(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7654356号
(P7654356)

(45)発行日　令和7年4月1日(2025.4.1)

(24)登録日　令和7年3月24日(2025.3.24)

(51)国際特許分類

		FI		
B65G	*1/00 (2006.01)*	B65G	1/00	501C
G05B	*19/418 (2006.01)*	G05B	19/418	Z

請求項の数　15　外国語出願　（全27頁）

(21)出願番号　特願2020-25066(P2020-25066)
(22)出願日　令和2年2月18日(2020.2.18)
(65)公開番号　特開2020-149675(P2020-149675A)
(43)公開日　令和2年9月17日(2020.9.17)
審査請求日　令和5年2月16日(2023.2.16)
(31)優先権主張番号　201921006550
(32)優先日　平成31年2月19日(2019.2.19)
(33)優先権主張国・地域又は機関
インド(IN)

(73)特許権者　510337621
タタ コンサルタンシー サービシズ リミテッド
TATA Consultancy Services Limited
インド国 マハーラシュトラ、ムンバイ ４０００２１、ナリマン ポイント、ナーマル ビルディング ９階
Nirmal Building, 9th Floor, Nariman Point, Mumbai 400021, Maharashtra, India.
(74)代理人　100094569
弁理士　田中 伸一郎
(74)代理人　100103610

最終頁に続く

(54)【発明の名称】　マルチロボット環境においてノンプリエンプティブタスクのスケジューリングを最適化するためのシステム及び方法

(57)【特許請求の範囲】

【請求項１】

マルチロボット環境においてノンプリエンプティブタスクのスケジューリングを最適化するための方法であって、

前記方法は、

ロボットのセットを介して、複数のタスクを定義するステップであって、前記複数のタスクの各々は、到着時間、実行時間、デッドライン、及び性能損失関数によって特徴付けられるノンプリエンプティブスケジューラブルタスクであり、前記複数のタスクの各々は、前記マルチロボット環境において実行され、前記性能損失関数は、

$$P(t_i) = \max(0, (c_i - d_i) \times p_i) \qquad \text{式（１）}$$

として定義され、c_iはタスクt_iについての完了時間を表し、t_iは前記複数のタスクに対応し、d_iはデッドラインを表し、p_iは性能損失値を表す、ステップ（２０１）と、

１又は２以上のハードウェアプロセッサによって、１又は２以上の予め割り当てられたタスクと、前記定義された複数のタスクから１又は２以上の新規に到着したタスクとをマージするステップであって、前記１又は２以上の予め割り当てられたタスクは、前記ロボットのセットの間で１又は２以上のロボットに既に割り当てられた非実行済みタスクであり、前記１又は２以上の新規に到着したタスクは、前記ロボットのセットの間で割り当てられることになる非実行済みタスクを含む、ステップ（２０２）と、

前記マルチロボット環境において前記１又は２以上のマージされたタスクのスケジューリングを最適化するステップ（２０３）と、
のプロセッサによって実施されるステップを含み、
前記最適化するステップは、
オンライン最小性能損失スケジューリング（ＯＭＰＬＳ）手法を実装することにより、前記ロボットのセットの間でタスクの第１のセットをスケジュールするステップであって、前記タスクの第１のセットは、前記１又は２以上のマージされたタスクの中でより高い性能損失値を有するタスクを含む、ステップと、
前記ＯＭＰＬＳ手法を介して、前記ロボットのセットの間でタスクの第２のセットをスケジュールするステップであって、前記タスクの第２のセットは、デッドライン内でスケジュールすることができかつ前記１又は２以上のマージされたタスクの中で低い性能損失値を有するタスクを含む、ステップと、
リストスケジューリング手法により、タスクの残りのサブセットの性能損失値を最小にするステップであって、前記タスクの残りのサブセットは、予め定義されたデッドライン内にスケジュールすることができないタスクを含む、ステップと、
を含む、
方法。

【請求項２】
前記タスクの残りのサブセットの前記性能損失値を最小にするステップは、前記リストスケジューリング手法を介して、前記ノンプリエンプティブタスクのスケジューリングを最適化するために前記新規に到着したタスクから１又は２以上の高優先度タスクを取り除くステップを含み、タスクの優先度は、単位時間遅延当たりにシステムに生じることになる前記性能損失値により判断され、前記タスクは前記優先度に基づいて分類され、最も高いペナルティ値（性能損失値＝１０）を有するタスクは、前記１又は２以上の高優先度タスクとして取り扱われる、請求項１に記載の方法。

【請求項３】
前記マージするステップは、前記１又は２以上の予め割り当てられたタスク及び前記１又は２以上の新規に到着したタスクに対応する非昇順の性能損失値に基づいて、前記１又は２以上の予め割り当てられたタスク及び前記１又は２以上の新規に到着したタスクをあるキューにおいてソートすることによって実行される、請求項１に記載の方法。

【請求項４】
前記取り除かれた１又は２以上の高優先度タスクは、前記ロボットのセットの間でアイドル状態のロボットに割り当てられ、前記アイドル状態のロボットは、前記複数のタスクのいずれも実行しないロボットである、請求項２に記載の方法。

【請求項５】
前記１又は２以上の高優先度タスクを取り除くステップは、前記複数のタスクの中の各タスクが前記ロボットのセットの間で少なくとも１つのロボットに割り当てられるまで、反復して実行される、請求項２に記載の方法。

【請求項６】
前記ＯＭＰＬＳ手法は、前記ノンプリエンプティブタスクのスケジューリングを最適化するため前記タスクの残りのサブセットの各々に対して優先度更新を実行するステップを含む、請求項１に記載の方法。

【請求項７】
前記優先度更新は、前記タスクの残りのサブセットの各々の前記実行時間が初期の予め定められた閾値よりも大きいと判断されると、前記タスクの残りのサブセットの各々の優先度関数をインクリメントすることによって実行され、前記初期の予め定められた閾値は、予め定められたデッドラインを用いて識別される、請求項６に記載の方法。

【請求項８】
マルチロボット環境においてノンプリエンプティブタスクのスケジューリングを最適化するためのシステム（１００）であって、

命令を記憶するメモリ（１０２）と、
１又は２以上の通信インタフェース（１０６）と、
前記１又は２以上の通信インタフェース（１０６）を介して前記メモリ（１０２）に結合された１又は２以上のハードウェアプロセッサ（１０４）と、
を含み、
前記１又は２以上のハードウェアプロセッサ（１０４）は、前記命令によって
ロボットのセットを介して、複数のタスクを定義することであって、前記複数のタスクの各々は、到着時間、実行時間、デッドライン、及び性能損失関数によって特徴付けられるノンプリエンプティブスケジューラブルタスクであり、前記複数のタスクの各々は、前記マルチロボット環境において実行され、前記性能損失関数は、

$$P(t_i) = \max(0, (c_i - d_i) \times p_i) \qquad \text{式（１）}$$

として定義され、c_iはタスクt_iについての完了時間を表し、t_iは前記複数のタスクに対応し、d_iはデッドラインを表し、p_iは性能損失値を表す、ことと、
１又は２以上の予め割り当てられたタスクと、前記定義された複数のタスクから１又は２以上の新規に到着したタスクとをマージすることであって、前記１又は２以上の予め割り当てられたタスクは、前記ロボットのセットの間で１又は２以上のロボットに既に割り当てられた非実行済みタスクであり、前記１又は２以上の新規に到着したタスクは、前記ロボットのセットの間で割り当てられることになる非実行済みタスクを含む、ことと、
前記マルチロボット環境において前記１又は２以上のマージされたタスクのスケジューリングを最適化することと、
を行うように構成され、
前記最適化することは、
オンライン最小性能損失スケジューリング（ＯＭＰＬＳ）手法を実装することにより、前記ロボットのセットの間でタスクの第１のセットをスケジュールすることであって、前記タスクの第１のセットは、前記１又は２以上のマージされたタスクの中でより高い性能損失値を有するタスクを含む、ことと、
前記ＯＭＰＬＳ手法を介して、前記ロボットのセットの間でタスクの第２のセットをスケジュールすることであって、前記タスクの第２のセットは、デッドライン内でスケジュールすることができかつ前記１又は２以上のマージされたタスクの中で低い性能損失値を有するタスクを含む、ことと、
リストスケジューリング手法により、タスクの残りのサブセットの性能損失値を最小にすることであって、前記タスクの残りのサブセットは、予め定義されたデッドライン内にスケジュールすることができないタスクを含む、ことと、
を含む、
システム（１００）。
【請求項９】
前記１又は２以上のハードウェアプロセッサ（１０４）は、前記リストスケジューリング手法を介して、前記ノンプリエンプティブタスクのスケジューリングを最適化するために前記新規に到着したタスクから１又は２以上の高優先度タスクを取り除くことによって、前記タスクの残りのサブセットの前記性能損失値を最小にするよう構成され、タスクの優先度は、単位時間遅延当たりにシステムに生じることになる前記性能損失値により判断され、前記タスクは前記優先度に基づいて分類され、最も高いペナルティ値（性能損失値＝１０）を有するタスクは、前記１又は２以上の高優先度タスクとして取り扱われる、請求項８に記載のシステム（１００）。
【請求項１０】
前記１又は２以上のハードウェアプロセッサ（１０４）は、前記１又は２以上の予め割り当てられたタスク及び前記１又は２以上の新規に到着したタスクに対応する非昇順の性能損失値に基づいて、前記１又は２以上の予め割り当てられたタスク及び前記１又は２以

上の新規に到着したタスクをあるキューにおいてマージするように構成される、請求項８に記載のシステム（１００）。

【請求項１１】

前記１又は２以上のハードウェアプロセッサ（１０４）は、前記取り除かれた１又は２以上の高優先度タスクを前記ロボットのセットの間でアイドル状態のロボットに割り当てるように構成され、前記アイドル状態のロボットは、前記複数のタスクのいずれも実行しないロボットである、請求項９に記載のシステム（１００）。

【請求項１２】

前記１又は２以上のハードウェアプロセッサ（１０４）は、前記複数のタスクの中の各タスクが前記ロボットのセットの間で少なくとも１つのロボットに割り当てられるまで、前記１又は２以上の高優先度タスクを取り除くステップを反復して実行するように構成される、請求項９に記載のシステム（１００）。

【請求項１３】

前記１又は２以上のハードウェアプロセッサ（１０４）は、前記ノンプリエンプティブタスクのスケジューリングを最適化するため前記タスクの残りのサブセットの各々に対して優先度更新を実行することにより、前記ＯＭＰＬＳ手法を実装するように構成される、請求項８に記載のシステム（１００）。

【請求項１４】

前記１又は２以上のハードウェアプロセッサ（１０４）は、前記タスクの残りのサブセットの各々の前記実行時間が初期の予め定められた閾値よりも大きいと判断されると、前記タスクの残りのサブセットの各々の優先度関数をインクリメントすることによって前記優先度更新を実行するように構成され、前記初期の予め定められた閾値は、予め定められたデッドラインを用いて識別される、請求項１３に記載のシステム（１００）。

【請求項１５】

１又は２以上の命令を含む１又は２以上の非一時的機械可読情報記憶媒体であって、前記命令は、１又は２以上のハードウェアプロセッサによって実行されたときに、

ロボットのセットを介して、複数のタスクを定義することであって、前記複数のタスクの各々は、到着時間、実行時間、デッドライン、及び性能損失関数によって特徴付けられるノンプリエンプティブスケジューラブルタスクであり、前記複数のタスクの各々は、マルチロボット環境において実行され、前記性能損失関数は、

$$P(t_i) = \max(0, (c_i - d_i) \times p_i) \qquad \text{式（１）}$$

として定義され、c_iはタスクt_iについての完了時間を表し、t_iは前記複数のタスクに対応し、d_iはデッドラインを表し、p_iは性能損失値を表す、こと（２０１）と、

１又は２以上のハードウェアプロセッサによって、１又は２以上の予め割り当てられたタスクと、前記定義された複数のタスクから１又は２以上の新規に到着したタスクとをマージすることであって、前記１又は２以上の予め割り当てられたタスクは、前記ロボットのセットの間で１又は２以上のロボットに既に割り当てられた非実行済みタスクであり、前記１又は２以上の新規に到着したタスクは、前記ロボットのセットの間で割り当てられることになる非実行済みタスクを含む、こと（２０２）と、

前記マルチロボット環境において前記１又は２以上のマージされたタスクのスケジューリングを最適化すること（２０３）と、

を行わせ、

前記最適化することは、

オンライン最小性能損失スケジューリング（ＯＭＰＬＳ）手法を実装することにより、前記ロボットのセットの間でタスクの第１のセットをスケジュールすることであって、前記タスクの第１のセットは、前記１又は２以上のマージされたタスクの中でより高い性能損失値を有するタスクを含む、ことと、

前記ＯＭＰＬＳ手法を介して、前記ロボットのセットの間でタスクの第２のセットを

スケジュールすることであって、前記タスクの第２のセットは、デッドライン内でスケジュールすることができかつ前記１又は２以上のマージされたタスクの中で低い性能損失値を有するタスクを含む、ことと、

リストスケジューリング手法により、タスクの残りのサブセットの性能損失値を最小にすることであって、前記タスクの残りのサブセットは、予め定義されたデッドライン内にスケジュールすることができないタスクを含む、ことと、

を含む、

１又は２以上の非一時的機械可読情報記憶媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

（関連出願の相互参照及び優先権）

本出願は、２０１９年２月１９日に出願されたインド国特許出願２０１９２１００６５５０に対する優先権を主張する。

【０００２】

（技術分野）

本明細書の開示は、全体的に、ロボットプロセスの自動化に関し、より詳細には、マルチロボット環境におけるノンプリエンプティブタスク(non-preemptive tasks)のスケジューリングの最適化に関する。

【背景技術】

【０００３】

デジタル及びオートメーション時代においては、マルチロボットシステム及び関連する技術が様々な用途及び産業で広く用いられている。例えば、倉庫調達システムでは、多種多様な対象物を倉庫内に保管し、またそこから回収するプロセスを自動化するためにマルチロボットシステムが配備されている。効率的な倉庫管理は、注文の物品をそれぞれの顧客に滑らかに配送することを保証し、現代のビジネスに大きな影響を与える。販売業者及び電子小売業者は、一般に、あらゆる物品の配送について到着予定時刻（ＥＴＡ）を提供し、付加価値サービスとして迅速且つ時間に制約のある配送を約束している。

【０００４】

物品又は対象物の倉庫内外への移動は、タスクのスケジューリング（物品に対する移動に関連した）に関して大きな課題である。膨大な量の需要がリアルタイムで倉庫調達システムに流入することで、最適なスケジューリングに対する要求がある。電子商取引及び最新のロジスティクスの急速な発展により、倉庫システムは、在庫の規模が非常に大きくなり、短いピッキング時間に対して幅広い需要があり、これによってロボットシステムの配備が求められている。

【０００５】

このように、産業及び他の技術分野における用途に起因して、マルチロボットシステムにおけるタスク割当て(task allocation)は、著しく重要性を増している。所与のシステムレベルのタスクのサブタスクに個々のロボットを割り当てるプロセスは、タスク割当てと呼ばれ、何れのマルチロボットシステムにも必要とされる重要な機能性を含む。タスク割当ての目的は、マルチロボットシステムの性能を向上させることになるように１又は複数のロボットタスクを割り当てることであり、これは、全体の実行時間の短縮を含む。一般に、タスク割当ては、未知の動的環境で機能しているマルチロボットシステムにとって必須の要件である。タスク割当ては、ロボットがシステム性能全体を改善するために、環境の変化又は他のロボットの動作に応答してロボットの挙動を変化させることが可能となる。

【０００６】

従って、マルチロボットシステムにおけるタスク割当ては、特に、異なる要件及び制約を伴う様々なタスクを最適な方式で実施するのに必要とされる異なる能力をロボットが備えたかかるロボット環境において、複雑な課題を提起する。タスク割当ての問題は、環境の変化を含む現象を伴う時間的に変化する動的決定問題であるので、問題は、指定された

時間間隔で反復して解決されなければならない。従って、タスク割当ての問題への取り組みはより複雑になる。検討中の特定の技術分野の要件が、マルチロボットタスク割当て問題の特徴及び複雑さに更に影響を及ぼしている。

【発明の概要】

【課題を解決するための手段】

【０００７】

本開示の実施形態は、本発明者らによって従来のシステムにおいて認識された上述の技術的問題の１又は２以上への解決策として技術改良を提示する。例えば、１つの実施形態において、マルチロボット環境においてノンプリエンプティブタスクのスケジューリング(scheduling)を最適化するための方法であって、本方法は、

ロボットのセットを介して、複数のタスクを定義するステップであって、複数のタスクの各々は、到着時間(arrival time)、実行時間(execution time)、デッドライン、及び時間単位(time unit)当たりの性能損失関数(performance loss function)によって特徴付けられるノンプリエンプティブスケジューラブルタスク(non-preemptive schedulable task)であり、複数のタスクの各々は、マルチロボット環境において実行される、ステップと、

１又は２以上のハードウェアプロセッサによって、１又は２以上の予め割り当てられたタスクと、定義された複数のタスクから１又は２以上の新規に到着したタスク(newly arriving tasks)とをマージするステップ(merging)であって、１又は２以上の予め割り当てられたタスクは、ロボットのセットの間で１又は２以上のロボットに既に割り当てられた非実行済みタスク(non-executed task)であり、１又は２以上の新規に到着したタスクは、ロボットのセットの間で割り当てられることになる非実行済みタスクを含む、ステップと、

マルチロボット環境においてマージされたタスクのスケジューリングを最適化するステップと、

を含み、最適化するステップは、

オンライン最小性能損失スケジューリング(Online Minimum Performance Loss Scheduling)（ＯＭＰＬＳ）手法(technique)を実装すること(implementing)により、ロボットのセットの間でタスクの第１のセットをスケジュールするステップ(scheduling)であって、タスクの第１のセットが、マージされたタスクの中でより高い性能損失値を有するタスクを含み、タスクの第１のセットは更に、待ち時間(waiting time)の閾値を上回り且つより高い性能損失値を有するタスクよりも優先度が低い少なくとも１つのタスクを含む、ステップと、

ＯＭＰＬＳ手法を介して、ロボットのセットの間でタスクの第２のセットをスケジュールするステップであって、タスクの第２のセットはタスクを含み、該タスクはそれらのデッドライン内でスケジュールすることができ且つマージされたタスクの中で低い性能損失値を有する、ステップと、

リストスケジューリング手法(list-scheduling technique)により、タスクの残りのサブセットの性能損失値を最小にするステップであって、タスクの残りのサブセットが、予め定義されたデッドライン内にスケジュールすることができないタスクを含む、ステップと、

リストスケジューリング手法を介して、ノンプリエンプティブタスクのスケジューリングを最適化するために新規に到着したタスクから１又は２以上の高優先度タスクを取り除くステップ(eliminating)と、

複数のタスクの中の各タスクが少なくとも１つのロボットに割り当てられるまで１又は２以上の高優先度タスクを取り除くステップを反復して実施するステップと、

ノンプリエンプティブタスクのスケジューリングを最適化するためタスクの残りのサブセットの各々に対して優先度更新(priority update)を実行するステップと、

を含み、タスクの残りのサブセットの各々の実行時間が初期の予め定められた閾値(initial pre-defined threshold)よりも大きいと判断されると、タスクの残りのサブセットの

各々の優先度関数をインクリメントすることによって実行され、初期の予め定められた閾値は、予め定められたデッドラインを用いて識別される、
方法が提供される。
【０００８】
別の態様において、マルチロボット環境においてノンプリエンプティブタスクのスケジューリングを最適化するシステムであって、
命令を記憶するメモリと、
１又は複数の通信インタフェースと、
１又は複数の通信インタフェースを介してメモリに結合された１又は２以上のハードウェアプロセッサと、
を含み、１又は２以上のハードウェアプロセッサは、命令によって、
ロボットのセットを介して、複数のタスクを定義することであって、複数のタスクの各々は、到着時間、実行時間、及び時間単位当たりの性能損失関数によって特徴付けられるノンプリエンプティブスケジューラブルタスクであり、複数のタスクの各々は、マルチロボット環境において実行される、ことと、
１又は２以上のハードウェアプロセッサによって、１又は２以上の予め割り当てられたタスクと、定義された複数のタスクから１又は２以上の新規に到着したタスクとをマージすることであって、１又は２以上の予め割り当てられたタスクは、ロボットのセットの間で１又は２以上のロボットに既に割り当てられた非実行済みタスクであり、１又は２以上の新規に到着したタスクは、ロボットのセットの間で割り当てられることになる非実行済みタスクを含む、ことと、
マルチロボット環境においてマージされたタスクのスケジューリングを最適化することと、
を含み、最適化することは、
オンライン最小性能損失スケジューリング（ＯＭＰＬＳ）手法を実装することにより、ロボットのセットの間でタスクの第１のセットをスケジュールすることであって、タスクの第１のセットが、マージされたタスクの中でより高い性能損失値を有するタスクを含み、タスクの第１のセットは更に、待ち時間の閾値を上回り且つより高い性能損失値を有するタスクよりも優先度が低い少なくとも１つのタスクを含む、ことと、
ＯＭＰＬＳ手法を介して、ロボットのセットの間でタスクの第２のセットをスケジュールすることであって、タスクの第２のセットは、デッドライン内でスケジュールすることができ且つマージされたタスクの中で低い性能損失値を有するタスクを含む、ことと、
リストスケジューリング手法により、タスクの残りのサブセットの性能損失値を最小にすることであって、タスクの残りのサブセットが、予め定義されたデッドライン内にスケジュールすることができないタスクを含む、ことと、
リストスケジューリング手法を介して、ノンプリエンプティブタスクのスケジューリングを最適化するために新規に到着したタスクから１又は２以上の高優先度タスクを取り除くことと、
取り除かれた１又は２以上の高優先度タスクをロボットのセットの間でアイドル状態のロボット(idle robot)に割り当てることであって、アイドル状態のロボットは、複数のタスクの何れかを実行しないロボットである、ことと、
複数のタスクの中の各タスクが少なくとも１つのロボットに割り当てられるまで、１又は２以上の高優先度タスクを取り除くことを反復して実施することと、
ノンプリエンプティブタスクのスケジューリングを最適化するためタスクの残りのサブセットの各々に対して優先度更新を実行することにより、ＯＭＰＬＳ手法を実装する(implement)ことと、
タスクの残りのサブセットの各々の実行時間が初期の予め定められた閾値よりも大きいと判断されると、タスクの残りのサブセットの各々の優先度関数をインクリメントすることによって優先度更新を実行することであって、初期の予め定められた閾値は、予め定められたデッドラインを用いて識別される、ことと、

を含む、システムが提供される。
【０００９】
更に別の態様において、１又は２以上の命令を含む１又は２以上の非一時的機械可読情報記憶媒体であって、命令が、１又は２以上のハードウェアプロセッサによって実行されたときに、１又は２以上のハードウェアプロセッサが、マルチロボット環境においてノンプリエンプティブタスクのスケジューリングを最適化する方法を実施させ、
前記方法は、
ロボットのセットを介して、複数のタスクを定義するステップであって、複数のタスクの各々は、到着時間、実行時間、デッドライン、及び時間単位当たりの性能損失関数によって特徴付けられるノンプリエンプティブスケジューラブルタスクであり、複数のタスクの各々は、マルチロボット環境において実行される、ステップと、
１又は２以上の予め割り当てられたタスクと、定義された複数のタスクから１又は２以上の新規に到着したタスクとをマージするステップであって、１又は２以上の予め割り当てられたタスクは、ロボットのセットの間で１又は２以上のロボットに既に割り当てられた非実行済みタスクであり、１又は２以上の新規に到着したタスクは、ロボットのセットの間で割り当てられることになる非実行済みタスクを含む、ステップと、
マルチロボット環境においてマージされたタスクのスケジューリングを最適化するステップと、
を含み、最適化するステップは、
オンライン最小性能損失スケジューリング（ＯＭＰＬＳ）手法を実装することにより、ロボットのセットの間でタスクの第１のセットをスケジュールするステップであって、タスクの第１のセットが、マージされたタスクの中でより高い性能損失値を有するタスクを含み、タスクの第１のセットは更に、待ち時間の閾値を上回り且つより高い性能損失値を有するタスクよりも優先度が低い少なくとも１つのタスクを含む、ステップと、
ＯＭＰＬＳ手法を介して、ロボットのセットの間でタスクの第２のセットをスケジュールするステップであって、タスクの第２のセットは、デッドライン内でスケジュールすることができ且つマージされたタスクの中で低い性能損失値を有するタスクを含む、ステップと、
リストスケジューリング手法により、タスクの残りのサブセットの性能損失値を最小にするステップであって、タスクの残りのサブセットが、予め定義されたデッドライン内にスケジュールすることができないタスクを含む、ステップと、
リストスケジューリング手法を介して、ノンプリエンプティブタスクのスケジューリングを最適化するために新規に到着したタスクから１又は２以上の高優先度タスクを取り除くステップと、
複数のタスクの中の各タスクが少なくとも１つのロボットに割り当てられるまで１又は２以上の高優先度タスクを取り除くステップを反復して実施するステップと、
ノンプリエンプティブタスクのスケジューリングを最適化するためタスクの残りのサブセットの各々に対して優先度更新を実行するステップと、
を含み、タスクの残りのサブセットの各々の実行時間が初期の予め定められた閾値よりも大きいと判断されると、タスクの残りのサブセットの各々の優先度関数をインクリメントすることによって実行され、初期の予め定められた閾値は、予め定められたデッドラインを用いて識別される、
非一時的機械可読情報記憶媒体が提供される。
【００１０】
前述の概要及び以下の詳細な説明は何れも例証で説明のためのものであり、請求項に記載された本発明の実施形態の更なる説明を行うことを意図している点を理解されたい。
【００１１】
本開示に組み込まれ且つその一部を構成する添付図面は、例示的な実施形態を例証しており、本明細書と共に本開示の原理を説明する役割を果たす。
【図面の簡単な説明】

【００１２】
【図１】本開示の一部の実施形態による、マルチロボット環境におけるノンプリエンプティブタスクのスケジューリングを最適化するためのシステムのブロック図である。
【図２】本開示の一部の実施形態による、マルチロボット環境におけるノンプリエンプティブタスクのスケジューリングを最適化するプロセスに関するステップを示したフロー図である。
【図３】本開示の一部の実施形態による、マルチロボット環境において高トラフィック期間中に到着したタスクについて累積されたデッドラインをミスしたタスク及びペナルティの数をグラフで示している。
【図４】本開示の一部の実施形態による、マルチロボット環境において高トラフィック期間中に到着したタスクについて累積されたデッドラインをミスしたタスク及びペナルティの数をグラフで示している。
【図５】本開示の一部の実施形態による、時間τ０＋２５にて達したタスクについて被った配分スロット及び性能損失をグラフで示している。
【図６】本開示の一部の実施形態による、従来のシステム及び方法と本開示の方法とを実装することによる、定常的な高トラフィックの線形的に増大するロボット、［１，１０］間で分散されるペナルティ値、及び３つの異なる期限パターンについての性能比較をグラフで示している。
【発明を実施するための形態】
【００１３】
例示的な実施形態について、添付図面を参照して記載される。図中、参照番号の最上位桁（単数又は複数）は、当該参照番号が最初に現われる図を表している。好都合であれば、複数の図面を通じて同じ又は同様の部分を指すために同じ参照番号が用いられる。開示される原理の実施例及び特徴が本明細書に記載されるが、開示される実施形態の技術的思想及び範囲から逸脱することなく、変更、適応、及び他の実施構成が実施可能である。以下の詳細な記載は、例証に過ぎず、真の範囲及び思想は添付の特許請求の範囲によって示されることが意図される。
【００１４】
本開示の実施形態は、マルチロボット環境におけるノンプリエンプティブタスクのスケジューリングを最適化するためのシステム及び方法を提供する。一般に、マルチロボットタスク割当ては、タスクスケジューリング又はタスク割当てを最適化するためにタスクをロボットの間で効率的に割り当てるステップを含む。例えば、顧客が対象物を発注するときに対象物が発送されるようにスケジュールされる、倉庫におけるタスク割当ては、対象物を保管ラックから梱包台へ引き出すステップを含む。
【００１５】
スマート環境、特にスマート倉庫環境においてノンプリエンプティブ又はアトミックタスクを均質で自律的なロボットに対してスケジューリングする問題は、均質なマルチプロセッサシステム上でのノンプリエンプティブスケジューリング又は散発的プロセススケジューリングに極めて類似する。目標は、ノンプリエンプティブタスクのうちの各々に対応するデッドライン内にノンプリエンプティブの最大数を最適にスケジューリングすることである。更に、ノンプリエンプティブタスクの各々は、デッドラインと、ノンプリエンプティブタスクのセットのうちの何れかがその対応するデッドライン内に完了されていない場合にマルチロボットシステムが被る可能性がある、これに起因する性能損失と、を有するので、タスクスケジューリングシステムの目標は、デッドラインミスの数の削減及び性能損失の最小化である。
【００１６】
特にロボット及び関連の技術を実装する倉庫において、従来のシステム及び方法は、ノンプリエンプティブタスクの各々に関連付けられたデッドライン及び性能損失を考慮にいれておらず、従って、かかるノンプリエンプティブタスクが必然的にデッドラインミスする場合、ノンプリエンプティブタスクのうち多くのタスクを破棄している。しかしながら

、一般には、対応するデッドラインをミスしたとしても、スマート倉庫環境における各タスクは完了される必要がある。ノンプリエンプティブタスクをマージするステップ及びこれらをマルチプロセッサシステム（又はマルチコアシステム）上にスケジューリングするステップは、主要なタスクスケジューリング問題を含む。

【００１７】

開示される方法は、従来のシステム及び方法が直面した課題の克服に取り組んでいる。例えば、開示される方法は、ノンプリエンプティブタスクの各々をスケジューリングし、これにより低い性能損失値を有する優先度の低いタスク(low-priority task)でもそのスケジューリングを可能にするスケジューリング手法に言及している。更に、開示される方法は、マルチロボット環境においてタスクを配分しながら、ノンプリエンプティブタスクの各々に関連付けられるデッドライン及び性能損失を考慮し、これによりマルチロボット環境でのスケジューリングの最適化及び性能損失全体の最小化を提供する。

【００１８】

図１は、本開示の実施形態による、マルチロボット環境におけるノンプリエンプティブタスクのスケジューリングを最適化するためのシステム１００の例示的なブロック図を示す。１つの実施形態において、システム１００は、１又は２以上のプロセッサ１０４、１又は複数の通信インタフェースデバイス又は１又は複数の入力／出力（Ｉ／Ｏ）インタフェース１０６、及び１又は２以上のプロセッサ１０４に動作可能に結合された１又は２以上のデータ記憶装置又はメモリ１０２を含む。ハードウェアプロセッサである１又は２以上のプロセッサ１０４は、１又は２以上のマイクロプロセッサ、マイクロコンピュータ、マイクロコントローラ、デジタル信号プロセッサ、中央処理装置、状態機械、論理回路、及び／又は動作命令に基づいて信号を操作する任意のデバイスとして実装する(implemented)ことができる。他の能力の中でも、１又は複数のプロセッサは、メモリ１０２に記憶されたコンピュータ可読命令をフェッチして実行するように構成される。１つの実施形態において、システム１００は、例えば、ラップトップコンピュータ、ノートブック、ハンドヘルドデバイス、ワークステーション、メインフレームコンピュータ、サーバ、ネットワーククラウド及び同様のものなど、様々なコンピューティングシステムに実装することができる。

【００１９】

メモリ１０２は、例えば、スタティックランダムアクセスメモリ（ＳＲＡＭ）及びダイナミックランダムアクセスメモリ（ＤＲＡＭ）のような揮発性メモリ、及び／又はリードオンリメモリ（ＲＯＭ）、消去可能プログラマブルＲＯＭ、フラッシュメモリ、ハードディスク、光ディスク、並びに磁気テープのような不揮発性メモリを含めて、当技術分野で公知の任意のコンピュータ可読媒体を含むことができる。１つの実施形態において、メモリ１０２は、マルチロボット環境におけるノンプリエンプティブタスクのスケジューリングを最適化することに関連付けられるあらゆるデータを記憶するよう構成することができる。１つの実施形態において、ロボット、マージされたタスク、スケジューリングの最適化、オンライン最小性能損失スケジューリング（ＯＭＰＬＳ）手法、性能損失最小化、その他に関する情報は、メモリ１０２に記憶される。更に、インタラクティブなユーザクエリに対する応答の生成に関する全ての情報（入力、出力など）は、参照の目的で、履歴データとしてマルチロボット環境におけるノンプリエンプティブタスクのスケジューリングを最適化することに基づいている。

【００２０】

１又は複数のＩ／Ｏインタフェースデバイス１０６は、様々なソフトウェア及びハードウェアインタフェース、例えば、ウェブインタフェース、グラフィカルユーザインタフェース、及び同様のものを含むことができ、有線ネットワーク（例えば、ＬＡＮ、ケーブルなど）及びＷＬＡＮ、セルラ、又は衛星のような無線ネットワークを含めて、多種多様なネットワークＮ／Ｗ及びプロトコルタイプ内の複数の通信を容易にすることができる。１つの実施形態において、１又は複数のＩ／Ｏインタフェースデバイスは、幾つかのデバイスを互いに又は別のサーバに接続するための１又は２以上のポートを含むことができる。

【００２１】
ここで、図面全体を通じて同じ参照符号が一貫して対応する特徴を示す図面、より詳細には図１～図６を参照すると、好ましい実施形態が示され、これらの実施形態は、以下の例示的なシステム及び／又は方法との関連において記載されている。
【００２２】
図２は、図１を参照して、本開示の幾つかの実施形態による、マルチロボット環境におけるノンプリエンプティブタスクのスケジューリングを最適化するための方法の例示的なフロー図を示す。１つの実施形態において、システム１００は、１又は２以上のハードウェアプロセッサ１０４に動作可能に結合されたメモリ１０２の１又は２以上のデータ記憶装置を含み、１又は２以上のプロセッサ１０４による方法のステップの実行のための命令を記憶するように構成される。次に、本開示の方法のステップについて、図１に示されるシステム１００の構成要素及びフロー図を参照して説明する。本開示の実施形態において、ハードウェアプロセッサ１０４は、命令が構成されたときに、本明細書に記載される１又は２以上の方法を実施する。
【００２３】
本開示の１つの実施形態によれば、ステップ２０１では、１又は２以上のハードウェアプロセッサ１０４は、ロボットのセットを介して、複数のタスクを定義するよう構成され、複数のタスクの各々は、到着時間、実行時間（又はタスク実行時間）、デッドライン、ノンプリエンプティブタスクを完了するための指定された時間間隔、及び時間単位当たりの性能損失関数によって特徴付けられるノンプリエンプティブスケジューラブルタスクである。本明細書で使用される場合の用語「性能損失(performance loss)」は、倉庫タスクスケジューリングにて一般的に使用される用語である用語「ペナルティ」と同様と理解することができ、タスクスケジューリングは、一般に、対応する「デッドライン」及び「ペナルティ」を含む。
【００２４】
更に、提案される方法は、倉庫システムを参照して説明しているが、１又は複数の倉庫システムのみに限定されると理解される訳ではない。提案される方法は、スケジューリング又はタスクスケジューリングがロボット又は関連の技術を用いて実施される、全てのかかる環境／応用／技術又は関連するシナリオ(scenario)に実装又は拡張することができる。複数のタスクを定義するステップは、以下の仮定のセットを考慮して検討することができる。
【００２５】
（仮定のセット）
（ｉ）タスクが到着すると、倉庫管理システムは、それぞれの物品の保管場所に基づいてタスクの実行時間を計算し、顧客に約束した全体の期待到着時間に基づいてデッドラインを割り当てて、顧客プロファイル及び対象物のタイプに基づいて性能損失を割り当てると仮定する；
（ｉｉ）任意の瞬間時間において、マルチロボット環境において利用可能であり、ｍ個の均質なロボットによって完了することが必要なｎ個のタスクが存在する。各ロボットは、タスクを完了するのに十分な利用可能なエネルギーを有する；
（ｉｉｉ）顧客が複数の物品を注文している場合、各物品のピックアップは、別個のタスクとして取り扱われる。この場合、各タスクは、異なる実行時間、デッドライン、及び性能損失値を有することができる；
（ｉｖ）簡単にするために、作動時間は、全てのタスクについて同じであり、タスク実行時間には含まれないと仮定することができる。従って、タスク実行時間のみが、ロボットの往復の移動時間を構成する；
（ｖ）顧客が複数の物品を注文している場合、ピックアップされる各物品は、別個のタスクとして取り扱われる。このシナリオでは、複数のタスクの各々は、異なる実行時間を有することができる。しかしながら、これらのタスクは、同じデッドライン及び同じ性能損失値に関連付けられる；

（ｖｉ）全てのタスクは、アトミックで且つノンプリエンプティブタスクである。ロボットは、一旦開始されると完了するまでタスクを実行しなければならない。ジョブは、２以上のロボットによって一斉に実行することはできない。タスクは独立しており、すなわち、１つのタスクの実行は、他の何れかのタスクの実行に左右されないと仮定することができる。

上記で検討したように、タスクを複数のロボットに配分する技術的問題は、（図示されていない）マルチプロセッサシステム上のタスクスケジューリングにマッピングすることができる。開示される方法は、オンザフライで均質なロボットのチームにアトミックタスクを割り当てることを容易にし、タスクがデッドラインを満たすことができないことによって、システムに最小限のペナルティが科せられるようになる。タスクは、非クレボヤント(non-clairvoyant)であり、各タスク（T_i）は、到着時間（a_i）、実行時間（e_i）、デッドライン（d_i）、及び性能損失値（p_i）によって特徴付けられる。タスクがデッドラインをミスして、時間（ｃｉ）で実行を完了した場合、（ｃｉ－ｄｉ）×ｐｉ単位の性能損失を生じる。

【００２６】

本開示の１つの実施形態によれば、ステップ２０２において、１又は２以上のハードウェアプロセッサ１０４は、１又は２以上の予め割り当てられたタスクと、定義された複数のタスクから１又は２以上の新規に到着したタスクとをマージするよう構成され、１又は２以上の予め割り当てられたタスクは、ロボットのセットの間で１又は２以上のロボットに既に割り当てられた非実行済みタスクであり、１又は２以上の新規に到着したタスクは、ロボットのセットの間で割り当てられることになる非実行済みタスクを含む。

【００２７】

１つの実施形態において、１又は２以上のハードウェアプロセッサ１０４は、１又は２以上の予め割り当てられたタスク及び１又は２以上の新規に到着したタスクに対応する非昇順(non-increasing order)の性能損失値に基づいて、１又は２以上の予め割り当てられたタスク及び１又は２以上の新規に到着したタスクをあるキューにおいてソートすることによってマージを実施するよう構成される。例示的なシナリオを考慮すると、性能損失値に基づいてタスクがソートされると、リストは、（３，２，４，５，１）として生成することができ、最初にスケジュールタスクｉｄ３をスケジュールすることができ、次いでｉｄ２、次にｉｄ４及び同様のものとすることができる。

【００２８】

本開示の１つの実施形態によれば、ステップ２０３において、１又は２以上のハードウェアプロセッサ１０４は、マルチロボット環境においてマージされたタスクのスケジューリングを最適化するように構成される。ステップ２０３において、１又は２以上のハードウェアプロセッサ１０４は、最初に、オンライン最小性能損失スケジューリング（ＯＭＰＬＳ）手法を実装することにより、ロボットのセットの間でタスクの第１のセットをスケジュールするよう構成することができ、タスクの第１のセットは、マージされたタスクの中でより高い性能損失値を有するタスクを含み、タスクの第１のセットは更に、待ち時間の閾値を上回り且つより高い性能損失値を有するタスクよりも優先度が低い少なくとも１つのタスクを含む。

【００２９】

更に、１又は２以上のハードウェアプロセッサ１０４は、ＯＭＰＬＳ手法を介して、ロボットのセットの間でタスクの第２のセットをスケジュールするよう構成され、タスクの第２のセットは、デッドライン内でスケジュールすることができ且つマージされたタスクの中で低い性能損失値を有するタスクを含む。ここで、提案されたＯＭＰＬＳ手法を介してスケジューリングを最適化するプロセスは、マルチロボット／マルチプロセス環境においてジョブをスケジューリング／配分している間に従来のシステム及び方法によって直面する技術的課題を考慮して詳細に検討することができる。

【００３０】

一般に、倉庫に配備されているロボットの数が制限されているので、ある時点において

、タスクの数は、ロボットのチームの能力よりも遙かに多い場合があり、このことは、一部のタスクがデッドライン内で提供されないシナリオにつながる可能性がある。倉庫は、全てのタスクを完了するよう約束しているので、システム（タスクを完了するためのロボットプロセスを実装した）上で遅延に比例した性能損失（又はペナルティ）を生じる。

【００３１】

タスクｔｉは、４つのタプル（a_i e_i d_i p_i）、すなわち、到着時間a_i、実行時間e_i、相対デッドラインd_i、及びペナルティp_iによって特徴付けられる。システム（システムを介してマルチロボット環境が実装されて実行される）が時間ゼロにてスタート（再スタート）し、ここでa_iは時間単位であり、その後で、タスクt_iがマルチロボット環境に到着して、ロボット上でe_i単位時間を占める必要があると仮定する。システムが時間s_i（＞a_i）でスタートした場合、システムは、s_i+e_i時間単位にて実行を終了する。ここでタスクは、（a_i+d_i）で与えられるデッドラインを超えた完了の単位時間遅延当たりにペナルティp_iを科す。

【００３２】

１つの実施形態において、性能損失関数は、

【数１】

$$P(t_i) = \max(0, (c_i - d_i) \times p_i) \qquad \text{式（１）}$$

として定義することができ、c_iはタスクt_iについての完了時間を表し、t_iは複数のタスクに対応する。式（１）を参照すると、タスクがデッドライン内に実行された場合には、性能損失がないが、タスクがデッドライン内に実行できない（又はデッドラインをミスする）と、性能損失は、時間に比例して増加する点に留意されたい。従って、複数のタスク、すなわち、ｍ個のロボットの間のｎ個のタスク及びロボットに割り当てられたタスクのサブセットを実行する順序を区分化することができ、Ｔ（r_j）はロボットr_jに割り当てられたタスクのサブセットを表し、タスクのサブセットは複数のタスクに対応する。

【００３３】

本開示の１つの実施形態によれば、ｍ個の同一のロボットが存在し、すなわち、Ｒ＝｛r_1，r_2，．．．，r_m｝が並行して動作して、Ｔ＝｛t_1，t_2，．．．，t_n｝として表される複数のタスクを完了し、ここで、タスクt_i（複数のタスクの中で）は、解放時間a_iにてスケジューラ（１又は２以上のハードウェアプロセッサ１０４を介してタスクをスケジュールするよう実装された）に到着したと仮定する。スケジューラは、ｍ個のロボット間のタスク（上記で検討したような）とロボットに割り当てられたタスクのサブセットを実行する順序とをマージし、Ｔ（r_j）はロボットr_jに割り当てられたタスクのサブセットを表す。従って、タスクスケジューラの目的関数は、以下の式（２）によって表すことができる。

【数２】

$$min \sum_{i=1}^{n} P(t_i) \qquad \text{式（２）}$$

【００３４】

また、これらのタスクが完了するときまでに、タスクの新しいセットが到着することができる。更に、ロボットの一群はまた、これらの一部がそのエネルギーを使い果たす可能性があり、又はエネルギーが増強された新しいロボットが加わることができるので、変化する可能性がある。従って、このような環境において、ロボット間のスケジューリングを最適化するために、最適化されたスケジューリング手法が必要とされる。

【００３５】

オンライン最小性能損失スケジューリング（ＯＭＰＬＳ）手法、すなわち、提案のＯＭ

ＰＬＳ手法は、マルチプロセッサシステム上で及びマルチロボット環境において、ノンプリエンプティブタスク（又は散発的タスク(sporadic task)）のセットのスケジューリングを最適化するのを容易にし、ノンプリエンプティブタスクのセットの各々は、デッドライン及び性能損失値を有する。提案される手法は、オンザフライで均質なロボットのチームにアトミックタスクを割り当てることを容易にし、タスクがデッドラインを満たすことができないことによって、システムに最小限のペナルティが科せられるようになる。

【００３６】

従って、ＯＭＰＬＳ手法は、新規に到着したタスク全てが記憶されている、ユニバーサルキューを維持することによって、マルチロボットシステム上で性能損失値を有する非クレボヤント(non-clairvoyant)でノンプリエンプティブタスクをスケジュールする。ＯＭＰＬＳ手法は更に、デッドライン内でタスクスケジューリングを最大にし、従来のシステム及び方法と比べて最大１の競合比を達成する。最後に、ＯＭＰＬＳ手法は、タスクがデッドライン内でスケジュールできないときにタスク実行の有界遅延を可能にする。従って、提案の手法は、タスクの枯渇を排除する。

【００３７】

本開示の１つの実施形態によれば、ＯＭＰＬＳ手法は、最初に、上記のステップ２０２からマージしたタスクを考慮する。上記のステップ２０２において検討したように、１又は２以上の予め割り当てられたタスクは、ロボットのセットの間で１又は２以上のロボットに既に割り当てられた非実行済みタスクであり、１又は２以上の新規に到着したタスクは、ロボットのセットの間で割り当てられることになる非実行済みタスクを含む。従って、予め割り当てられたタスクのセットは、後で到着するより高い優先度のタスクに置き換えられる可能性があるため、デッドライン前に実行されることを保証しないスケジュール可能なタスク(schedulable task)として定義することができる。タスクの優先度は、単位時間遅延当たりにシステムに生じることになる性能損失値によって判断される。タスクは、この優先度に基づいて分類される。最も高いペナルティ値（性能損失＝１０）を有するタスクは、最優先タスク（優先度＝１）として取り扱われる。

【００３８】

従って、ＯＭＰＬＳ手法は、キューにおけるマージされたタスクを考慮し、最重要なものとしてスケジュールしようと努め、これは、デッドラインを超えて遅延した場合に高いペナルティが生じることになるタスクである。遅延したタスクは、再評価されて、システム全体の性能損失を最小限にするようにスケジュールされる。ＯＭＰＬＳ手法は、しばらくの間低い性能損失値を有するタスクを遅延させて、高いペナルティを有するタスクを早期にスケジュールする。しかしながら、低いペナルティ値を有するタスクの連続した遅延は、その枯渇につながる可能性があり、従って、最適化されたスケジューリングが必要とすることができる。

【００３９】

１つの実施形態において、任意の瞬間時間τにおいて、マルチロボットシステムは、以下の状態のうちの１つの状態とすることができる。

全てのロボットは、それぞれのタスクの実行に従事しており、実行を続けている；

現在のタスクを完了する一部のロボットは、新しいタスクをピックする準備ができているものがあり：

新しいタスクがシステムに到着していない場合、それぞれのロボットのタスクキューからタスクをピックする；

新しいタスクがシステムに導入された場合、サブルーチンＴａｓｋＳｃｈｅｄｕｌｉｎｇに続く；

到着した新しいタスクが存在せず、ロボットのタスクキューに利用可能なタスクが存在しない場合、アイドル状態(idle)のままでいる。

【００４０】

１つの実施形態において、１又は２以上のハードウェアプロセッサ１０４は、ロボットがタスクを終了し、かつ少なくとも１つの新しいタスクがマルチロボットシステムに導入

されるか又は新しいロボットがロボット群に加わるたびに、サブルーチンＴａｓｋＳｃｈｅｄｕｌｉｎｇを実施する(implement)。直感的に、予め割り当てられたタスクのセット（既にロボットに割り当てられた）を新しく到着したタスクのセットとマージし、次いでスケジュールを決定すると、性能損失値が最小になり、スケジューリングが最適化される。サブルーチンＴａｓｋＳｃｈｅｄｕｌｉｎｇに続いて、タスクがロボットに配分される。

アルゴリズム：オンライン最小性能損失スケジューリング（ＯＭＰＬＳ）手法

ノンプリエンプティブタスクｔをスケジュールするための発見的問題解決法により、時間τにおけるペナルティ全体が最小になる。

入力：配分待ちの待機キュー（ω）のタスク。倉庫におけるアクティブなロボットＲ

出力：各ロボット（$R_{schedule}$）のタスクのスケジュール

1. Routine $TaskScheduling\ (T, R)$
2. $//Schedule\ High-priority\ Tasks$

$\bar{\omega} \leftarrow \omega\ \forall T_i \in \omega, T_i.priority = 0;$

3. $R \leftarrow compaction(\bar{\omega})$;
4. $R \leftarrow listScheduling\ (\bar{\omega})$;
5. $//Schedule\ Remaining\ Tasks$

$R \leftarrow compaction(\bar{\omega})$;

6. $R \leftarrow listScheduling\ (\omega)$
7. $updatePriority\ ()$;

【００４１】

１つの実施形態において、最初に、マージされた複数のタスクの各々は、ロボットのセットから利用可能なロボット（すなわち、タスクを実行していないロボット）の間でスケジューリングを考慮することができる。次いで、１又は２以上のハードウェアプロセッサ１０４は、コンパクションチェックを実施して、デッドライン内のタスクt_iをスケジュールすることができ、コンパクションチェックは、時間間隔［a_i，a_i+d_i］において各ロボットにわたって実施することができる。換言すると、ＯＭＰＬＳ手法は、ｅｉ連続時間単位がこの期間にロボットに配置される可能性があるかどうかのチェックを実施し、予め割り当てられたタスクのセットの中の他のタスクにデッドラインミスを強制することはない。

【００４２】

１つの実施形態において、ロボットr_kでタスクt_iをスケジュールしてもスケジュール可能なタスクがそのデッドラインをミスしないことを更に確実にするために、下記の式（３）及び（４）の両方を満たす必要がある。

【数３】

$$d_l - \left(\sum_{\substack{t_j \in T(r_k) \\ d_j \leq d_l}} e_j + e_l \right) \geq 0 \qquad \text{式（３）}$$

【数４】

$$\forall t_j \in T(r_k) : d_j > d_i : d_j - \left(\sum_{\substack{t_j \in T(r_k) \\ d_l \leq d_j}} e_l + e_j \right) \geq 0 \qquad \text{式（４）}$$

式（３）は、ロボットr_kでスケジュールされた全てのタスクを実行した後、タスクt_lを実行するのに十分なスペースを提供し、t_lよりも小さいデッドラインを有し、式（４）は、既にｒｋにスケジュールされており且つt_iよりも大きいデッドラインを有するタスクが、ｔｉをスケジュールする場合でもスケジュール可能であることを保証する。

【００４３】

１つの実施形態において、ステップ２０３にて、１又は２以上のハードウェアプロセッサ１０４は、最後に、リストスケジューリング手法により、タスクの残りのサブセットの性能損失値を最小にするように構成され、タスクの残りのサブセットは、予め定義されたデッドライン内にスケジュールすることができないタスクを含む。［a_i，$a_i - d_i$］の後に始まるようにスケジュールすることができるタスクt_iは、予め定義されたデッドライン内に実行を完了するタスクである。（$d_i - e_i$）後に実行を開始するタスクは、d_iがマルチロボットシステムに性能損失を科した後に実行を完了すべきである。（$d_i - e_i$）後に実行を開始するタスクは、リストスケジューリング手法を実装することによって、ロボットの間で配分することができる。

【００４４】

１つの実施形態において、タスクの残りのサブセットの性能損失値を最小にするステップは、リストスケジューリング手法を介して、ノンプリエンプティブタスクのスケジューリングを最適化するために新規に到着したタスクから１又は２以上の高優先度タスクを取り除くステップを含み、取り除かれた１又は２以上の高優先度タスクは、ロボットのセットの間でアイドル状態のロボットに割り当てられ、アイドル状態のロボットは、複数のタスクのいずれも実行していないロボットである。従って、最も高い優先度を有するタスクは、待機キューから削除され、第１のロボット（例えば、r_k、$1 \leqq k \leqq m$）に割り当てることができ、ロボットr_kは、アイドル状態であるか、又はロボットのセットの間で最先のタスクを終了するロボットである。１又は２以上のハードウェアプロセッサ１０４は、複数のタスクの中の各タスクが少なくとも１つのロボットに割り当てられるまで、１又は２以上の高優先度タスクを反復して取り除くステップを実施する。

【００４５】

前述のように、枯渇につながる可能性がある、低いペナルティ値を有するタスクの連続的な遅延が存在する場合に、ロボットプロセスを実装する従来のシステム及び方法は機能しない。従来のシステム及び方法によって直面されるこの制限を克服するために、開示される方法、すなわち、ＯＭＰＬＳ手法は、ノンプリエンプティブタスクのスケジューリングを最適化するためタスクの残りのサブセットの各々に関して優先度更新を実行する。１つの実施形態において、各タスクがロボットに割り当てられた（又はロボットの待機キューに配分された）後、１又は２以上のハードウェアプロセッサ１０４は、デッドラインに違反する可能性があるロボットに対して優先度値実行（又は起動）する。

【００４６】

このため、マルチロボット環境におけるノンプリエンプティブタスクのスケジューリングを最適化するために、１又は２以上のハードウェアプロセッサ１０４は、タスクの残りのサブセットの各々の実行時間が初期の予め定められた閾値よりも大きいと判断されると、タスクの残りのサブセットの各々の優先度関数をインクリメントすることによって、優先度更新を実行し、該初期の予め定められた閾値は、予め定められたデッドラインを用いて識別される。従って、現在時間τがその初期の予め定められた閾値を超えた場合には、タスクのサブセットの各々の優先度は、１単位だけインクリメントされる。

【００４７】

１つの実施形態において、タスクについての初期の予め定められた閾値は、最終デッドラインと同一に設定することができる。従って、閾値（ｔｈ$_i$）は、以下の式（５）のように定義することができる。

【数５】

$$th = \begin{cases} th + (d_i - a_i), if\ r \geq th_i \\ th,\ otherwise \end{cases} \qquad 式（５）$$

上記の式（５）を参照することにより、タスクＴ$_i$がその予め定められた初期閾値(pre-defined initial threshold)内に実行できない場合、１又は２以上のハードウェアプロセッサ１０４は、相対デッドラインに等しい初期値だけ上記閾値をインクリメントし、タスク優先度は、単位ステップだけインクリメントすることができ、閾値をインクリメントすることにより、ｐｒｉｏｒｉｔｙ$_i$の反復の後、タスクＴ$_i$は、クリティカルタスク（ｐｒｉｏｒｉｔｙ＝０）として識別され、これにより最高優先度を得ることができる点に留意されたい。１又は２以上のハードウェアプロセッサ１０４は、かかる１又は複数のクリティカルタスクをロボットに割り当て、このようにして識別され割り当てられた１又は複数のクリティカルタスクは、他の何れかの１又は複数のタスクと置き換えることができない。

【００４８】

本開示の１つの実施形態によれば、ＯＭＰＬＳ手法の性能評価は、シミュレーション結果を介して検討することができる。開示される方法は、ＯＭＰＬＳ手法の性能を評価するための容量制約式ビークルルーティング問題（ＣＶＲＰ）のデータセット（ｖｒｐ）を実装し、デポットロケーションは、パッケージドックロケーションを用いることができ、サイトロケーションは、ロボットによってピックアップされる１又は複数の物体の保管場所として用いることができる。シミュレーション結果は、このセットから１０００タスクだけ（すなわち、サイトロケーション）について検討している。

【００４９】

前述のように、各タスクは、タプルとして表すことができ、すなわち、ｔｉ＝＜a$_i$，e$_i$，d$_i$，p$_i$＞、ここで実行時間（e$_i$）は、パッケージドックから物体の保管場所（ストレージロケーション）までのロボットの往復の移動時間として設定することができる。実行時間としては、以下のストラテジーを考慮することができる：

d$_i$＝２e$_i$、ここでタスクのデッドラインd$_i$は、対応する実行時間e$_i$の２倍に設定される。ロボットの数と比べてタスクの数が多いことに起因して、複数のタスクがデッドラインをミスする可能性があるが、このような希なシナリオは、開示される方法を実装することによって効果的に対処される；

d$_i$∈［２e$_i$，１０e$_i$］、デッドラインd$_i$は、２e$_i$，１０e$_i$の範囲でランダム値に設定され、ここで２e$_i$，１０e$_i$はランダムに分散される；

d$_i$∈［５e$_i$，１０e$_i$］、デッドラインd$_i$は、５e$_i$，１０e$_i$の範囲で均一な分布を用いて設定される。これは、大量のタスクが、その（全てではないが）の各々の対応するデッドライン内にスケジューリング可能であるシナリオを表しており；

ｄ←ｍｉｘ、スケジューラがデッドラインの特定の分布を優先しないことを確保するために等確率の３つの以前のデータセットのミックスを含む。

【００５０】

更に、上述の実行時間とデッドラインの各組み合わせにおいて、１又は２以上のハードウェアプロセッサ１０４は、以下の３つのストラテジーを用いて各タスクに対するペナルティを割り当てる：

ｐ←［１，１０］、性能損失は、１～１０の範囲で均一な分布を用いて生成される１又は２以上のランダム値に設定される；

ｐ←ｓａｍｅ、性能損失は、複数のタスクの各々について１つの同一の値（例えば、１）に設定される；

ｐ←ｅｘｔｒｅｍｅ［１　ｏｒ　１０］、性能損失は、２つの極値の何れか、すなわち、

１又は１０に等確率で設定される。

【００５１】

１つの実施形態において、ＯＭＰＬＳ手法は、２つのメトリクス、すなわち、全てのタスクを完了している間に被った全体の性能損失と、時間τ_0とτ’（ここでτ’＞τ_0）の間のデッドラインミスの数とに基づいて評価することができる。スケジューラの性能を確認するために、安定したマルチロボットシステムが時間τ_0まで表すことができる。時間τ_0の後、タスクのバーストがマルチロボットシステムにおいて時間τ’までに到着し、時間τ’の後、タスクインフローが減少する。図３及び図４を参照することにより、バースト中のタスク到着パターンを参照することができ、図３は、特定の時間スパン－定常高トラフィックに対する連続した高インフローレートを示し、図４は、複数のタスクが小時間スケールで出現しており、パターンが一部の時間－スパイクトラフィックで続いていることを示す。

【００５２】

１つの実施形態において、デッドラインミスのタスクの数並びに被った対応する性能損失値は、マルチロボットシステムにおいてタスクが出現したレート及びタスクを実行するのに利用可能なロボットの数に依存する。アクティブなロボットの数（ロボットの総数の中の）は、タスクバーストの前後マルチロボットシステムの堅牢性を把握するために、５０個に固定することができ、このような高トラフィック時間の間におよそ１０００個が追加された。開示の方法は、例えば、Ｔｕｒｎｅｒ他による最短デッドライン優先－新規タスク優先（ＥＤＦ－ＮＴＦ）など、従来のシステム及び方法と比較した。

【００５３】

図３及び図４を参照することにより、高トラフィック期間中に到着したタスクについて累積されたデッドラインをミスしたタスクの数及び性能損失について参照することができる。マルチロボットシステムは、タスクの急増が生じる前に安定状態であった。図３及び図４を再度参照することにより、固定数のロボットでは、タスクインフローのレートが大幅に増大すると、マルチロボットシステムは、安定性を失い、実行中の多数のタスクがデッドライン制約に違反することにつながる点に留意されたい。性能損失はまた、漸次的に累積し始める。タスクバーストの終わりには、タスク到着レートが漸次的に正常値に下がり、マルチロボットシステムが安定性を獲得して、更なるデッドラインミス及び性能損失の累積は存在しない。タスク到着パターンの両方の種類（安定した高トラフィック及びスパイクトラフィック）では、スケジューラ性能が類似している。高トラフィックの間に実験が実施された。

【００５４】

図５を参照することにより、時間τ_0＋２５にて到着したタスクについての実行パターンを参照することができる。ＯＭＰＬＳ手法は、例えば、ＥＤＦ－ＮＴＦ手法のような従来のシステム及び方法と比べて、より少ないタスクがデッドラインをミスするようにトラフィックをスケジュールする。ＯＭＰＬＳ手法は、より低いペナルティ値（例えば、タスクｉｄ５８）でタスクを遅延させて、より高いペナルティを有するタスクを優先させて、システムに対する全体の損失／ペナルティを最小限にする。低ペナルティタスクの実行における一定の遅延は、枯渇につながる可能性がある。枯渇回避もまた、上記のステップ２０３で検討したような提案のＯＭＰＬＳ手法によって排除される。

【００５５】

本開示の１つの実施形態に従って、提案の開示の堅牢性、結果、技術的利点、性能損失の最小化、並びに従来のシステム及び方法に対する比較について更に考察することができる。以下の表１～４を参照することにより、ＯＭＰＬＳ手法は、特定のタスク特性を優先しない点に留意されたい。ＯＭＰＬＳ手法は更に、ペナルティ値及び性能損失値を従来のシステム及び手法（ｄ＝例えば、オークションベースのアルゴリズム（ＡＵＣ－オフライン））と比較することにより評価することができる。

【表１】

性能損失	デッドライン	AUC-オフライン			EDF-NTF		
		$P(10^2\times)$	$DL-M$		$P(10^2\times)$	$DL-M$	
$p_i \in [1,10]$	$d_i = 2e_i$	14408	622		13226	737	
	$d_i \in [e_i, 10e_i]$	5343	197		5578	405	
	$d_i \in [5e_i, 10e_i]$	2954	149		3110	172	
$d \leftarrow mix$		7633	218		7608	597	
$p \leftarrow same$	$d_i = 2e_i$	2602	622		2378	93	
	$d_i \in [2e_i, 10e_i]$	966	197		1004	48	
	$d_i \in [5e_i, 10e_i]$	520	149		574	172	
	$d \leftarrow mix$	1315	218		1370	597	
$p \leftarrow extreme$							
	$d_i = 2e_i$	13812	622		12571	737	
	$d_i \in [2e_i, 10e_i]$	5096	197		5381	405	
	$d_i \in [5e_i, 10e_i]$	2419	149		2796	172	
	$d \leftarrow mix$	7086	218		7356	597	

【表２】

性能損失	デッドライン	OMPLS手法		
		$P(10^2\times)$	$DL-M$	
$p_i \in [1,10]$	$d_i = 2e_i$	10482	699	
	$d_i \in [e_i, 10e_i]$	3925	290	
	$d_i \in [5e_i, 10e_i]$	1619	214	
$d \leftarrow mix$		6098	311	
$p \leftarrow same$	$d_i = 2e_i$	2526	624	
	$d_i \in [2e_i, 10e_i]$	918	205	
	$d_i \in [5e_i, 10e_i]$	498	156	
	$d \leftarrow mix$	1212	214	
$p \leftarrow extreme$				
	$d_i = 2e_i$	6791	683	
	$d_i \in [2e_i, 10e_i]$	1172	243	
	$d_i \in [5e_i, 10e_i]$	686	181	
	$d \leftarrow mix$	2157	267	

【表３】

性能損失	デッドライン	性能損失値低減 %	
		$AUC-offline$に対する	$EDF-NTF$に対する
$p_i \in [1,10]$	$d_i = 2e_i$	27.25	20.14
	$d_i \in [e_i, 10e_i]$	26.55	29.64
	$d_i \in [5e_i, 10e_i]$	45.17	47.92
	$d \leftarrow mix$	20.10	19.85
$p \leftarrow same$	$d_i = 2e_i$	2.94	-6.23
	$d_i \in [2e_i, 10e_i]$	5.00	8.56
	$d_i \in [5e_i, 10e_i]$	4.32	13.29
	$d \leftarrow mix$	7.77	11.51
$p \leftarrow extreme$	$d_i = 2e_i$	50.83	45.98
	$d_i \in [2e_i, 10e_i]$	76.98	78.20
	$d_i \in [5e_i, 10e_i]$	71.63	75.45
	$d \leftarrow mix$	69.55	70.67

【表４】

性能損失	デッドライン	競合比 (デッドライン適合) $AUC-offline$に対する
$p_i \in [1,10]$	$d_i = 2e_i$	0.79
	$d_i \in [e_i, 10e_i]$	0.88
	$d_i \in [5e_i, 10e_i]$	0.92
$d \leftarrow mix$		0.88
$p \leftarrow same$	$d_i = 2e_i$	0.99
	$d_i \in [2e_i, 10e_i]$	0.99
	$d_i \in [5e_i, 10e_i]$	0.99
	$d \leftarrow mix$	1
$p \leftarrow extreme$	$d_i = 2e_i$	0.83
	$d_i \in [2e_i, 10e_i]$	0.94
	$d_i \in [5e_i, 10e_i]$	0.96
	$d \leftarrow mix$	0.93

【００５６】

ＡＵＣオフライン手法では、デッドライン内にできるだけ多くのタスクを実行することで利益を最大化しようとしているが、スケジュール可能なタスクの最後に残りのタスク（デッドラインをミスした）のスケジューリングを提供しない。ＯＭＰＬＳ手法は、スケジュール可能なタスクの最後に残りのタスク（デッドラインをミスした）をスケジュールする。これにより、ＡＵＣベースのスケジューリングにおいて累積した性能値が得られる。表３は、デッドラインに適合するタスクの数の観点で、ＡＵＣオフライン手法に対するＯＭＰＬＳ手法の優位性を提供する。表４を参照することにより、ＯＭＰＬＳ手法は、最大競合比１及び最小値０．８３を得る点に留意されたい。

【００５７】

図６を参照することにより、提案のＯＭＰＬＳ手法は、ノンプリエンプティブタスクのスケジューリングを最適化することにより、従来のシステム及び方法よりも性能が優れている点に留意されたい。直感的には、タスクを実行するために利用可能なロボットの数が多い場合のＯＭＰＬＳ手法では、デッドラインミス及び性能損失値の最小化に関してスケジューラの性能を改善する必要がある。この傾向は、ＡＵＣオフライン手法、ＥＤＦ－ＮＴＦ手法、並びに開示された方法によって反映される。利用可能なロボットの数は、２０～１００の間で変化し、この変化は２つの態様、すなわち、リソース（ロボット）の変化する数に伴うスケジューラの堅牢性と、リソースの数の増加に伴う性能の改善に直面する。

【００５８】

タスクが遙かに大きなデッドラインを有する、$d_i \in [5e_i, 10e_i]$では、７０個のロボットを用いることにより、ほとんど全てのタスクは、デッドライン制約のどれもミスすることなく、提案のＯＭＰＬＳ手法を実装することによって成功裏に実行される。同じ

性能を達成するためには、ＥＤＦ－ＮＴＦ手法では、より多数のロボットが必要であった。更に、$d_i \in [2e_i, 10e_i]$及びｄ←ｍｉｘでは、驚異的な数のタスクがデッドラインをミスすることになる場合、提案のＯＭＰＬＳ手法は、発生した性能損失値の点で、ＡＵＣオフライン手法及びＥＤＦ－ＮＴＦ手法の両方よりも優れている。デッドラインをミスするタスクの数は、常にＡＵＣオフライン手法よりも少なく、これは、ＡＵＣオフライン手法は、開始時にタスク設定全体がスケジューラに既知であるオフライン手法であるということに起因することができる点に留意されたい。提案のＯＭＰＬＳ手法は、ペナルティの少ないタスクを遅延させ、他方、従来のシステム及び手法、例えば、ＥＤＦ－ＮＴＦ手法は、タスクをそのデッドラインを超えて遅延させている。

【００５９】

本明細書は、当業者が本発明の実施形態を実施及び利用できるように、本明細書で主題を記載している。本主題の実施形態の範囲は、請求項によって定められ、当業者であれば想起される他の修正形態を含むことができる。他のこのような修正形態は、請求項の文言と相違しない同様の要素を有する場合、又は請求項の文言と僅かな差異を有する均等な要素を含む場合には、請求項の範囲内にあるものとする。

【００６０】

本明細書における本開示の実施形態は、マルチロボット環境におけるノンプリエンプティブタスクのスケジューリングを最適化するという未だ解決されていない問題に取り組んでいる。すなわち、本実施形態は、ＯＭＰＬＳ手法を実装することにより、最初に、より高い性能損失値を有するタスク、次に、デッドライン内にスケジュールすることができ、かつマージされたタスクの中で低い性能損失値を有するタスクをスケジューリングすること、及び最後に予め定義されたデッドライン内にスケジュールすることができないタスクの残りのサブセットの性能損失値を最小化することを提供する。また、本明細書の実施形態は更に、ノンプリエンプティブタスクのスケジューリングを最適化するためタスクの残りのサブセットの各々に対し優先度更新の実行を提供する。

【００６１】

本特許の保護範囲は、このようなプログラムにまで、及びこれに加えてメッセージを内部に有するコンピュータ可読手段にまで拡張され、かかるコンピュータ可読記憶手段は、プログラムがサーバ又はモバイルデバイス上で又は何れかの好適なプログラマブルデバイス上で実行されたときに、本方法の１又は２以上のステップを実装する（実施する）ためのプログラムコード手段を含む点に留意されたい。ハードウェアデバイスは、例えば、サーバ又はパーソナルコンピュータ又は同様のもの、もしくはこれらの組み合わせのような何れかの種類のコンピュータを含む、プログラムすることができる何れかの種類のデバイスとすることができる。デバイスはまた、例えば、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）などのハードウェア手段、又は、例えば、ＡＳＩＣ及びＦＰＧＡ、又はソフトウェアモジュールが配置された少なくとも１つのマイクロプロセッサ及び少なくとも１つのメモリなどのハードウェアとソフトウェアの組み合わせとすることができる手段を含むことができる。従って、本手段は、ハードウェア手段とソフトウェア手段の両方を含むことができる。本明細書で記載される方法の実施形態は、ハードウェア及びソフトウェアにおいて実装することができる。デバイスはまた、ソフトウェア手段を含むことができる。代替として、本実施形態は、例えば、複数のＣＰＵを用いて、様々なハードウェアデバイス上で実装することができる。

【００６２】

本明細書の実施形態は、ハードウェア要素とソフトウェア要素とを含むことができる。ソフトウェアにて実装される実施形態は、限定ではないが、ファームウェア、常駐ソフトウェア、マイクロコード、その他を含む。本明細書で記載された種々のモジュールによって実施される機能は、他のモジュール又は他のモジュールの組み合わせにおいて実装することができる。本明細書において、コンピュータ使用可能又はコンピュータ可読媒体は、命令実行システム、装置又はデバイスによって又はこれらと接続して使用されるプログラムを包含、記憶、通信、伝搬、又は伝送することができる何らかの装置とすることができ

る。

【００６３】

例示されるステップは、図示の例示的な実施形態を説明するために提示され、進行中の技術開発が、特定の機能が実施される方式を変化させるであろうことは理解されたい。これらの実施例は、本明細書では限定ではなく説明の目的で提示されている。更に、説明の便宜上、本明細書では機能的構成単位の境界が任意に定義された。指定される機能及びそれらの関係が適切に行われる限り、代替の境界を定義することができる。（本明細書に記載されるものの等価物、拡張、変形、逸脱などを含む）別の代替手段は、本明細書に含まれる教示に基づいて当業者には明らかであろう。かかる代替手段は、開示される実施形態の範囲及び技術的思想の範囲内にある。更に、単語「含む（ｃｏｍｐｒｉｓｉｎｇ）」、「有する（ｈａｖｉｎｇ）」、「含む（ｃｏｎｔａｉｎｉｎｇ）」及び「含む（ｉｎｃｌｕｄｉｎｇ）」並びに他の同様の語形は、意味が等しいことが意図され、これらの単語の何れかに続く単数又は複数の項目が単数又は複数のかかる項目の網羅的なリストであることを意味せず、リストされた単数又は複数の項目のみに限定されることも意味しないという点で非限定的であることが意図される。同様に、本明細書及び添付の特許請求の範囲において、文脈が明らかに別のことを指示しない限り、単数形「１つの（ａ）」、「１つの（ａｎ）」、及び「前記（ｔｈｅ）」が複数の参照を含むことも留意しなくてはならない。

【００６４】

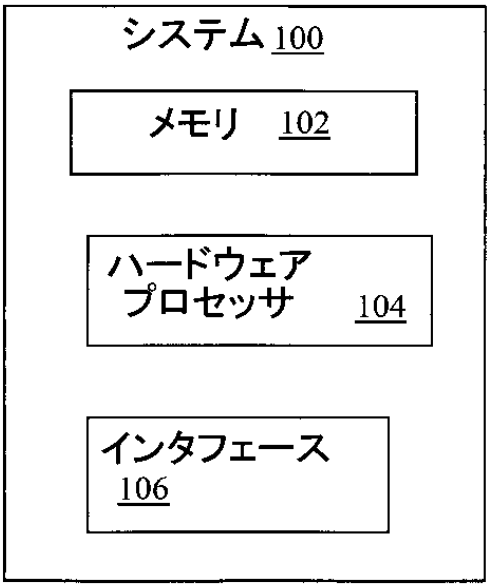
更に、本開示に適合する実施形態を実装する際に、１又は２以上のコンピュータ可読記憶媒体を利用することができる。コンピュータ可読記憶媒体は、プロセッサによって可読な情報又はデータを記憶することができる何れかのタイプの物理メモリを指す。従って、コンピュータ可読記憶媒体は、本明細書に記載される実施形態に適合するステップ又は段階をプロセッサ（単数又は複数）に行わせるための命令を含む、１又は２以上のプロセッサによる実行のための命令を記憶することができる。用語「コンピュータ可読媒体」は、有形の品目を含み、搬送波及び過渡信号を除く、すなわち非一時的であると理解されるべきである。実施例は、ランダムアクセスメモリ（ＲＡＭ）、リードオンリメモリ（ＲＯＭ）、揮発性メモリ、不揮発性メモリ、ハードドライブ、ＣＤ ＲＯＭ、ＤＶＤ、フラッシュドライブ、ディスク及びその他の既知の物理記憶媒体を含む。

【００６５】

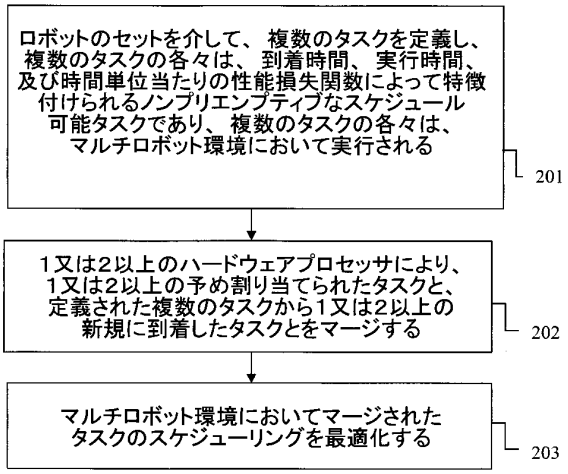
本開示及び実施例は、単に例示的なものとみなされ、開示される実施形態の真の範囲及び精神は、添付の特許請求の範囲によって示される。

【図面】

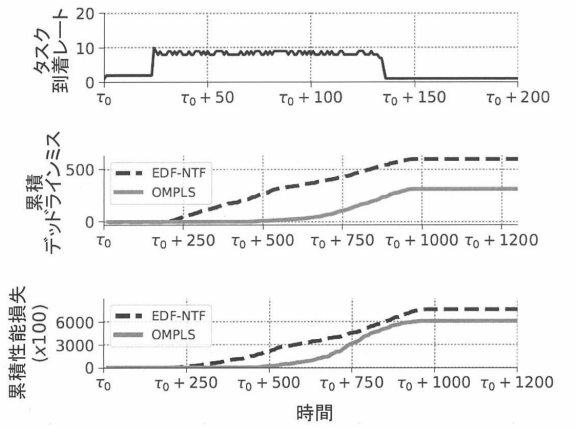
【図１】


システム 100
メモリ 102
ハードウェア
プロセッサ 104
インタフェース
106

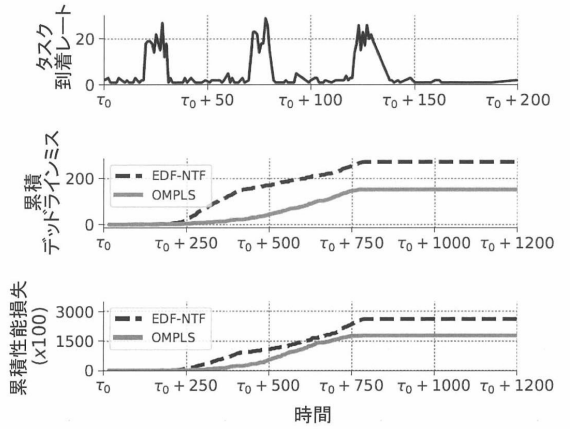
【図２】


ロボットのセットを介して、 複数のタスクを定義し、
複数のタスクの各々は、 到着時間、 実行時間、
及び時間単位当たりの性能損失関数によって特徴
付けられるノンプリエンプティブなスケジュール
可能タスクであり、 複数のタスクの各々は、
マルチロボット環境において実行される
201
1又は2以上のハードウェアプロセッサにより、
1又は2以上の予め割り当てられたタスクと、
定義された複数のタスクから1又は2以上の
新規に到着したタスクとをマージする
202
マルチロボット環境においてマージされた
タスクのスケジューリングを最適化する
203

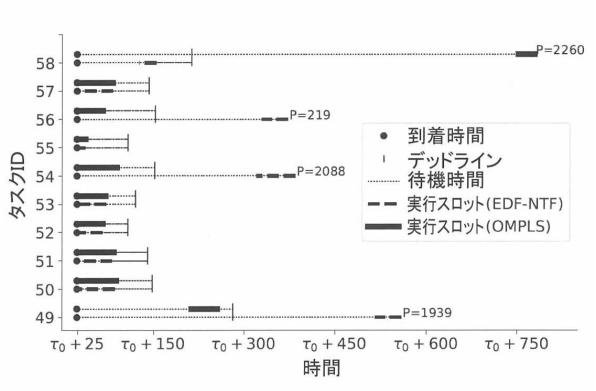
【図３】


タスク
到着レート
累積
デッドラインミス
EDF-NTF
OMPLS
累積性能損失
(x100)
時間

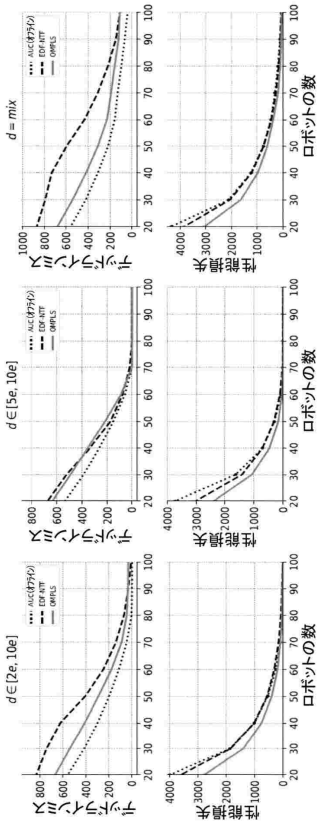
【図４】


タスク
到着レート
累積
デッドラインミス
EDF-NTF
OMPLS
累積性能損失
(x100)
時間

【図５】


タスクID
58
57
56
55
54
53
52
51
50
49
P=2260
P=219
P=2088
P=1939
到着時間
デッドライン
待機時間
実行スロット(EDF-NTF)
実行スロット(OMPLS)
τ_0+25
τ_0+150
τ_0+300
τ_0+450
τ_0+600
τ_0+750
時間

【図６】


$d=mix$
$d\in[5e,10e]$
$d\in[2e,10e]$
AUC(オフライン)
EDF-NTF
OMPLS
タスクのスループット
性能損失
ロボットの数

フロントページの続き

弁理士　▲吉▼田 和彦
(74)代理人　100109070
弁理士　須田 洋之
(74)代理人　100067013
弁理士　大塚 文昭
(74)代理人
西島 孝喜
(74)代理人
上杉 浩
(74)代理人　100120525
弁理士　近藤 直樹
(74)代理人　100139712
弁理士　那須 威夫
(74)代理人　100196612
弁理士　鎌田 慎也
(72)発明者　マリーチ アガルワル
インド ７００１６０ ウェスト ベンガル コルカタ ラジャルハット ディストリクト ノース ２４ パルガナス ラジャルハット ポリス ステイション ニュー タウン エム ワイド （アール オー ダブリュ） ロード ストリート ５９ プロット ナンバー セカンドエフ／１２ （オールド ナンバー エイエイ セカンド／ブロック ３ アイ ティー） エコ スペース ブロック １ビー タタ コンサルタンシー サービシズ リミテッド内
(72)発明者　チャヤン サルカール
インド ７００１６０ ウェスト ベンガル コルカタ ラジャルハット ディストリクト ノース ２４ パルガナス ラジャルハット ポリス ステイション ニュー タウン エム ワイド （アール オー ダブリュ） ロード ストリート ５９ プロット ナンバー セカンドエフ／１２ （オールド ナンバー エイエイ セカンド／ブロック ３ アイ ティー） エコ スペース ブロック １ビー タタ コンサルタンシー サービシズ リミテッド内
審査官　森林 宏和

(56)参考文献　米国特許出願公開第２０１７／０１７３７８４（ＵＳ，Ａ１）
米国特許出願公開第２０１８／０２７５６７９（ＵＳ，Ａ１）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｂ６５Ｇ　１／００ － １／２０
Ｇ０５Ｂ　１９／４１８
Ｇ０６Ｑ　１０／００ － ９９／００