(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7177768号
(P7177768)

(45)発行日　令和4年11月24日(2022.11.24)

(24)登録日　令和4年11月15日(2022.11.15)

(51)国際特許分類

Ｇ０１Ｃ　21/20　(2006.01)		Ｇ０１Ｃ	21/20	
Ｇ０９Ｂ　29/00　(2006.01)		Ｇ０９Ｂ	29/00	Ｚ

請求項の数　7　（全25頁）

(21)出願番号　特願2019-512974(P2019-512974)
(86)(22)出願日　平成29年9月15日(2017.9.15)
(65)公表番号　特表2019-529898(P2019-529898A)
(43)公表日　令和1年10月17日(2019.10.17)
(86)国際出願番号　PCT/US2017/051800
(87)国際公開番号　WO2018/053277
(87)国際公開日　平成30年3月22日(2018.3.22)
審査請求日　令和1年10月23日(2019.10.23)
(31)優先権主張番号　15/267,942
(32)優先日　平成28年9月16日(2016.9.16)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　505035585
ポラリス インダストリーズ インコーポレーテッド
アメリカ合衆国 ミネソタ州 ５５３４０
メディナ ハイウェー ５５ ２１００
(74)代理人　100073184
弁理士　柳田 征史
(74)代理人　100123652
弁理士　坂野 博行
(74)代理人　100175042
弁理士　高橋 秀明
(72)発明者　フリード，エリック エス
アメリカ合衆国 ミネソタ州 ５５１１２
ニュー ブライトン セブンティーンス アヴェニュー ノースウエスト １５４６

最終頁に続く

(54)【発明の名称】　ルート計画計算デバイスを改良するためのデバイス及び方法

(57)【特許請求の範囲】

【請求項１】

ルート計画計算デバイスのルート生成器の情報処理方法であって、

複数のブロックを有し、該複数のブロックの各々がルートトラバーサル値を有し、物理的地形の異なるセクションに対応するトラバーサルマップを生成するステップと、

ブロック群の各ブロックが、前記ブロック群の他の前記ブロックと如何なるエッジも共有しないように前記複数のブロックから１つのブロック群を選択するステップと、

前記ブロック群のルートトラバーサル値を同時に計算し、それによって前記物理的地形をトラバースするルートを生成するステップであって、前記ブロック群の各ブロックは複数のセルを含み、前記ルートトラバーサル値は、（ｉ）地形移動コストデータと、(ii)ルート終点への進行を示すデータとを、セル毎に検討した値である、ステップと

を含む方法。

【請求項２】

前記ブロック群に関する前記ルートトラバーサル値は、グラフィック処理ユニットによって同時に計算される、請求項１に記載の方法。

【請求項３】

前記複数のブロックの各ブロックは、２つのパリティの値のうちの一方が割り当てられる、請求項１又は２に記載の方法。

【請求項４】

前記複数のブロックの各々は四角形を有し、第１のパリティを有する、前記複数のブロ

ックの各四角形のブロックは、第２のパリティのみを有する、前記複数のブロックの他の四角形のブロックに対して側方に隣接する、請求項３に記載の方法。
【請求項５】
前記複数のブロックの各々は四角形であり、第１のパリティを有する、前記複数のブロックの各四角形のブロックは、第１のパリティのみを有する前記複数のブロックの他の四角形のブロックに対して対角線方向に隣接する、請求項３又は４に記載の方法。
【請求項６】
前記ブロック群の各ブロックは、同じパリティを有する、請求項３～５のいずれか１項に記載の方法。
【請求項７】
いずれの前記ブロック内のいずれの前記セルに関する前記トラバーサル値を計算する際、前記ブロック内の全ての前記セルに関する前記トラバーサル値を計算する必要がある、請求項１～６のいずれか１項に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本開示は一般に、ルート計画デバイス及び上記デバイスを操作する方法に関する。より具体的には、本開示は、ルート計画デバイスの速度及び効率を向上させる、上記ルート計画デバイスの計算動作の改善に関する。
【背景技術】
【０００２】
全地球測位システム（ＧＰＳ）を採用したもの等のルート計画デバイスは、ある場所から別の場所へのルートを計画するために使用できる。Ａ＊（「Ａスター（Ａ　ｓｔａｒ）」）は、ノードと呼ばれる複数の点間の効率的に走行可能な経路をプロットするために、経路探索及びグラフトラバーサルに幅広く使用されているコンピュータアルゴリズムである。Ａ＊は情報探索アルゴリズム又は最良優先探索であり、つまり、ソリューション（目標）への全ての可能な経路の中から、発生するコストが最小である（移動距離が最小である、時間が最短である等）ものを探索することによって、問題を解決し、またこれらの経路の中から、ソリューションに最も迅速にたどり着くと思われるものを最初に検討する。これは重み付けグラフで定式化される：グラフの特定のノードから開始して、該ノードから始まる複数の経路のツリーを構成し、その経路のうちの１つが所定の目標ノードにおいて終端するまで、一度に１ステップずつ上記経路を拡張する。
【０００３】
ルート計画が道路に限定されている場合、限定された移動の選択肢（限定されたノードのセット）により、Ａ＊は、経路が長い経路であっても、ユーザが許容可能な期間内に経路を計算できる。しかしながら、Ａ＊によるルート計画は、時間がかかるものとなり得、特にオフロードでの移動に関しては、リアルタイムルート設定には遅すぎる場合がある。移動のための可能な領域（ノード）が道路に限定されていない場合（徒歩、又はレクリエーション用車両等によるオフロード移動等）、増加したノードがＡ＊に殺到して、出力される経路への到達までに必要な時間を大幅に増加させる場合がある。
【０００４】
また更に、ルート設定デバイスのプロセッサによるＡ＊の命令の解釈により、ルート設定デバイス内で特定の動作が引き起こされる。Ａ＊の動作は、あるノードに関して行われる計算が他のノードに影響を及ぼす可能性を有することを禁じている。従ってノードは連続的に、一度に１つの方式で処理されるのが普通であり、複数の同時処理ソース／コアによる計算を検討する能力を有しない。これに関連して、Ａ＊の命令は、ＧＰＵで採用されているような超並列スレッド化を可能とするようには構成されていない。
【発明の概要】
【発明が解決しようとする課題】
【０００５】

従って必要とされているのは、ルート設定デバイスの動作を改善するための、オフロード移動の多ノード環境で迅速かつ効率的に動作できる、デバイス及び方法である。

【課題を解決するための手段】

【０００６】

本開示は、ルート生成器を操作する第１の実施形態の方法を含み、上記方法は、第１のグループ内の複数のブロックに関して、ルートトラバーサル値を同時に計算するステップであって、各上記ブロックは複数のセルを含み、上記トラバーサル値は、地形移動コストデータと、ルート終点への進行を示すデータとを、セル毎に検討した値である、ステップを含み、上記複数のブロックは、上記第１のグループ内の上記ブロックが、上記第１のグループ内の他のグループと、いずれのエッジを共有できないように選択される。

【０００７】

本開示はまた、ルート生成器を操作する第２の実施形態の方法を含み、上記方法は：コストマップのセルの第１のブロックに関するデータを、グラフィック処理ユニットのキャッシュメモリにロードするステップであって、上記第１のブロック内の１つの上記セルに関するデータのロードは、上記第１のブロック内の全ての上記セルに関するデータのロードを必要とする、ステップ；及びセルの第２のセットをロードするステップであって、上記第２のセット内の各上記セルは、上記第１のブロック内にはなく、上記第１のブロック内のあるセルとエッジを共有するか、又は対角線方向に隣接する、ステップを含み、上記第１のブロック内の上記セルの数と、上記第２のセット内の上記セルの数との比は、１：８未満であり、上記セルの第１のブロックと上記セルの第２のセットとの組み合わせは、上記第１のブロックの全ての上記セルに関するトラバーサル値を生成するために必要な全てのマップデータを提供し、上記トラバーサル値は、コストデータと、ルート終点への進行を示すデータとを検討した値である。

【０００８】

更に別の実施形態では、本開示は、ルート生成器を操作する方法を含み、上記方法は：ルートの決定の一部として第１のセルが処理されることになることを決定するステップ；上記第１のセルが、少なくとも２つのセルを内包する第１のブロック内にあることを決定するステップ；上記第１のブロック内の全ての上記セルを分析するために必要なデータを、第１の不揮発性メモリから、プロセッサによってアクセス可能な揮発性メモリにロードするステップであって、上記第１のブロック内の全ての上記セルを分析するために必要な上記データは、上記第１のブロック内のある１つの上記セルとエッジ及びコーナーのうちの少なくとも１つを共有する全てのセルを含む、ステップ；及び上記第１のブロック内の全ての上記セルを分析するために必要な上記データを全て上記揮発性メモリにロードした後、上記第１のブロック内の全ての上記セルを分析するステップを含む。

【０００９】

これより、本発明の実施形態を図面に関連して説明する。

【図面の簡単な説明】

【００１０】

【図１】例示的なルート計画システムの概略図

【図２】図１のシステムが検討する例示的な土地被覆マップ

【図３】図２の土地被覆マップを用いて生成された、例示的なコストマップ

【図４】図２及び３のマップに適用されるパリティブロックの例示的なセット

【図５】図１のシステムの動作を示すフローチャート

【図６】図５の動作のルートフラッディングの動作を示すフローチャート

【図７】図６のノードの処理を示すフローチャート

【図８】例示的な論理トラバーサルブロック構成及びそれに対する影響の領域

【図９Ａ】図７のブロックスウィーパーの動作を示すフローチャート

【図９Ｂ】図９Ａの続き

【図１０】図７の終点セルシーダーの動作を示すフローチャート

【図１１】図９Ａ～Ｂのブロックのスウィープに関する更なる詳細を示すフローチャート

【図１２】図７の隣接するノードの追加の後に採用されるブロックリデューサーの動作を示すフローチャート
【図１３】図６のブロックプライオリタイザーの動作を示すフローチャート
【図１４】図５のルートのベクトル化の動作を示すフローチャート
【図１５】ルートが重ねられたトラバーサルマップの視覚的描写を示す例示的な図
【図１６】セルに関する代替的な隣接の検討を示す図
【発明を実施するための形態】
【００１１】
複数の図にわたり、対応する参照符号は対応する部品を示す。本明細書に記載される適例は、本発明の実施形態を例示し、このような適例は、本発明の範囲をいかなる様式でも限定するものと解釈されないものとする。
【００１２】
本明細書に開示される実施形態は、包括的であること、又は以下の「発明を実施するための形態」において開示される正確な形態に本開示を限定することを意図したものではない。寧ろこれらの実施形態は、当業者がこれらの教示を利用できるように、選択され、説明されている。
【００１３】
本明細書中で使用される場合、用語「論理（ｌｏｇｉｃ）」又は「制御論理（ｃｏｎｔｒｏｌ　ｌｏｇｉｃ）」は、１つ以上のプログラマブルプロセッサ、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、デジタル信号プロセッサ（ＤＳＰ）、ハードワイヤードロジック、又はこれらの組み合わせにおいて実行される、ソフトウェア及び／又はファームウェアを含んでよい。従って実施形態によると、様々な論理を、いずれの適切な様式で実装してよく、またこれは本明細書で開示される実施形態に従ったものであり続ける。
【００１４】
図１は、ルート計画計算システム１０を示す。システム１０は、１つ以上のプロセッサ１２、１４、キャッシュメモリ１６ａ、ｂ、ＲＡＭ１８、及びハードドライブ２０を含む。プロセッサ１２は中央演算処理装置（ＣＰＵ）１２であり、プロセッサ１４はグラフィック処理ユニット１４である。各プロセッサ１２、１４はその上にキャッシュメモリ１６ａ、ｂを有する。ＲＡＭ１８は、プロセッサ１２及びプロセッサ１４に共有されているものとして図示されている。しかしながら、各プロセッサ１２、１４が別個のＲＡＭ１８を有する実施形態が想定される。ビデオＲＡＭ２２は、ＧＰＵ１４による使用のためだけに及び／又はＧＰＵ１４による使用のために優先的に設けられたＲＡＭである。ハードドライブ２０は従来のマスストレージデバイスである。
【００１５】
コンピュータ分野では容易に理解されるように、メモリは速いほど高価になるのが一般的である。従って計算システムは、長期格納のための大型で比較的遅い不揮発性メモリを有するように設計されてきた。より小型で速いメモリの１つ以上の層も、プロセッサが現在使用している又は使用する可能性があるデータの保持のために設けられる。従ってＲＡＭ１８及びビデオＲＡＭ２２は、ハードドライブ２０のメモリよりアクセス時間が短い。ＲＡＭ１８及びビデオＲＡＭ２２のサイズ（データ容量）は、ハードドライブ２０より小さい。同様に、キャッシュ１６ａ、ｂはＲＡＭ１８及びビデオＲＡＭ２２よりアクセス時間が短い。キャッシュ１６ａ、ｂのサイズ（データ容量）は、ＲＡＭ１８及びビデオＲＡＭ２２より小さい。このようなサイズは必須ではないことを理解されたい。しかしながら、プライスポイントとのトレードオフで計算速度を提供することに関心が持たれているため、上述のような相対的なサイズ設定が一般的となっている。
【００１６】
従ってプロセッサ（１２又は１４）がデータに対して動作するために、上記データを主にハードドライブ２０に格納する。プロセッサ１２、１４がデータを要求すると、メモリ１６、１８、２０、２２は、典型的なキャッシングの形態で動作し、最も近く速いメモリ

（キャッシュ１６ａ、ｂ）をまずチェックする。所望のデータがメモリ（１６、１８、２０、２２）のあるレベルに存在しない場合、次のレベルをチェックする。所望のデータが見つかると、該データがキャッシュ１６ａ、ｂ内に存在するようになり、プロセッサ１２、１４によって直接アクセス可能となるまで、各レベルを通して該データを引き上げる。

【００１７】

キャッシュ１６ａ、ｂのサイズが小さいことにより、その中に保持されるデータは多くの場合、プロセッサ１２、１４が現在実行中の動作のために即座に必要とするデータに限定される。メモリ間のデータの移動は、このような移動に時間がかかり、かつメモリの切り替えにビットを必要とするという点で、処理及び電力消費のオーバーヘッドを招く。

【００１８】

ＧＰＵ１４は、巨大なバッチのデータを取得し、同一の動作を何度も極めて迅速に実行するために最適化されたプロセッサである。ＣＰＵマイクロプロセッサは、リクエストを処理する場所全体をスキップする傾向がある。ＣＰＵ１２は数個のコアで構成され、一度に数個のソフトウェアスレッドを処理できる。対照的に、ＧＰＵ１４は、同時に数千個のスレッドを処理できる数百個のコアで構成される。

【００１９】

図２は、マップデータベース７０に格納された土地被覆マップ２００の一部分を示す。ある例では、タイルデータが、ＵＳＧＳデータベースから取得され、及び／又はＥＳＲＩ，　Ｉｎｃ．（カリフォルニア州レッドランズ）のＭａｐＩｎｆｏ等の標準的な地理情報システム（ＧＩＳ）を介して取得される。土地被覆マップ２００は複数のタイルに分割される（図１５を参照）。各タイルは複数のブロック２１０（例えば６４個のブロック）を内包する。図２は１つのこのようなブロック２１０を示す。各ブロック２１０は例えば、セル２２０の８×８のグリッドを含み、ブロックあたり合計６４個のセルを含む。各セル２２０は、単一の土地タイプを内包する。従ってルート設定を目的として、セル２２０内の土地タイプを同種と見なす。マップデータベース７０は更に、標高マップ（図示せず）を含む。

【００２０】

土地被覆マップ２００及び標高マップをプロセッサ１２、１４で処理して、コストマップ３００（図３）を生成する。この処理の更に具体的な詳細は以下で提供される。ここで提供されている例では、コストマップ３００は、現在の輸送モード（歩行、車輪での搬送等）に関して、マップの各セルのトラバーサルにどの程度のコストが掛かるか、又はトラバーサルがどの程度遅いかを表す、コスト決定器９６によって生成される。コスト値５＝速い、１５＝遅い、及び∞＝通行不可能を、土地被覆マップから決定されたセルの特徴に基づいて割り当てる。コストマップ３００は例えば、キャッシュ１６ｂ、ビデオＲＡＭ２２、及びＲＡＭ１８のうちの１つ以上に保持される。ルート設定ルーチンによって考察される各輸送モードに対して異なるコストマップが生成されることを理解されたい。更に、コスト決定器９６はＧＰＵ１４内に位置するものとして図示されているが、より一般的には、コスト決定器９６及びルート生成器８０がＣＰＵ１２内でインスタンス化される実施形態が想定される。

【００２１】

オフロードルートを生成する際、ルートは可能性として、１個のセルから隣接する８個のセル（側面が隣接する４個のセル、及びコーナーが隣接する４個のセル）へと進むことができる。いずれのセルＡから隣接するセルＢに移動する場合、セルＡとセルＢとの間の平均コストにセルＡ及びセルＢの中心間の距離を掛けたものに等しいコストが累積される。

【００２２】

従ってルートの計算時、検討されるセルに関するいずれのコストを、プロセッサ１２、１４によってアクセス可能となるように、キャッシュ１６ａ、ｂにロードする。よって特定の実施形態では、１回に１個のセルを検討する場合、関心対象のセル及びその周囲の８個のセルをキャッシュ１６ａ、ｂ内に移動させる。他のセルの検討によってこれらのセルのうちのいくつかが既にキャッシュ１６ａ、ｂ内にある可能性があるが、このような状態

の発生は確実ではないか又は期待できない。従って各セルは潜在的には、９個のセルを記述するデータをメモリ（１６、１８、２０）間で移動させることを要求する。換言すれば、あるセルの検討は、別の８個のセルの移動のオーバーヘッドを招く。同様に、あるセルの処理は、周囲の８個のセルのデータを検討する。

【００２３】

本実施形態では、セルはブロック２１０のレベルにおいて処理される。図２は、６４個のセルを内包するように８×８となったブロックを示す。ブロック２１０の他の実施形態は、４×４、１６×１６、又は３２×３２のセルの大きさである。しかしながら、他のサイズのブロックも想定されることを理解されたい。簡単にするために、４×４のブロック４１０について議論する（図４）。一般に、１辺の長さが２のべき乗個のセルとなるブロックを考察する。

【００２４】

ブロック４１０は、どのようなチェッカーの色又はパリティであるかによって分類される。図４に示すように、灰色のブロックはチェッカー値０を有し、白色のブロックはパリティ値１を有する。パリティ値は：

パリティ＝（ブロックＸ＋ブロックＹ）ｍｏｄ２

によって計算され、ｍｏｄ２は、奇数値に対して操作を行う場合は１の値を返し、偶数値に対して操作を行う場合には０の値を返す、よく知られた操作である。

【００２５】

これにより、同様のパリティのブロックが対角線方向にしか互いに隣接しない「チェッカーボード（ｃｈｅｃｋｅｒｂｏａｒｄ）」タイプのパターンが生成される。同様のパリティを有するブロックは、同一のパリティを有する他のブロックとエッジを共有しない。実際にはデータに色分けがあるわけではなく、ブロック４１０に割り当てられるパリティ値が存在するだけであることを理解されたい。ここでは理解を容易にするために、色分けが議論及び図示されている。このパリティの割り当ての有用性については、以下で更に議論する。

【００２６】

各ノード／ブロックはキー値を含む。キー値は、ノードの優先順位（又は関連するルート設定情報を提供する上で考えられる重要性）を示す。キー値が低いノードは高い重要性を有するものとして示され、より早期に処理されるようスケジュール設定されることになる。

【００２７】

所与のルート設定作業に関して、システム１０はコストマップ３００を用いてトラバーサルマップを生成する。ルート生成器８０は例えば、コードを実行するプロセッサ１４である。しかしながら、ルート生成器８０が別個の論理である実施形態も想定される。ルート生成器８０はトラバーサルマップ１５００を生成し、これは、そのコストマップとの１：１セル対応を有する（コストマップ３００はトラバーサルマップ１５００に対応しない）。ルート生成器８０の動作を図５に示す。探索初期化により、コストマップと同一サイズの初期トラバーサルマップを生成し、ルート探索の開始に備える（要素５１０）。ルートフラッディングは、ルート開始点から探索領域をフラッディングし、上記トラバーサルマップを、ルート始点からルート終点まで到達するための累積コストを表す有限値で更新する（要素５２０）。本明細書中で使用される場合、用語「フラッディング（ｆｌｏｏｄｉｎｇ）」及び「フラッディングする（ｆｌｏｏｄ）」は、ルートを生成してルートマップを配置するプロセスを指す。図１５の模式図では、ルート探索、生成及び／又は配置が、ルート始点から、まるでそこに水源が存在し、そこから水が流れ出て探索領域を溢れさせるかのように、外側に拡大されていくことを理解されたい。遡行初期化により、決定されたルートをルートフラッダー（ｆｌｏｏｄｅｒ）が遡行する準備を行う（要素５３０）。遡行をルートフラッダーによって実施し、探索領域をルート終点から始点に戻るようにたどり、トラバーサル値の符号を正から負に反転させる（要素５４０）。ルートのベクトル化は、負のトラバーサル値の流れをたどり、最終的なルートを表す座標のリストを構築

する（要素５５０）。全体として、トラバーサル値は、地形移動コストデータと、ルート終点への進行を示すデータとを、セル毎に検討して、検討された各セルに関するトラバーサル値に到達することを理解されたい。

【００２８】

探索イニシャライザー８５による探索初期化（５１０）は、システム１０を初期化することにより、ルートフラッディングの動作及びルート探索の実施を提供する。フラッドモードは、後でルートのスウィープに使用されるトラバーサルモードに設定される（フラッドモードはトラバーサル又は遡行に設定できる）。トラバーサルマップは、コストマップと１：１のセル比を有するマップを生成することによって初期化される。上記トラバーサルマップは、全てのセルに無限の値（走行不可能）が与えられるように設定される。現在のノードのリストを、上記リストが開始位置を含むノードを含むように生成する。あるブロックにわたって探索を実施する場合、ブロックのある辺から別の辺へと処理を行う（スウィープ）によって、セルを検討する。スウィープは４方向（上、下、左、右）に実施できる。初期化により、４つのスウィープ方向全てが可能となるようにも設定される。このスウィーププロセスについては、以下で更に議論する。

【００２９】

ルートフラッダー９０によるルートフラッディング（要素５２０）は、「第１の草案（ｆｉｒｓｔ　ｄｒａｆｔ）」ルートを生成するプロセスである。ルートフラッディングは、現在のノードのリスト内に未処理のノードが存在する限り（要素６１０）、及び最大反復回数を超えない限り（要素６２０）、行われる。許容される反復の回数は、ルート始点と終点との間の距離に基づいて変化する。ルートフラッディングの第１の部分は、現在の探索に最も関連しそうな領域がどれかを理解することである。これは、ブロックプライオリタイザーを用いて達成される。上記ブロックプライオリタイザーは、現在のノードのリスト中のノードを、現在処理されるべきノードと、後で処理されるべきノードとに分ける（要素６３０）。現在処理されるべきノードは、現在のノードのリスト内にとどまる。後で処理されるべきノードは、将来のノードのリストに移動される。

【００３０】

本実施形態では、ノードはブロックである。本実施形態は、セルのエッジではなくセルに関して、コスト及びトラバーサル値を追跡する。従って、以前のルート設定操作が各セルに関して８個（側部及びコーナーが隣接するトラバーサル点それぞれに関して１個）のトラバーサル値を生成していたが、本実施形態は各セルに関して、８個ではなく単一の値で動作する。

【００３１】

現在のノードのリスト内の各ノードに関して、ブロックを処理する（要素６４０）。この実行により、与えられたブロックに関してトラバーサルマップが更新され、影響を受けたいずれの近隣のブロックに関するノードが、隣接するノードのリストに追加される。ＧＰＵ１４では、現在のノードのリスト中のブロックを、それぞれの（以下で議論されるパリティチェックを既に受けた）ストリーミングマルチプロセッサ上で並列に実行できる。

【００３２】

現在のノードのリスト中の全てのノードを処理した後（現在のノードのリスト内にノードがそれ以上存在しない（要素６１０））、将来のノードのリスト及び隣接するノードのリストが空ではない（要素６６０）限り、将来のノードのリスト及び隣接するノードのリストからのノードを現在のノードのリストに移動させて処理する（要素６５０）。

【００３３】

ノード／ブロックの処理（要素６４０）に関して、図７を参照して更なる詳細を示す。まず、現在のブロックに関するトラバーサルマップをロードする（要素７１０）。上述のように、初期化時に、上記トラバーサルマップには、全てのセルに関する無限値が配置される。あるブロックをロードする際、この要素８１０の全てのセルを、１セル幅のパディングリング８２０と共にロードする（図８）。パディングリング８２０は、実際に（側方及び対角線方向に）隣接するブロックの一部であるセルから構成される。トラバーサルセ

ルの変化はその近隣のセルのうちのいずれかに影響を及ぼし得るため、トラバーサルブロックのエッジに沿ったセルにおけるいずれの変化は、特定の隣接するトラバーサルブロックのセルに影響を及ぼし得る。ある隣接するブロックに影響を及ぼす、トラバーサルブロックのエッジのセルのセットは、この隣接するブロックの影響領域の一部であると表現される。従って、要素８１０内のセルに影響を及ぼす可能性のあるセルの包括的なリストが処理の一部として存在するように、パディングリング８２０がロードされる。セルを一度に１個処理する場合、パディングリングのロードにより、１個のセルに対して８個のパディングのセルが提示される（即ちパディングのオーバーヘッドと処理されるセルとの比が８：１である）。この（少なくとも）６４セルのブロックの例は、ブロックの６４個のセルに対して、３６個（以下）のパディングのセルを提示する。これは、０．５６２５：１という、パディングのオーバーヘッドと処理されるセルとの比を提示する。従ってブロックの処理は、およそ１４倍のオーバーヘッドの低減を提供する。よって本実施形態は、８：１未満であるパディングとセルとの比を考察する。実際には、１６個のセルを有するブロックでは、パディングは２０個以下のセルを有する。一般に、パディングに必要なセルの個数は、ブロックの幅（セルの個数で表される）の４倍に（コーナーの）４を加えたものである。

【００３４】

次に、現在処理されているブロックに対応するコストマップもロードする（要素７２０）。コストマップもまた同様に、１セル幅のパディングリングと共にロードされる。次にブロックのハッシュ値をゼロに設定する（ブロックのハッシュについては以下で更に詳細に議論する）（要素７３０）。

【００３５】

システム１０は、優先順位指定済みリスト中のノードを用いて、探索作業項目をスケジューリングする。作業項目はエッジを用いずにトラバーサルセルのブロックとして表され、これにより、作業項目の数及びスケジューリングのオーバーヘッドが低減される。換言すれば、システム１０は本質的にラスターベースであり、各ブロックを単一のノードとして表現する。ルートマッピングシステムの他の実施形態は、グラフのベクトルベースの表現、即ち全ての頂点及びエッジの明示的なセットに依存し、これらの実施形態は接続された各縁部に対して１個ずつの８個のノードを用いて各セルを表現するが、各ブロックを単一のノードとして提供することにより、約８倍の（各セルエッジを考慮する必要がないため、１ブロック中のセルの個数倍、例えば８×３２×３２＝８１９２倍の）メモリ要件の低減が提供される。このメモリ要件の低減により、計算スループットは大幅に改善され、サポートできるルート探索のサイズが増大する。

【００３６】

上述のように、各ブロックは８個の隣接するブロックを有する。現在のブロックの外縁部のセルに関するトラバーサル値の変化は、隣接するブロック内のセルに関する値に影響を及ぼす可能性がある。ハッシュ値は、これらの外側のセルに関するトラバーサル値が変化する時を追跡し、ハッシュ値をこのブロックに関して調整する。従ってハッシュ値は、現在のブロックに対する操作によってどの隣接するブロックが変化する可能性があるかを追跡する。よってハッシュ値の変化により、適切な隣接するブロックが識別され、「隣接するノード（Ａｄｊａｃｅｎｔ　Ｎｏｄｅ）」のリストに入れられる。

【００３７】

よって、ブロックの処理はまず、ブロックのハッシュをゼロに設定する（要素７３０）。次に、処理されるブロックに関するトラバーサルマップの初期状態に基づいて、前処理ハッシュが設定される（要素７４０）。ハッシュは、ブロック内のデータにハッシュ関数を適用して、各セル内のデータを表す値を出力することによって設定される。次に終点をシードする（要素７４５）。終点のシードは、ルートの根源の場所の値を、イプシロンのトラバーサル値（1.4×10^{-45}）を有するように設定することを含む。（以下で議論する遡行モードの場合、終点のシードは、ルート終点に関連付けられた値の（正から負への）反転を含む）。（以下で議論する）スウィープ操作を実施する際、トラバーサルマッ

プに対するいずれの修正が必要に応じて行われる（要素７５０）。ブロックが処理されると、ハッシュ操作をトラバーサルマップに再び適用して、トラバーサルマップのハッシュを生成する（後処理、要素７６０）。この後処理ハッシュを、いずれの正の値が負になり、いずれの負の値が正に変わるように、反転させる。続いてこの後処理ハッシュを前処理ハッシュに加える。この操作は、前処理ハッシュと後処理ハッシュとを比較する役割を果たす（要素７７０）。所与のセルに対して変化がない場合、前処理ハッシュと反転された後処理ハッシュとは、上記加算中に互いを相殺する。従って、前処理ハッシュと後処理ハッシュとの合計がゼロでない、影響領域内のいずれのセルは、変化したものとして識別される。このような変化したセルによって影響される可能性のある隣接するブロックは、影響される各ブロックに関して新たなノードを生成することによって更に処理されることになる隣接ブロックとして識別される（要素７８０）。このような影響されるブロックを、隣接するノードのリストに加える（要素７９０）。

【００３８】

ある好適なハッシュ関数は、ＩＥＥＥ‐７５４仕様の３２ビット浮動小数点から３２ビット整数へのビット単位コピーであり、これはＣ／Ｃ＋＋及び他の同様のプログラミング言語での演算である：

```
int HashFunction(float traversal) {
        int hash = *(int *)(&traversal);
        return hash;
        }
```

図１０は、終点シード操作（要素７４５）の更なる詳細を示す。終点のシードはまず、採用するフラッドモード（トラバーサル又は遡行）を決定する（要素１０１０）。遡行モードに関しては後述する。フラッドモードがトラバーサルである場合、システム１０は、ルートの始点が現在のトラバーサルブロック内にあるかどうかを決定する（要素１０２０）。ルートの始点が現在のトラバーサルブロック内にある場合、この始点セルにイプシロン（ε）の値を与える（要素１０３０）。これは、ゼロよりも明確に大きい、最小の数である。コストは全くかからず、いかなる距離も移動しない（始点に到達する）。しかしながら上記システムは、遡行フラッドモードでは正の数を負の数から区別するため、εの使用によって両方の目的が達成される。

【００３９】

フラッドモードが遡行である場合、システム１０は、ルートの終点が現在のトラバーサルブロック内にあるかどうかを決定する（要素１０４０）。ルートの終点が現在のトラバーサルブロック内にある場合、この終点セルに、反転させた値を与える（トラバーサル値５は－５に変更される）（要素１０５０）。

【００４０】

図９Ａ～Ｂは、スウィープ操作（図７の要素７５０）の更なる詳細を示す。ブロックのスウィープは、最高４回のスウィープ（上／北「Ｎ」、下／南「Ｓ」、左／西「Ｗ」、及び右／東「Ｅ」）を含む可能性がある。最初にブロックを受け取ると、４回のスウィープ全てを実施するようにスケジューリングする。従って該ブロックは、スウィープレジスタ内に、４回全てのスウィープ（Ｎ、Ｓ、Ｅ、Ｗ）が保留中であることを表す、該ブロックに関連付けられた指標を有する。過去の処理と、その後の１つ以上の隣接するブロックの処理とが、このブロックが変化した可能性があることを示すことを理由として、このブロックが処理のために入力された場合、スウィープレジスタによって、全てのスウィープより少ないスウィープをスケジューリングしてよい。

【００４１】

このスウィーププロセスはまず、現在検討されているブロックに関していずれのスウィープが保留中であるかどうかを決定する（要素９１０）。少なくとも１つのスウィープが保留中である場合、システム１０はスウィープの実行を開始する。「スウィープの実行（ｅｘｅｃｕｔｉｎｇ　ｓｗｅｅｐ）」に関する更なる詳細は、図１１を参照して後に提供する。システム１０はまず、「Ｎ」のスウィープが保留中であることを確認するためにチ

ェックを行う（要素９１５）。保留中でない場合、システム１０は、「Ｓ」のスウィープが保留中であるかどうかの確認に進む（要素９４０）。

【００４２】

Ｎのスウィープが保留中である場合、Ｎのスウィープを実行する（要素９２０）。結論として、スウィープレジスタを、Ｎのスウィープが完了したことを反映するように更新する（要素９２５）。次にシステム１０は、Ｎのスウィープがいずれのトラバーサル値を変化させたかどうかを決定する（要素９３０）。いずれのトラバーサル値も変化していなかった場合、システム１０は、「Ｓ」のスウィープが保留中であるかどうかの確認に進む（要素９４０）。Ｎのスウィープが少なくとも１つのトラバーサル値を変化させていた場合、スウィープレジスタを、Ｅ及びＷのスウィープを保留中としてマークするように更新する（要素９３５）。Ｅ及びＷのスウィープは既に保留中として留意されている場合があることを理解されたい。続いてシステム１０は、「Ｓ」のスウィープが保留中であるかどうかの確認に進む（要素９４０）。

【００４３】

Ｓのスウィープが保留中でない場合、システム１０は、「Ｅ」のスウィープが保留中であるかどうかの確認に進む（要素９６５）。Ｓのスウィープが保留中である場合、Ｓのスウィープを実行する（要素９４５）。結論として、スウィープレジスタを、Ｓのスウィープが完了したことを反映するように更新する（要素９５０）。次にシステム１０は、Ｓのスウィープがいずれのトラバーサル値を変化させたかどうかを決定する（要素９５５）。いずれのトラバーサル値も変化していなかった場合、システム１０は、「Ｅ」のスウィープが保留中であるかどうかの確認に進む（要素９６５）。Ｓのスウィープが少なくとも１つのトラバーサル値を変化させていた場合、スウィープレジスタを、Ｅ及びＷのスウィープを保留中としてマークするように更新する（要素９６０）。ここでも、Ｅ及びＷのスウィープは、開始値、Ｎのスウィープからの変化等によって、既に保留中として留意されている場合があることを理解されたい。続いてシステム１０は、「Ｅ」のスウィープが保留中であるかどうかの確認に進む（要素９６５）。

【００４４】

Ｅのスウィープが保留中でない場合、システム１０は、「Ｗ」のスウィープが保留中であるかどうかの確認に進む（要素９９０）。Ｅのスウィープが保留中である場合、Ｅのスウィープを実行する（要素９７０）。結論として、スウィープレジスタを、Ｅのスウィープが完了したことを反映するように更新する（要素９７５）。次にシステム１０は、Ｅのスウィープがいずれのトラバーサル値を変化させたかどうかを決定する（要素９８０）。いずれのトラバーサル値も変化していなかった場合、システム１０は、「Ｗ」のスウィープが保留中であるかどうかの確認に進む（要素９９０）。Ｅのスウィープが少なくとも１つのトラバーサル値を変化させていた場合、スウィープレジスタを、Ｎ及びＳのスウィープを保留中としてマークするように更新する（要素９８５）。続いてシステム１０は、「Ｗ」のスウィープが保留中であるかどうかの確認に進む（要素９９０）。

【００４５】

Ｗのスウィープが保留中でない場合、システム１０は、いずれのスウィープが保留中であるかどうかの確認に戻る（要素９１０）。Ｗのスウィープが保留中である場合、Ｗのスウィープを実行する（要素９９２）。結論として、スウィープレジスタを、Ｗのスウィープが完了したことを反映するように更新する（要素９９４）。次にシステム１０は、Ｗのスウィープがいずれのトラバーサル値を変化させたかどうかを決定する（要素９９６）。いずれのトラバーサル値も変化していなかった場合、システム１０は、いずれのスウィープが保留中であるかどうかの確認に進む（要素９１０）。Ｗのスウィープが少なくとも１つのトラバーサル値を変化させていた場合、スウィープレジスタを、Ｎ及びＳのスウィープを保留中としてマークするように更新する（要素９９８）。続いてシステム１０は、いずれのスウィープが保留中であるかどうかの確認に進む（要素９１０）。

【００４６】

最終的に、スウィープは、トラバーサル値に変化を発生させなくなり、これによりトラ

バーサル値は定常状態に到達する。これが起こると、変化がないことにより、全てのスウィープを完了させることができ、他のスウィープを要求することはなくなる。保留中のスウィープが存在しない状態で、システム１０はトラバーサル値の保存に進む（要素９１５）。保存を行うと、システム１０は動作を継続し、後処理ハッシュを実行する（要素７６０）。

【００４７】

上述のように、多くの場合、スウィープはトラバーサル値を変化させ、具体的にはトラバーサル値を、所与のブロックのエッジに沿って変化させ、このトラバーサル値の変化が、隣接するブロック内のセルに関する計算に影響を及ぼす可能性がある。このような影響されたブロックを、隣接するノードのリストに加える（要素７９０）。隣接するノードのリストにブロックを加えるステップは、加えられるノードに関するプロパティを確立するステップを含む。これらのプロパティとしては、キー値（これは上記ノードの優先順位を決定する）、上記ノードの場所（潜在的には座標のセット）、及び初期スウィープ（該ブロックの処理時に実施されるスウィープ操作）が挙げられる。

【００４８】

キー値は、例えば以下の式によって決定される：

キー＝ｍｉｎ（影響領域内のトラバーサルセル）＋最小コスト＊到達距離。

【００４９】

「影響領域内のトラバーサルセル（ｔｒａｖｅｒｓａｌ　ｃｅｌｌｓ　ｉｎ　ｒｅｇｉｏｎ　ｏｆ　ｉｎｆｌｕｅｎｃｅ）」は、現在のブロック内の、検討対象の隣接するセルの境界をなすセルであり、従って、隣接するブロックがロードされていた場合、この「影響領域内のトラバーサルセル」も、上記ブロックに関するパディングリングの一部としてロードされていることになる。８×８セルのブロックにおいて、側方に隣接するブロックに関して、システム１０は８個の、影響領域内のトラバーサルセルを予想する。対角線方向に隣接するブロックに関しては、システム１０は１個の、影響領域内のトラバーサルセルを予想する。よって上記式の「ｍｉｎ（　）」の部分は、関連するグループから最小のトラバーサル値を有するセルを見出し、その値を使用する。

【００５０】

「最小コスト（Ｍｉｎｉｍｕｍ　Ｃｏｓｔ）」は、可能な最小コストのセルの値である。一例として、可能な最小コスト値は「５」である。到達距離は、影響領域内のあるセルから、隣接するブロック内の最も近いセルへ移動するための距離（セルの長さを単位とする）である。北、南、西又は東の直近に隣接するブロック（側方に隣接）に関しては、到達距離は１である。４つの対角線方向に隣接するブロック（対角線方向に隣接）に関しては、到達距離は$\sqrt{2}$である。

【００５１】

上述のように、「スウィープ操作（ｓｗｅｅｐｉｎｇ）」は、トラバーサルセル値を更新するものである。このスウィープ操作は、４方向（Ｎ、Ｓ、Ｅ、Ｗ）で行われる（要素９２０、９４０、９７０、９９２）。図１１は、スウィープ操作が何を伴うかについての更なる詳細を提供する。ブロックのスウィープ操作は、ブロックを上記４方向のうちの１つにおいてスウィープすることに焦点を当てており、各スウィープは等しく、各セルに関する８個の近隣のセルのうちの３つを評価する。

【００５２】

まず、いずれのトラバーサルブロックセルがブロックのスウィープによって修正されたかどうかを示すブール値であるＳｗｅｅｐｅｒ　Ｃｈａｎｇｅｄ　ＴｒａｖｅｒｓａｌがＦａｌｓｅに設定される（要素１１１０）。この時点において、システム１０は、ランク値を後方から前方へと反復するための外側のｆｏｒループを確立する（要素１１２０）。ここでランクは、Ｗ若しくはＥのスウィープに関するブロック内のセルの列、又はＮ若しくはＳのスウィープに関するブロック内のセルの行である。プロセッサＸＸがＣＰＵである場合、全てのセル（行）を通して反復する内側のループが存在する（要素１１３０）。プロセッサＸＸがＧＰＵである場合、ある行の全てのセルが、それぞれの別個のストリーミ

ングプロセッサ上で同時に実行される。処理時、各セルに関して、「後方（ｂａｃｋ）」「後方左（ｂａｃｋ　ｌｅｆｔ）」及び「後方右（ｂａｃｋ　ｒｉｇｈｔ）」セルが識別される（要素１１４０）。方向「後方」、「後方左」及び「後方右」は、スウィープの方向を基準とする。また、トラバーサルセルの現在の値を「現在のセル（ｃｕｒｒｅｎｔ　ｃｅｌｌ）」に設定することによって、変化したトラバーサル値を後で決定できる。

【００５３】

後方セルからのトラバーサル値は、以下のように決定される（要素１１５０）：

後方からのトラバーサル＝トラバーサル（後方セル）＋１／２＊（コスト現在のセル）＋コスト（後方セル））

後方左セルからのトラバーサル値は、以下のように決定される：

後方左からのトラバーサル＝トラバーサル（後方左セル）＋√２／２＊（コスト（現在のセル）＋コスト（後方左セル））

後方右セルからのトラバーサル値は、以下のように決定される：

後方右からのトラバーサル＝トラバーサル（後方右セル）＋√２／２＊（コスト（現在のセル）＋コスト（後方右セル））

続いて、「後方」、「後方左」及び「後方右」セルから現在のセルに到達するためのコストを計算して、最小コストの可能性を見出すことにより、現在のトラバーサル値を決定する。これを実施するために、上記システムは、現在のフラッドモード（要素１１６０）に応じて、トラバーサルセルオプティマイザー（要素１１７０）又はトラバーサルセルリトレーサー（要素１１８０）を動作させて、古い値及び３つの後方トラバーサル値に基づいて適切な新しいトラバーサルセル値を計算する。現在のトラバーサルセルが修正されている場合、いずれのモジュールは、Ｓｗｅｅｐｅｒ　Ｃｈａｎｇｅｄ　ＴｒａｖｅｒｓａｌをＴｒｕｅに設定する。

【００５４】

所与のランクの値は、１つ前のランクからの値のみに左右されることを理解されたい。従ってあるランク内の複数のセルをＧＰＵで同時に処理できる。というのは、同時処理されるセルは互いに独立しているためである。

【００５５】

フラッドモードがトラバーサルである場合、トラバーサルセルオプティマイザーをブロックのスウィープに使用して、トラバーサルセルがどのように値を変更し、最良のルートの探索を伝播するかを定義する。これは以下のように動作する：

まず、システム１０は新しいトラバーサル値を評価する：

トラバーサル（現在のセル）＝ｍｉｎ（｛古いトラバーサル，後方からのトラバーサル，後方右からのトラバーサル，後方左からのトラバーサル｝）。

【００５６】

ａｂｓ（トラバーサル（現在のセル）－古いトラバーサル）＞トレランスである場合、Ｓｗｅｅｐｅｒ　Ｃｈａｎｇｅｄ　Ｔｒａｖｅｒｓａｌの値をＴｒｕｅに設定する。これによりシステム１０は、このトラバーサルブロックの既知の最良の経路が更に改善されていること、及び異なる複数の方向の更なるスウィープが要求されていることを把握する。

【００５７】

フラッドモードが遡行である場合、以下で更に議論するように、ブロックスウィーパーによってトラバーサルセルリトレーサーが使用される。

【００５８】

隣接するノード中の各ノードに関して、システム１０は、ノードに関するヒューリスティック関数を計算し、これをノードのキーに加える（要素６４２）。公知のＡ＊アルゴリズムと同様に、キーは、それまでに累積された合計コストと、目標までのヒューリスティックコスト又は推定コストとを足したものを表す。隣接するノード中のノードのキーは合計コスト又はトラバーサルのみを表す。

【００５９】

ヒューリスティック関数は、ルートの最後のセルから、ブロック内の最も近いセルまで

の距離に、最小コストマップを掛けたものとして計算される。これを計算するために、システム１０は、水平距離ｄｘ＝ｍａｘｆ$_0$（｛終点Ｘ－（ノードＸ＊Ｂ＋（Ｂ－１）），（ノードＸ＊Ｂ）－終点Ｘ，０｝）を計算し、ここで中括弧内の第１の項は、ルートの終点がブロックの東側の場合のｄｘを決定し、第２の項は西側の場合であり、第３の項は終点がブロックの西側及び東側境界内にある場合である。次にシステム１０は、計算垂直距離ｄｙ＝ｍａｘｆ$_0$（｛終点Ｙ－（ノードＹ＊Ｂ＋（Ｂ-１）），（ノードＹ＊Ｂ）－終点Ｙ，０｝）を計算する。続いてシステム１０は、距離ｄ＝√（ｄｘ＾２＋ｄｙ＾２）を計算する。そしてヒューリスティック関数は、ｄ＊最小コストとして計算される。

【００６０】

ルートフラッダーが現在のノード中の全てのブロックを実行した後、現在のノードをクリアできる（要素６４５）。残りの作業は、将来のノードのリスト及び隣接するノードのリストに見出される。次のブロックプランナーは、隣接するノード中のノードに優先順位を付け、現在のノードのリストを、隣接するノードのリストと将来のノードのリストとを連結したもので置換し、ブロックリデューサーを呼び出すことによっていずれの重複したノードをマージする（要素６５０）。

【００６１】

隣接するノードの加算（要素７９０）及び次のブロックのプランニングにより、更なるノードが現在のノードのリストに、他のノードがリスト内に既にあることを検討することなく加えられるため、重複したノードが存在する場合がある。ブロックリデューサーは、同一ブロックを表す複数のノードインスタンスがどこに存在するかを発見し、これらを１つにまとめる。複数のノードをまとめる際、ブロックリデューサーは、まとめられたインスタンスが元のインスタンスの情報を十分にキャプチャしていることを提供する。例えばノードＡが北及び東の初期スウィープを有し、ノードＢが南及び東の初期スウィープを有していた場合、まとめられたノードは、北、南及び東の初期スウィープを有する。同様に、ノードＡが１００のキーを有し（優先順位が高い）、ノードＢが８００のキーを有する（優先順位が低い）場合、まとめられたキーは１００となる。

【００６２】

より具体的には、ブロックリデューサーは図１２に示すように動作する。ブロックリデューサーは、現在のノードのリストに何個のノードが存在するかを確認するためにチェックを行う。現在のノードのリスト中に２個未満のノードが存在する場合（要素１２１０）、要素１２６０までスキップする。３個以上のノードが存在する場合、ブロックリデューサーはノードのＳｏｒｔＢｙ値を、ノードＸ　ＸＯＲ（ノードＹ＜＜１３）に等しくなるように設定し続け、ここでＸＯＲは論理排他的ＯＲ演算であり、「＜＜」は論理ビットシフトレフト演算であり、これらの演算はいずれも、いずれのソフトウェアプラットフォームで容易に利用可能である。ＸＯＲ及び＜＜演算は、ほとんど全てのプロセッサで利用可能な周知の演算である。ＳｏｒｔＢｙプロパティはハッシュコードとして機能し、同一のブロックを表すいずれのノードが同一のＳｏｒｔＢｙ値を有することができる。これを現在のノード中の各ノードに対して行う（要素１２１５）。

【００６３】

次に、現在のノードのリスト中のノードを、そのＳｏｒｔＢｙプロパティによってソートする（要素１２２０）。例えばＣＰＵ上でのＱｕｉｃｋｓｏｒｔ又はＧＰＵ上でのＢｉｔｏｎｉｃ　Ｓｏｒｔといったプログラムを、このプロセスのために実例として使用する。ソートの方向は問題にならない。問題は、同一のブロックに関するノードが、ソート済みのリスト内で互いに隣接することである。

【００６４】

次にブロックリデューサーは、２つのノードポインタを初期化する。ノードＡは、現在のノード中の第１のノードで始まり、ノードＢは第２のノードで始まる（要素１２２５）。

【００６５】

次にシステムは、ノードＢが現在のノードの最後にあるかどうかをチェックする（要素１２３０）。ブロックリデューサーが現在のノードの最後にある場合、要素１２６０に進

む。ブロックリデューサーが現在のノードの最後にない場合、ブロックリデューサーは、ノードＡの値がノードＢの値に等しいかどうかを確認する（要素１２３５）。これらが等しくない場合、ノードＡをノードＢの値に設定し（要素１２４０）、ノードＢを、現在のノード中の次の値に向くように１だけ増分させる（次のノードＡは以前のノードＢとなる）（要素１２４２）。ノードＡ及びノードＢが同一の値を有することが分かる場合、重複するノードが発見されている。ノードマージャをこれらのノードに対して動作させて、これらのノードのプロパティをマージし、各ノードから情報（キー値、スウィープ）をまとめられたノードへと抽出し、これをノードＡとして保存する（要素１２４５、１２５０）。ノードＢに関するキー値は無限に設定され、これによって削除のためにマークされる（要素１２５５）。次にノードＢを、現在のノードのリスト中の次のノードに向くように前進させ、ブロックリデューサーは要素１２３０に戻る。

【００６６】

現在のノードのリストを処理した後、ブロックリデューサーは、無限のキー値を有する現在のノードのリスト中のノードの除去に進む（要素１２６０）。

【００６７】

図１３は、要素６３０（図６）のブロックプライオリタイザーの実行に関する更なる詳細を提供する。ブロックプライオリタイザーは、現在のノード中のノードに優先順位を付け、後で処理されるいずれのノードを将来のノードに移動させることによって、保留中のトラバーサルブロックのうちのいずれを、ブロックエグゼキューターにおいて次に処理するべきかを決定する。現在のノード中の残りのノードを選択して、これらをいずれの順序で、又は並列でも、ブロックエグゼキューターによって処理できる。選択されるノードは、最小のキーを有する（即ち優先順位が最も高い）ノード及び同様の小さなキーを有する他のいずれのノード（要素１３１０）、並びに同一のブロックパリティ（図４を参照）を有するノード（要素１３２０）を含む。キーがどの程度類似しているかは、キーカットオフ値によって決定され、これは所望の性能のために実験によって微調整できる（要素１３３０）。図示されている例では、カットオフは、最小コスト値に「Ｂ」の２５倍を掛けたものに等しい（Ｂはあるブロックのあるランク（又はファイル）内のセルの数である。数２５は、好適な個数のブロックを、ブロックの再処理を過剰に必要とすることなく処理できるようにするために、経験的に選択されたものである）。同一のパリティの全てのノードを選択することにより、これらがエッジ上で互いに接触しない（コーナーにおいてのみ接触する）ことを保証して、これらの相互依存性を最小化し、並列実行をサポートする。ＭｉｎＰａｒｉｔｙ値はＰａｒｔｉｔｙ（ＭｉｎＮｏｄｅ）に設定され、これにより、最小コストのノードが、次の処理ウェーブで使用するパリティを定義する（要素１３２０）。カットオフキーの設定後、将来のノードのリスト及び隣接するノードのリストをクリアする（要素１３４０）。次に、現在のノードのリスト中のどのノードが、カットオフキーより大きなノードキーを有するか、又はＭｉｎＰａｒｉｔｙとは異なるパリティを有するかを決定する（要素１３５０）。このようなノードを現在のノードのリストから除去し、将来のノードのリストに付加する（要素１３６０）。

【００６８】

遡行初期化（要素５３０）は、遡行イニシャライザー９２によって実施される。遡行初期化は、システム１０を第２の実行のために再構成するか、又はルートフラッダーを、ルート遡行を実施するように再構成する。遡行イニシャライザーは、フラッドモードを「遡行（Ｒｅｔｒａｃｅ）」に設定する。遡行イニシャライザーはまた、現在のノードのリストを、ルート終点を含むブロックを有する単一のノードを内包するように設定し、また上記ノードに関するスウィープ方向を設定する。

【００６９】

次にシステム１０は、生成されたルートの遡行に進む。フラッドモードが遡行である場合、トラバーサルセルリトレーサーをブロックスウィーパー（ブロックスウィープコードを実行するプロセッサ）が用いて、最終的なルート内にあるいずれのトラバーサルセルをマークする（要素５４０）。ルート始点から全ての方向にトラバーサルマップを拡張する

トラバーサルセルオプティマイザーとは異なり、トラバーサルセルリトレーサーは、ルート終点から始点に戻るまでのトラバーサルセルの細い道筋のみを更新する。トラバーサルセルリトレーサーは、ルートの一部であるトラバーサルセルをマークし、その符号を反転させて負にする。これは以下のように動作する。

Ｉｆ：

１．古いトラバーサル＞０、かつ

２．以下のうちの１つがＴｒｕｅ：

ａ．ａｂｓ（古いトラバーサル＋後方セルからのトラバーサル）＜トレランス、

ｂ．ａｂｓ（古いトラバーサル＋後方左セルからのトラバーサル）＜トレランス、又は

ｃ．ａｂｓ（古いトラバーサル＋後方右セルからのトラバーサル＜トレランス、

Ｔｈｅｎ：

１．トラバーサル（現在のセル）＝－（古いトラバーサル）、即ち現在のセルを最終的なルートの一部としてマークする。

【００７０】

２．Ｓｗｅｅｐｅｒ　Ｃｈａｎｇｅｄ　ＴｒａｖｅｒｓａｌをＴｒｕｅに設定する。即ちルートエグゼキューターに、異なる複数の方向の更なるスウィープが必要であることを通知する。

【００７１】

上述の条件を更に説明すると：条件１は、現在のトラバーサルセルが最終的なルートの一部としてまたマークされていないことを保証することにより、二重に戻ること又はループに囚われることを防止する。条件２ａは、トレランスが実際にはゼロであるため、古いトラバーサルと後方セルからのトラバーサルとが等しくかつ反対であるかどうかをチェックする。条件１及び２ａがいずれも真である場合、後方セルからのトラバーサルが負であると推論でき、従って後方セルは最終的なルートの一部として既にマークされたものである。また、遡行プロセスの開始前に古いトラバーサルと後方セルからのトラバーサルとが等しく、従って現在のセルは後方セルへの最速の経路の一部であると結論付けることもできる。よって、トラバーサル（現在のセル）をマークするべきである。条件２ｂ～ｃは、これらが後方及び後方左の近隣のセルに適用される以外は、条件２ａと同一である。

【００７２】

次に、ルートベクトライザー９４によってルートをベクトル化する（要素５５０）。ルートベクトライザー９４は例えば、コードを実行するプロセッサである。ルートベクトライザー９４の動作に関する更なる詳細を図１４に提供する。ルートベクトライザー９４は、ルート生成のブロック及びセルから最終的なルートを定義する頂点のリストを生成する。ルートベクトライザー９４は、ルートの終点のトラバーサルセルで開始され、始点セルに向かって逆方向に動作する。各点において、ルートベクトライザー９４は、どの隣接するセルが最大の負のトラバーサル値を有するかを見出し、その点をルート点のリストに加え、その点へと進む。ルートの始点に到達すると、ルート点のリストを逆転させて、マッププセル座標に最終的なルートを生成する。

【００７３】

ルートベクトライザー９４はまず、ルート点をクリアする（要素１４１０）。続いてルート終点を始点に設定する（要素１４２０）。ルート終点（ベクトル化の始点）をルート点リストに付加する（要素１４３０）。次にルートベクトライザー９４は、これがルート始点に到達するかどうかを確認するためにチェックを行う（要素１４４０）。ルート始点に到達しない場合、ルートベクトライザー９４は、現在検討中のセル（最初はルートの終点）に隣接する全てのセルをチェックして、最大値の負の値（－イプシロンが最大値を有する負の値となる）を有するセルを見つける（要素１４５０）。発見された点を、ルート内の次の点として設定し（要素１４６０）、ルートに付加する（要素１４７０）。この要素１４５０、１４６０及び１４７０のプロセスを、ルートベクトライザー９４がルートの始点に到達するまで繰り返す。ルートベクトライザー９４がルートの始点に到達すると、ルートは、ルート終点からルート始点まで連なる頂点のリストで構成される。よって、次

にこれらのルート点を逆転させることによって、ルートを定義する頂点のリストを生成する（要素１４８０）。

【００７４】

長さ５０マイル（８０．４６７２ｋｍ）等の長いルートに関しては、探索する必要がある領域は極めて大きい。このような大きな領域を表すために十分に、コストマップ及びトラバーサルマップを大きくするには、相当量のメモリを割り当てる必要があり得る。ＢｌｏｃｋＳｗａｔｈは：１）マップ上のセルのいずれのブロックに対する迅速なランダムアクセスを提供し；２）メモリの無駄を削減するためにまばらなマッピングを提供することによって、大きなマップを表現するための、簡単で効果的なデータ構造である。

```
class BlockSwath
    {
        int BlockSize;
        int ChunkSize;
        int WidthChunks;
        int HeightChunks;
        float [ ][ ][ ] Data; // float ***Data in C/C++
        }
```

ＢｌｏｃｋＳｉｚｅはセル内の各ブロックの幅及び高さを記述する。ＢｌｏｃｋＳｉｚｅは２のべき乗になるはずである。ＣｈｕｎｋＳｉｚｅは、セル内の各チャンクの幅及び高さを記述する。ＣｈｕｎｋＳｉｚｅは２のべき乗になり、かつＢｌｏｃｋＳｉｚｅより大きくなるはずである。ＷｉｄｔｈＣｈｕｎｋｓは、スウォース（ｓｗａｔｈ）内のチャンクの列の数を記述する。ＨｅｉｇｈｔＣｈｕｎｋｓは、スウォース内のチャンクの行の数を記述する。Ｄａｔａは、実際のセルのデータを内包する不揃いなアレイを記述し、これらはＤａｔａ［ｃｈｕｎｋ］［ｂｌｏｃｋ］［ｃｅｌｌ］としてアドレス指定でき、ここでｃｈｕｎｋは、セルを内包するチャンクのインデックスを表し：

```
chunk = [x/ChunkSize] + [y/ChunkSize] * WidthChunks
```

ｂｌｏｃｋは、チャンク内のブロックのインデックスを表し：

$$block = [x/BlockSize] + [y/BlockSize] * ChunkSize)Mod\ ChunkSize^2$$

ｃｅｌｌは、ブロック内のセルのインデックスを表し：

$$cell = (x + y * BlockSize)Mod\ BlockSize^2$$

ｘ及びｙは、ブロック全体のスウォースの北西方向のコーナーからの、セルのｘ及びｙ座標である。

【００７５】

データをこのようにして表現することにより、到達されていないトラバーサルブロックを割り当てる必要がなくなり、ブロックあたり（例えば）４ＫＢのメモリが節約される。同様に、到達されていないチャンクを割り当てる必要がなくなり、（ＣｈｕｎｋＳｉｚｅを２５６バイトと仮定すると）チャンクあたり２５６バイトが節約され、これは、極めて巨大な探索領域に対処する場合には、最終的に大きな節約となる。

【００７６】

トラバーサルマップ１５００は、生成されたトラバーサル値、及びシステム１０によって生成されたルート１５１０の、視覚的な図である。マップ１５００は、始点「ｄ」及び終点「ｅ」を含み、これらの間にルート１５１０がある。共通の細線（細線１５２０、１５３０、１５４０、１５５０等）上に位置決めされた場所は、始点「ｄ」から同様の移動時間距離を有する場所を表す。

【００７７】

双方向探索を採用する実施形態も想定される。双方向探索は、２つのルート探索の生成を伴い、これらは１つずつ、ルートの各端部において開始され、これらが中央で出会った時に停止する。これは、探索される総面積が低減されるため、より効率的であり、またルートの端部が到達不可能な島に存在するような通行不可能なルートの検出に役立つ。

【００７８】

更に、多重分解能ルート設定を採用する実施形態も想定される。多重分解能ルート設定は、粗いコストマップを用いて粗レベルルートを生成するステップと、これを用いて、精細レベルでのルート生成を、前のルートの周りの細長い領域に沿って限定するステップとを含む。より明確には、一実施形態において、６４個のセルのブロックを、単一のコスト値を有する単一のエンティティとして扱う。上述のルート設定を、このような粗いブロックに関する値のみを用いて実施して、上記ブロックを、最終的なルートを内包する可能性が高まるように狭める。

【００７９】

更に、評価を、８個の隣接するセルへの移動よりも拡張して、１６点の隣接を検討する実施形態も想定される。これらの更なる隣接を図１６に示す。上記更なる隣接は、「チェス・ナイト（ｃｈｅｓｓ‐ｋｎｉｇｈｔ）」型の隣接を含む。８点の隣接に対して、１６点の隣接は、ますます滑らかなルート及び精密なコスト計算を提供できる可能性を有し、また角度付き方向の移動の選択肢が増加する。

【００８０】

ドッグレッグ（ｄｏｇｌｅｇ）除去を行う実施形態も想定される。ドッグレッグ除去は、コストが均一な（又は略均一な）領域にわたるルートの生成を試みる際に一般的なアーティファクトである、ギザギザの、即ち「ドッグレッグ」型の経路（例えば、ＮＮＥ方向にまっすぐ約１８０メートル進むのではなく、Ｎ方向に１００メートル進んだ後、ＮＥ方向に１００メートル進む経路）の排除を支援する。

【００８１】

先読みマップロードを採用する実施形態も想定される。先読みマップロード（Ｌｏｏｋ‐Ａｈｅａｄ　Ｍａｐ　Ｌｏａｄｉｎｇ）は、ルートの探索が、ロードされていないマップのタイルにどの程度近いかを確認して、どのマップのタイルを次にロードするかの優先順位を指定することによって、ルートの生成がマップのロードを待機しないことを保証する。

【００８２】

本発明を、例示的な設計を有するものとして説明したが、本発明は本開示の精神及び範囲内で更に修正してよい。従って本出願は、本発明の一般原理を用いた本発明のいずれの変形、使用又は改造を包含することを意図している。更に本出願は、本発明が属する技術分野の公知の又は慣例的な実施の範囲内にあるような、本開示からの発展形態を包含することを意図している。

以下、本発明の好ましい実施形態を項分け記載する。

実施形態１

ルート生成器を操作する方法であって、前記方法は：

第１のグループ内の複数のブロックに関して、ルートトラバーサル値を同時に計算するステップであって、各前記ブロックは複数のセルを含み、前記トラバーサル値は、地形移動コストデータと、ルート終点への進行を示すデータとを、セル毎に検討した値である、ステップ

を含み、

前記複数のブロックは、前記第１のグループ内の前記ブロックが、前記第１のグループ内の他のグループと、いずれのエッジを共有できないように選択される、方法。

実施形態２

前記複数のブロックに関する前記ルートトラバーサル値は、グラフィック処理ユニットによって同時に計算される、実施形態１に記載の方法。

実施形態３

前記ブロックはマップの一部であり、

各前記ブロックは、２つの極性の値のうちの一方が割り当てられる、実施形態１又は２に記載の方法。

実施形態４

第１の前記極性を有する各前記ブロックは、第２の前記極性のみを有する他の前記ブロックに対して側方に隣接する、実施形態３に記載の方法。

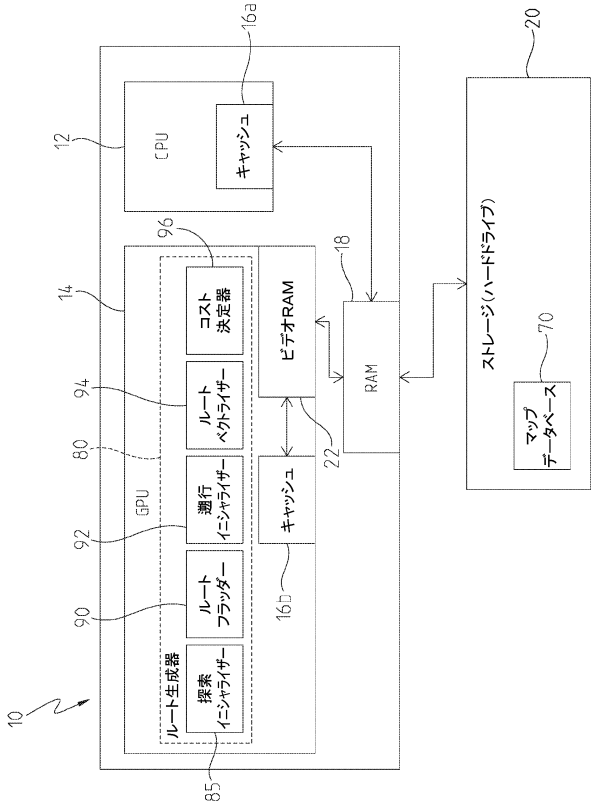
実施形態５
前記第１の極性を有する各前記ブロックは、前記第１の極性のみを有する他の前記ブロックに対して対角線方向に隣接する、実施形態３又は４に記載の方法。
実施形態６
前記第１のグループ内の前記ブロックは、共通の前記極性を共有する、実施形態３～５のいずれか１項に記載の方法。
実施形態７
いずれの前記ブロック内のいずれの前記セルに関する前記トラバーサル値を計算する際、前記ブロック内の全ての前記セルに関する前記トラバーサル値を計算する必要がある、実施形態１～６のいずれか１項に記載の方法。
実施形態８
ルート生成器を操作する方法であって、前記方法は：
コストマップのセルの第１のブロックに関するデータを、グラフィック処理ユニットのキャッシュメモリにロードするステップであって、前記第１のブロック内の１つの前記セルに関するデータのロードは、前記第１のブロック内の全ての前記セルに関するデータのロードを必要とする、ステップ；及び
前記セルの第２のセットをロードするステップであって、前記第２のセット内の各前記セルは、前記第１のブロック内にはなく、前記第１のブロック内のあるセルとエッジを共有するか、又は対角線方向に隣接する、ステップ
を含み、
前記第１のブロック内の前記セルの数と、前記第２のセット内の前記セルの数との比は、１：８未満であり、
前記セルの第１のブロックと前記セルの第２のセットとの組み合わせは、前記第１のブロックの全ての前記セルに関するトラバーサル値を生成するために必要な全てのマップデータを提供し、
前記トラバーサル値は、コストデータと、ルート終点への進行を示すデータとを検討した値である、方法。
実施形態９
前記第１のブロックに関する前記コストデータを処理することによって、前記第１のブロックに関する前記トラバーサル値を生成するステップを更に含む、実施形態８に記載の方法。
実施形態１０
前記第１のブロックは少なくとも１０２４個のセルを含み、前記第２のセットは２７２個以下のセルを含む、実施形態８又は９に記載の方法。
実施形態１１
いずれの前記ブロック内のいずれの前記セルに関する前記トラバーサル値を計算する際、前記ブロック内の全ての前記セルに関する前記トラバーサル値を計算する必要がある、実施形態８～１０のいずれか１項に記載の方法。
実施形態１２
前記セルの前記第１のブロックに関する前記トラバーサル値を生成するステップを更に含み、
前記セルの前記第１のブロックに関する前記トラバーサル値を生成するために使用されるコスト値のみが、前記セルの第１のセット及び前記第２のセットのコスト値である、実施形態８～１１のいずれか１項に記載の方法。
実施形態１３
生成された前記トラバーサル値に応答して、開始位置から目的位置までのルートを生成するステップを更に含む、実施形態８～１２のいずれか１項に記載の方法。
実施形態１４
各前記セルは、各前記セルに関連付けられた単一の前記コスト値及び単一の前記トラバーサル値のみを有する、実施形態８～１３のいずれか１項に記載の方法。

実施形態１５
ルート生成器を操作する方法であって、前記方法は：
ルートの決定の一部として第１のセルが処理されることになることを決定するステップ；
前記第１のセルが、少なくとも２つのセルを内包する第１のブロック内にあることを決定するステップ；
前記第１のブロック内の全ての前記セルを分析するために必要なデータを、第１の不揮発性メモリから、プロセッサによってアクセス可能な揮発性メモリにロードするステップであって、前記第１のブロック内の全ての前記セルを分析するために必要な前記データは、前記第１のブロック内のある１つの前記セルとエッジ及びコーナーのうちの少なくとも１つを共有する全てのセルを含む、ステップ；及び
前記第１のブロック内の全ての前記セルを分析するために必要な前記データを全て前記揮発性メモリにロードした後、前記第１のブロック内の全ての前記セルを分析するステップを含む、方法。
実施形態１６
ロードされる各前記セルは、各前記セルに割り当てられた単一のコスト値及び単一のトラバーサル値のみを有する、実施形態１５に記載の方法。
実施形態１７
前記第１のセルが処理されることになることを決定する前記ステップは、前記第１のセルを内包しない第２のブロックを処理するステップを含み、
前記処理は、前記第１のブロックが前記第２のブロックに隣接することを示す、実施形態１５又は１６に記載の方法。
実施形態１８
前記プロセッサはグラフィック処理ユニットであり、
複数の前記ブロックが同時に処理される、実施形態１５～１７のいずれか１項に記載の方法。
実施形態１９
各前記ブロックは、２のべき乗の個数の前記セルを含む、実施形態１５～１８のいずれか１項に記載の方法。
実施形態２０
前記揮発性メモリは、前記プロセッサ上のキャッシュメモリである、実施形態１５～１９のいずれか１項に記載の方法。
【符号の説明】
【００８３】
１０　ルート計画計算システム
１２　プロセッサ、中央演算処理装置
１４　プロセッサ、グラフィック処理ユニット、ＧＰＵ
１６　メモリ
１６ａ、１６ｂ　キャッシュメモリ、キャッシュ
１８　ＲＡＭ、メモリ
２０　ハードドライブ、メモリ
２２　ビデオＲＡＭ、メモリ
７０　マップデータベース
８０　ルート生成器
８５　探索イニシャライザー
９０　ルートフラッダー
９２　遡行イニシャライザー
９４　ルートベクトライザー
９６　コスト決定器
２００　土地被覆マップ
２１０、４１０　ブロック

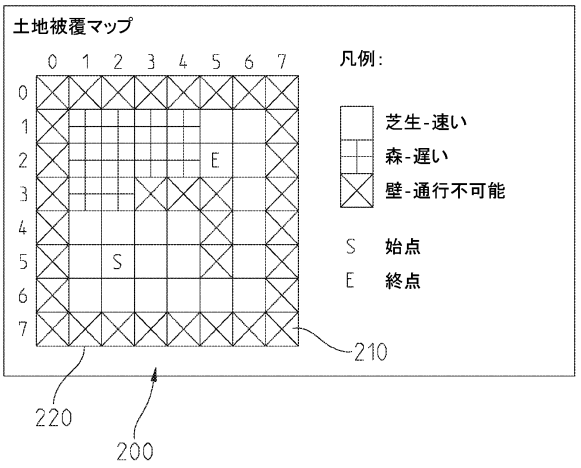
２２０　セル
３００　コストマップ
８１０　要素
８２０　パディングリング
１５００　トラバーサルマップ
１５１０　ルート
１５２０、１５３０、１５４０、１５５０　細線

【図面】

【図１】



【図２】



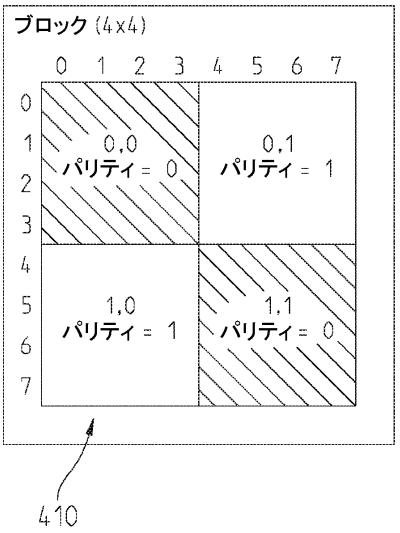
【図３】

コストマップ

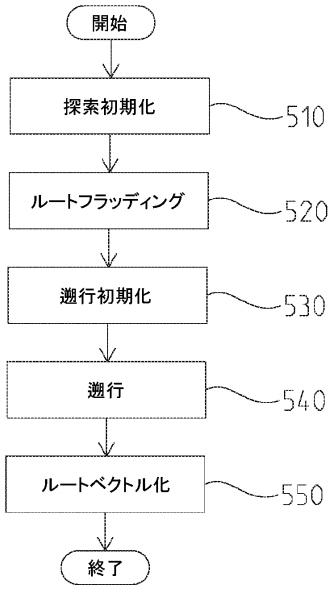
	0	1	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	15	15	15	15	5	5	∞
2	∞	15	15	15	15	5	5	∞
3	∞	15	15	∞	∞	∞	5	∞
4	∞	5	5	5	5	∞	5	∞
5	∞	5	5	5	5	∞	5	∞
6	∞	5	5	5	5	5	5	∞
7	∞	∞	∞	∞	∞	∞	∞	∞

300

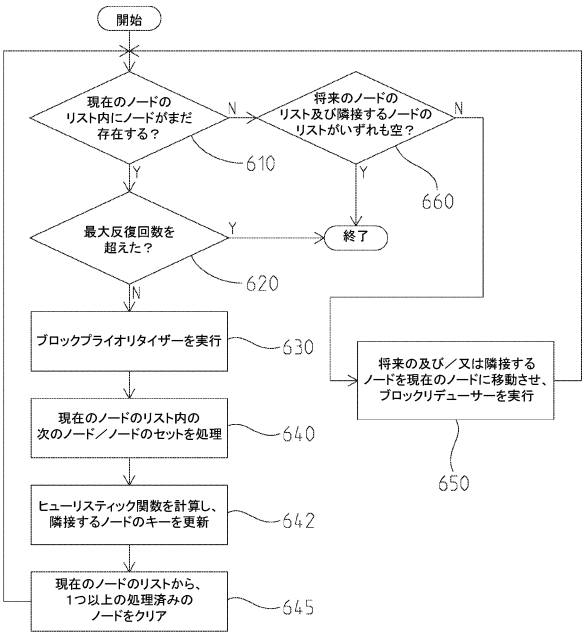
【図４】



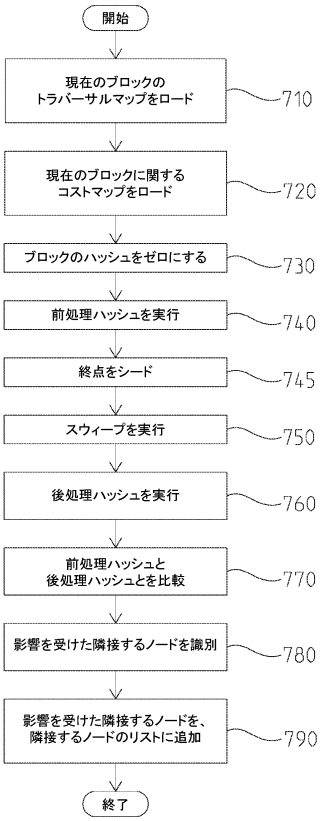
【図５】



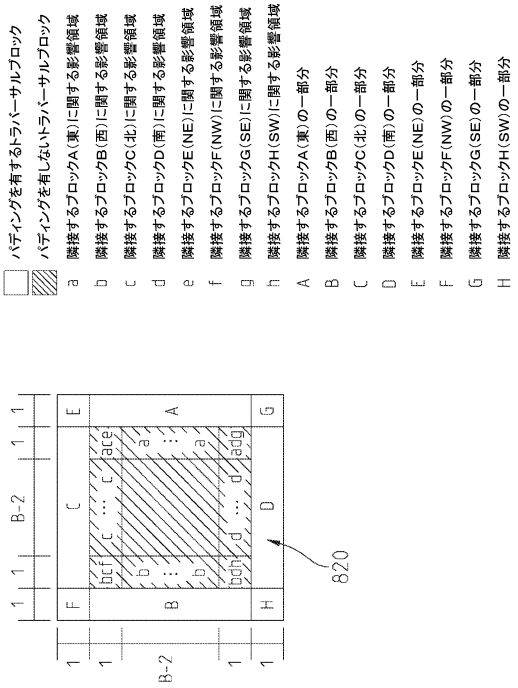
【図６】



【図７】


開始
現在のブロックの
トラバーサルマップをロード
710
現在のブロックに関する
コストマップをロード
720
ブロックのハッシュをゼロにする
730
前処理ハッシュを実行
740
終点をシード
745
スウィープを実行
750
後処理ハッシュを実行
760
前処理ハッシュと
後処理ハッシュとを比較
770
影響を受けた隣接するノードを識別
780
影響を受けた隣接するノードを、
隣接するノードのリストに追加
790
終了

【図８】


パディングを有するトラバーサルブロック
パディングを有しないトラバーサルブロック
a 隣接するブロックA(東)に関する影響領域
b 隣接するブロックB(西)に関する影響領域
c 隣接するブロックC(北)に関する影響領域
d 隣接するブロックD(南)に関する影響領域
e 隣接するブロックE(NE)に関する影響領域
f 隣接するブロックF(NW)に関する影響領域
g 隣接するブロックG(SE)に関する影響領域
h 隣接するブロックH(SW)に関する影響領域
A 隣接するブロックA(東)の一部分
B 隣接するブロックB(西)の一部分
C 隣接するブロックC(北)の一部分
D 隣接するブロックD(南)の一部分
E 隣接するブロックE(NE)の一部分
F 隣接するブロックF(NW)の一部分
G 隣接するブロックG(SE)の一部分
H 隣接するブロックH(SW)の一部分
820

【図９Ａ】

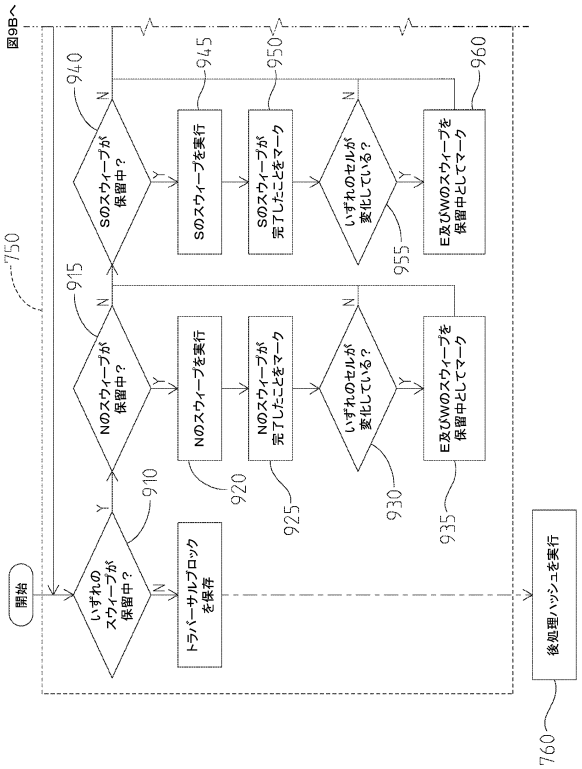


図9Bへ
開始
750
いずれの
スウィープが
保留中？
910
トラバーサルブロック
を保存
915
Nのスウィープが
保留中？
Nのスウィープを実行
920
Nのスウィープが
完了したことをマーク
925
いずれのセルが
変化している？
930
E及びWのスウィープを
保留中としてマーク
935
940
Sのスウィープが
保留中？
Sのスウィープを実行
945
Sのスウィープが
完了したことをマーク
950
いずれのセルが
変化している？
955
E及びWのスウィープを
保留中としてマーク
960
後処理ハッシュを実行
760

【図９Ｂ】

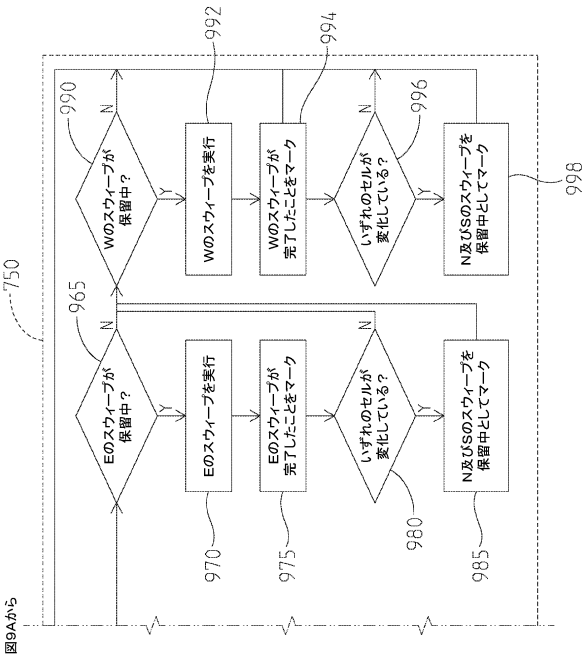
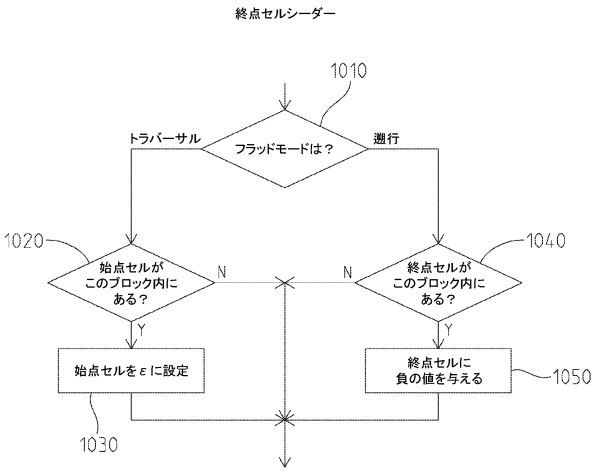
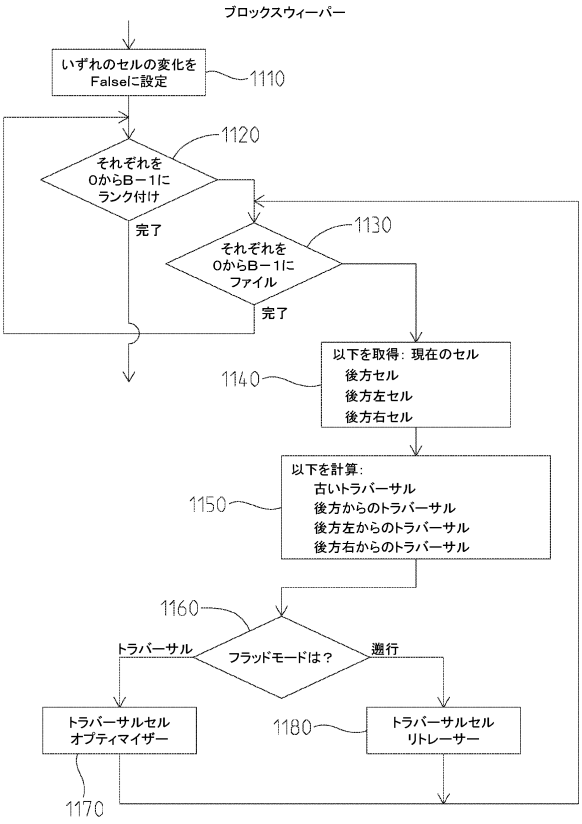


図9Aから
750
990
Wのスウィープが
保留中？
992
Wのスウィープを実行
994
Wのスウィープが
完了したことをマーク
996
いずれのセルが
変化している？
998
N及びSのスウィープを
保留中としてマーク
965
Eのスウィープが
保留中？
970
Eのスウィープを実行
975
Eのスウィープが
完了したことをマーク
980
いずれのセルが
変化している？
985
N及びSのスウィープを
保留中としてマーク

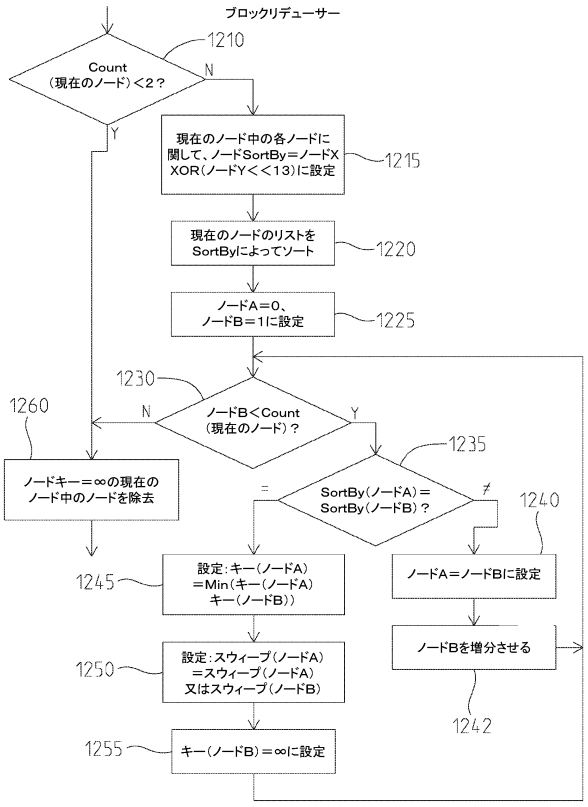
【図１０】



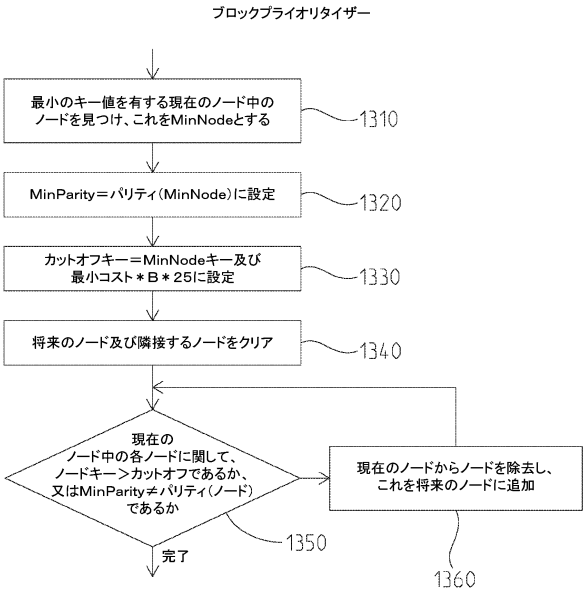
【図１１】



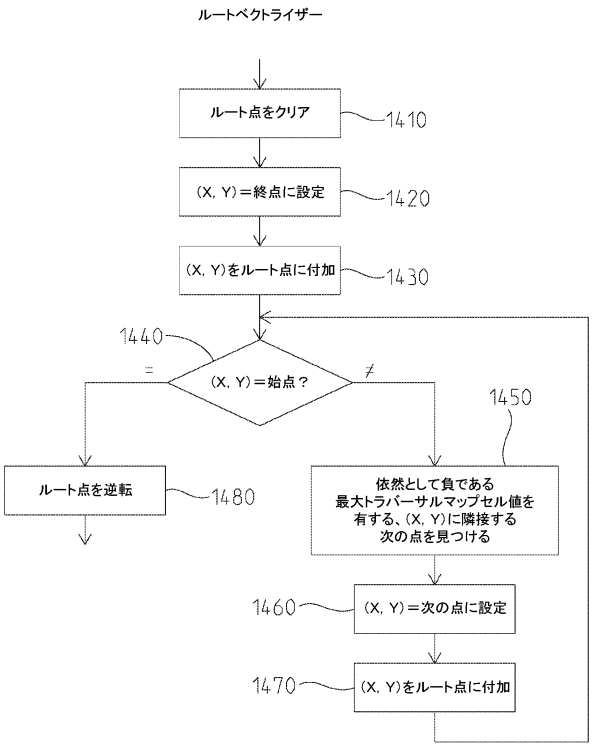
【図１２】



【図１３】



【図１４】



【図１５】

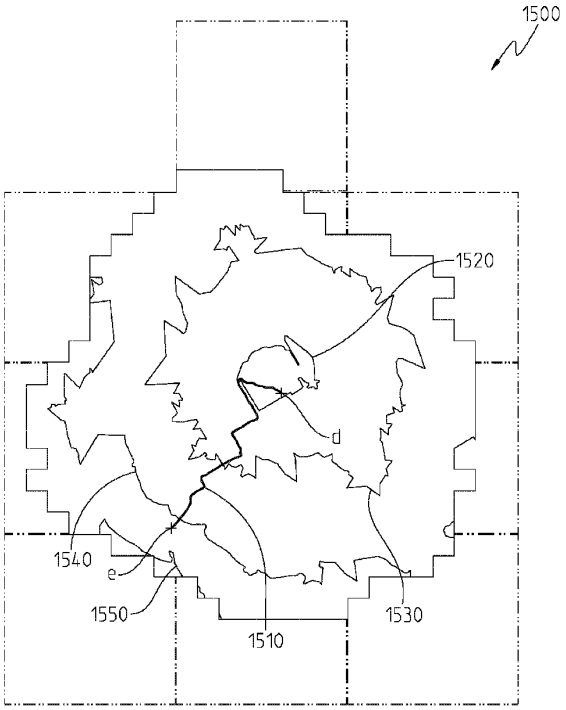


FIG. 15

【図１６】

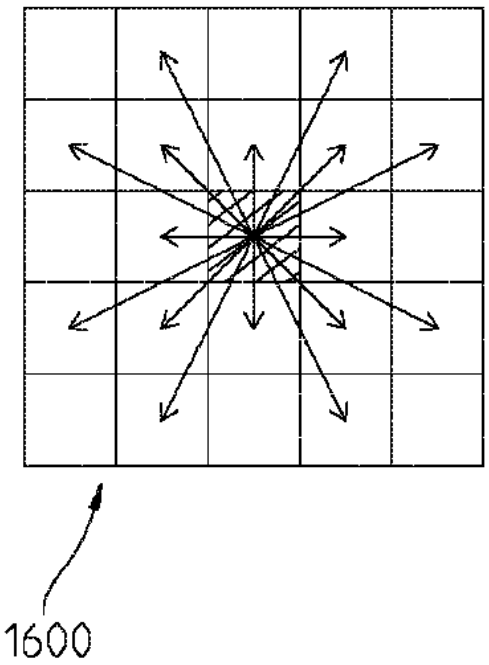


FIG. 16

フロントページの続き

審査官　　佐々木　佳祐
(56)参考文献　　国際公開第２０１６／０９２９４８（WO，Ａ１）
(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０１Ｃ　２１／００－２５／００
Ｇ０８Ｇ　　１／００－９９／００
Ｇ０９Ｂ　２９／００