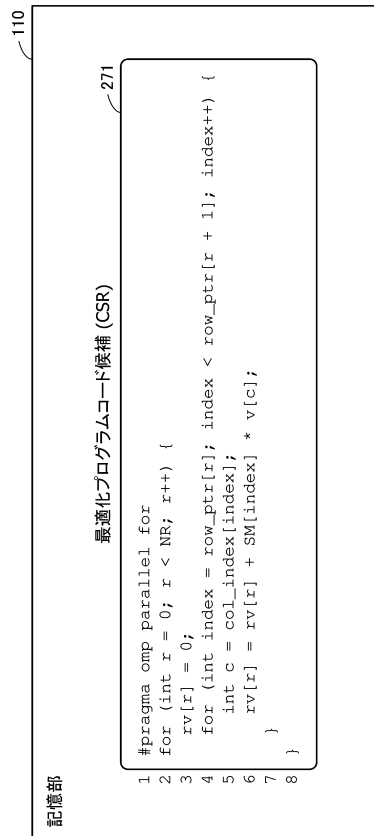
10

20

【図 19】



【図 20】



30

40

50

【 図 2 1 】

```

110
最適化プログラムコード候補 (CSR)
272
1 #pragma omp parallel for
2 for (int r = 0; r < NR; r++) {
3   rv[r] = 0;
4 }
5 #pragma omp parallel for
6 for (int r = 0; r < NR; r++) {
7   for (int index = row_ptr[r]; index < row_ptr[r + 1]; index++) {
8     int c = col_index[index];
9     rv[r] = rv[r] + SM[index] * v[c];
10  }
11 }

```

【 図 2 2 】

```

110
最適化プログラムコード候補 (CSC)
273
1 #pragma omp parallel for
2 for (int r = 0; r < NR; r++) {
3   rv[r] = 0;
4 }
5 for (int c = 0; c < NC; c++) {
6   #pragma omp parallel for
7   for (int index = col_ptr[c]; index < col_ptr[c + 1]; index++) {
8     int r = row_index[index];
9     rv[r] = rv[r] + SM[index] * v[c];
10  }
11 }

```

【 図 2 3 】

```

110
最適化プログラムコード候補 (COO)
274
1 #pragma omp parallel for
2 for (int r = 0; r < NR; r++) {
3   rv[r] = 0;
4 }
5 for (int index = 0; index < NNZ; index++) {
6   int r = row[index];
7   int c = column[index];
8   rv[r] = rv[r] + SM[index] * v[c];
9 }

```

【 図 2 4 】

110

記憶部

250

疎行列特殊化情報		
項目	min	max
1次元インデックス	0	10000
2次元インデックス	0	127
データ値	1.0	1.0

10

20

30

40

50

【 図 2 5 】

110

最適化プログラムコード候補 (CSR)

276

```

1 #pragma omp parallel for
2 for (int r = 0; r < NR; r++) {
3   rv[r] = 0;
4   for (int index = row_ptr[r]; index < row_ptr[r + 1]; index++) {
5     char c = col_index[index];
6     rv[r] = rv[r] + 1.0 * v[c];
7   }
8 }

```

記憶部

【 図 2 6 】

110

最適化プログラムコード候補 (CSC)

277

```

1 #pragma omp parallel for
2 for (int r = 0; r < NR; r++) {
3   rv[r] = 0;
4 }
5 for (char c = 0; c < NC; c++) {
6   #pragma omp parallel for
7   for (int index = col_ptr[c]; index < col_ptr[c + 1]; index++) {
8     int r = row_index[index];
9     rv[r] = rv[r] + 1.0 * v[c];
10  }
11 }

```

記憶部

10

20

【 図 2 7 】

110

記憶部

260

最適化戦略指示情報	
項目	パラメータ
並列化	ON
ベクトル化	OFF
ループ展開	OFF
データ特殊化	ON
アーキテクチャ	x86_64

30

40

50