学術論文

ロボットの動的制御における計算の並列化問題に対する
最適化および準最適化アルゴリズム田 川 聖 治* 福 居 毅 至* 神 吉 良 英*
太 田 有 三* 羽根田 博 正*Optimization and Quasi-Optimization Algorithms
for Parallel Processing of Robot-Arm Dynamic Control ComputationKiyoharu Tagawa*, Tuyoshi Fukui*, Yoshihide Kanki*,
Yuzo Ohta* and Hiromasa Haneda*

A parallel processing scheme is described for robot control computation on MIMD shared memory multi-processor model. Since dynamic control law of robot arm usually contains a large amount of operations compared with the number of available processors, it is important to consider not only the effect of parallel processing but also the efficiency of serial processing in each of the processors. In order to obtain such a desirable solution for the complex scheduling problem, optimization and quasi-optimization algorithms are proposed. The excellent optimization algorithm is based on a branch-and-bound method. On the other hand, the practical quasi-optimization algorithm is based on a partial enumeration method which effectively combines the optimization algorithm and the approximation algorithm (GCF/LPT); that was recently developed by authors. The proposed scheduling algorithms are applicable to parallel processing of any kinds of control laws represented by sum of products. The test results on the dynamic control computation of robot arms demonstrate the usefulness of the algorithms.

Key Words: Parallel Processing, Scheduling Algorithm, Branch-and-Bound, Factorization

1. は じ め に

ロボット・アームに対して、その非線形な動力特性を補償するような動的制御[1]を行うためには、複雑な関節トルクの計算を実時間で行うことが必要となる。この問題に対する一つの解決策として、複数のプロセッサを用いた並列処理がある[2]~[4]。本論文では、ロボット・アームの制御則が多変数の積和式で表されることに着目し、積和式の計算処理を複数のプロセッサに割り当てることを考える。

計算の並列化問題については、結合則と交換則を用いた計算木の高さの削減アルゴリズム（ツリーハイトリダクション）などがよく知られている[5][6]。ところが、2分木状の演算を効率よく行うためには、プロセッサ間の通信が自由に効率よく行える必要があるが、実際の並列計算機においては、プロセッサ間の結合網はハードウェア量などの問題から制限されることが多い[7]。また、ロボット・アームの制御則の計算に必要な演算量は膨大であるため、これらの並列計算手法が有効に適用できるほど十分な数のプロセッサを準備することも現実的ではない。さらに、近年、VLSI技術の進歩に伴いプロセッサ単体の機能が向上しており、例えばDSP（ディジタル・シグナル・プ

ロセッサ）[8]のようにプロセッサ内部において演算のパイプライン処理を行うような場合には、プロセッサ間の頻繁な通信処理や同期処理が各プロセッサにおける演算処理の効率を低下させ、期待される程の台数効果が得られない。

このようにプロセッサの数や通信が制限されている場合、計算の並列化のみならず、Fig. 1に示すような因数分解による冗長な演算の削除も考慮すべきである。特に積和式で与えられるロボット・アームの制御則は、共通な因子を多く含んでいるので、因数分解による計算式の簡単化は有効である[9]。しかし、Fateman [10]の指摘にもあるように、計算式の並列化と簡単化は一般に相矛盾する問題であり、各プロセッサにおける逐次処理の効率を考慮した計算の並列化問題は極めて難しいスケジューリング問題となる。

本論文では、上記の並列化問題に対して最適化アルゴリズムと準最適化アルゴリズムを提案する。ロボット制御則の計算のように、際限なく何度も繰り返される処理の並列化問題において、最適解、あるいは、それに極めて近い準最適解を求める意義は大きい。今回提案する最適化アルゴリズムは分枝限定法[11]に基づき最適解を厳密に求めるものである。分枝限定法においては、いかに効率的に適切な部分問題を選ぶかが生成される部分問題の個数、すなわちアルゴリズムの計算時間を左右する。したがって、提案する最適化アルゴリズムにおいては、等価な部分問題を排除して必要最少限の部分問題を生成するた

原稿受付 1995年7月27日

*神戸大学工学部

*Faculty of Engineering, Kobe University

Fig.1 Desirable transformation

めの分枝操作と、問題固有の知識を利用した下界値による限定操作を考案することにより、不要な部分問題の発生を極力防ぐような工夫を凝らしている。

また、準最適化アルゴリズムは一種の部分列挙法[12]に基づくものであり、定義領域を分割して解に大きな影響を与える領域に対しては上記の最適化アルゴリズムを、影響の小さい領域に対しては筆者らが先に提案した近似アルゴリズム[13]を適用する。この手法によれば、現実には発生する大規模な並列化問題に対処することができるほか、2種類のアプローチの適用範囲の割合によって、得られる準最適解の精度と求解に要する計算時間を調節することも可能となる。

以降、第2章では、数式処理システムを用いて展開したロボット・アームの制御則が積和式で表されることを示す。第3章では、並列計算機モデルと制御則の並列処理手法について説明し、並列化問題を定式化する。第4章では、分枝限定法に基づく最適化アルゴリズムを提案し、計算例によって探索効率を評価する。第5章では、実用的な準最適化アルゴリズムを提案し、得られる解の精度とアルゴリズムの計算時間の関係について考察を行う。最後に、第6章では、本研究の成果をまとめると共に、今後の課題について触れる。

2. 動的制御におけるロボットの制御則

ロボット・アームの動的制御とは、ロボット・アームの運動方程式（非線形モデル）に基づき制御系を設計することであり、非線形特性による影響が無視できない高速、高精度な制御に有効である[1]。

N 個の関節 (N 自由度) を持つロボット・アームの Lagrange 運動方程式は次式で表される。

$$\mathbf{f} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\dot{\mathbf{q}}, \mathbf{q})\dot{\mathbf{q}} + D\dot{\mathbf{q}} + g(\mathbf{q}) \quad (1)$$

ただし、 $\mathbf{f} \in \mathbf{R}^N$ は関節に加える駆動トルクまたは力、 $\mathbf{q} \in \mathbf{R}^N$ は関節の変位を表す。また、式 (1) 右辺の第1項は慣性力を、第2項はコリオリ力と遠心力を、第3項は摩擦抵抗を、第4項は重力の影響を表す。

Fig.2 Two-link robot-arm

式 (1) によりモデル化されたロボット・アームに対して動的制御を行うためには、次の式 (2) に示す逆動力学計算、あるいは式 (3) に示す線形化補償器の制御則に基づき、関節に加えるトルクまたは力 $\mathbf{f}$ の計算を実時間で行う必要がある[14][15]。

$$\mathbf{f} = M(\mathbf{q})\ddot{\mathbf{p}} + C(\dot{\mathbf{q}}, \mathbf{q})\dot{\mathbf{q}} + D\dot{\mathbf{q}} + g(\mathbf{q}) \quad (2)$$

$$\mathbf{f} = M(\mathbf{q})\ddot{\mathbf{p}} + C(\dot{\mathbf{q}}, \mathbf{q})\dot{\mathbf{p}} + D\dot{\mathbf{q}} + g(\mathbf{q}) \quad (3)$$

ただし、 $\mathbf{p} \in \mathbf{R}^N$ は適当な制御量を表す。

ロボット・アームの幾何学的な形状は、D-H 記法などに基づく座標変換行列によって表されるため[16]、式 (2) (3) に示したロボット・アームの制御則も一般に行列の式として記述される。REDUCE や Mathematica などの数式処理システムを用いてこれら行列の式をスカラ式に展開すると、共に式 (4) に示すような多変数の積和式の形となる[17][18]。

$$f_n = \sum_{j \in J_n} T_j, \quad (n = 1, 2, \dots, N) \quad (4)$$

$$T_j = \pm a_k * \left(\prod_{m \in M_j} v_m * h(m, j) \right)$$

ただし、 a_k は定数パラメータ、 v_m は入力変数、 $h(m, j)$ は v_m の乗数である。また、制御則に含まれる三角関数は独立の入力変数と考える。 J_n は積項 T_j に対する添字集合の部分集合、 M_j は全入力変数 v_m に対する添字集合の部分集合である。

数式処理システムを用いると、三角関数の加法定理や三角恒等式を考慮したロボット制御則の簡単化を行うことも可能である[19]。しかし、本論文では次章において、ロボット制御則に含まれる三角関数の値はあらかじめ計算してあるものとして、式 (4) に示した制御則の計算時間を評価する。したがって、数式処理システムを用いて式 (4) を導出する際には、加法定理に基づく三角関数の簡単化は行わないものとする。

例えば、Fig.2 に示す2自由度のロボット・アームの第2関節に対して、式 (3) の制御則を導出すると式 (5) が得られる。

$$f_2 = \sum_{j \in J_2} T_j, \quad J_2 = \{1, 2, \dots, 7\} \quad (5)$$

$$\begin{aligned} T_1 &= b_3 * \ddot{p}_1, & T_2 &= L_1 * b_4 * c_2 * \ddot{p}_1, \\ T_3 &= b_3 * \ddot{p}_2, & T_4 &= L_1 * b_4 * s_2 * \dot{q}_1 * \dot{p}_1, \\ T_5 &= D_2 * \dot{q}_2, & T_6 &= -G * b_4 * s_1 * s_2, \\ T_7 &= G * b_4 * c_1 * c_2. \end{aligned}$$

ただし, c_n, s_n は三角関数 $\cos(q_n), \sin(q_n)$ を表すものとする. また, b_m は基底パラメータ [20], L_n は各リンクの長さ, D_n は摩擦係数, G は重力加速度を表す.

3. 計算の並列化問題

本論文では, 式 (4) に示したロボット・アームの制御則のうち, 任意に与えられた関節トルク $f_n (1 \leq n \leq N)$ の計算式に対して MIMD 分散メモリ型 [7] の並列計算機モデルによる並列処理を考える. すなわち, 並列計算機は処理能力の等しい P 個 ($P \geq 2$) のプロセッサから成るものとし, さらに, 各プロセッサは専用のメモリを持つものとする.

まず, ある任意の関節 n について式 (4) を次の式 (6) (7) に示すように分割する.

$$f_{np} = \sum_{j \in J_{np}} T_j, \quad (p = 1, 2, \dots, P) \quad (6)$$

ただし,

$$J_n = J_{n1} \cup \dots \cup J_{np} \cup \dots \cup J_{nP}, \\ J_{np} \neq \phi, \quad J_{np} \cap J_{nq} = \phi \quad (p \neq q).$$

$$f_n = \sum_{p=1}^P f_{np} \quad (7)$$

式 (6) に示した P 個の積和式の計算は, 並列化された P 個のプロセッサにより同時に行われる. 一方, 式 (7) の計算は, 式 (6) の計算が終了した後, 各プロセッサ間の結合網の状態などを配慮して, 適当なプロセッサによって行うものとする [21].

一般に, 式 (4) に示したロボット・アームの制御則は, 膨大な数の積項から成り, 各積項は多くの共通な因子を含んでいる. このことを踏まえて本論文では, 各プロセッサにおける逐次処理による計算の効率も考慮して, 式 (4) の計算を式 (6) に示したように積項単位で P 個のプロセッサに割り当てた後に, 式 (6) の各積和式から因数分解や共通部分式の括り出し等によって冗長な演算を削除する. この計算式の簡単化には, Breuer [22] が提案した因数分解のアルゴリズムを筆者らが改良した手法 [9] を用いるものとする.

例えば, 式 (5) に示した 2 自由度のロボット・アームの制御則に対しては, プロセッサの個数を $P = 2$ とすると, 式 (8) に示すような並列化と簡単化が可能である.

$$\begin{aligned} f_{21} &= T_1 + T_3 + T_4 + T_5 \\ &= L_1 * b_4 * s_2 * \dot{q}_1 * \ddot{p}_1 + b_3 * (\ddot{p}_2 + \ddot{p}_1) + D_2 * \dot{q}_2 \\ f_{22} &= T_2 + T_6 + T_7 \\ &= b_4 * (G * (-s_1 * s_2 + c_1 * c_2) + L_1 * c_2 * \ddot{p}_1) \\ f_2 &= f_{21} + f_{22} \end{aligned} \quad (8)$$

一般に, プロセッサ数 P に対して式 (4) の制御則に含まれる積項の数が十分に多いため, 並列処理においては, 式 (6) の計算時間が式 (7) の計算時間と各プロセッサ間の通信時間に比べて支配的となる. したがって, 式 (7) の計算時間と通信時間は無視できるものとする. また, 乗算器を内蔵した最近のプロ

セッサでは加減算と乗算の処理時間がほぼ等しいことから [8], 式 (6) に示した各計算式のプロセッサにおける処理時間は, 含まれる演算回数によって評価する. ただし, べき乗は乗算に展開し, 三角関数の値はあらかじめ計算してあるものとする.

ここで, 因数分解等により簡単化した積和式を $Fac(\cdot)$, 計算式に含まれる演算回数 (加減算と乗算の総数) を $Op(\cdot)$ によって表す. また, 積項の添字集合 J_n の P 個の空でない部分集合への任意の分割を $D(J_n; P)$ と表すと, 本論文で考えるロボット制御則の計算の並列化問題は以下のように定式化できる.

【並列化問題】

入力: 関節 n に対する式 (4) の積項 $T_j (j \in J_n)$ とプロセッサの個数 $P (2 \leq P \leq |J_n|)$, $|J_n|$ は積項数.

目的: 式 (9) の目的関数 $E(\cdot)$ の最小値と, それを与える積項の P 個のプロセッサへの配分方法 $D(J_n; P)$.

$$E(D(J_n; P)) = \max_{1 \leq p \leq P} \left\{ Op \left(Fac \left(\sum_{j \in J_{np}} T_j \right) \right) \right\} \quad (9)$$

ただし,

$$\begin{aligned} D(J_n; P) &= \{J_{n1}, \dots, J_{np}, \dots, J_{nP}\}, \\ J_n &= J_{n1} \cup \dots \cup J_{np} \cup \dots \cup J_{nP}, \\ J_{np} &\neq \phi, \quad J_{np} \cap J_{nq} = \phi \quad (p \neq q). \end{aligned}$$

である. $\square$

並列化問題に対する許容解の総数, すなわち, 与えられた積項の添字集合 J_n を, プロセッサの数である P 個の互いに素な空でない部分集合に分割する方法の総数は, 第 2 種スターリング数となり膨大である. さらに, NP 困難であることが知られている先行制約条件のないスケジューリング問題 [23] が, 上記の並列化問題に多項式時間変換可能であることから, この問題は NP 困難なクラスに属する極めて難しい組合せ最適化問題であることが証明できる [13].

4. 最適化アルゴリズムと評価

NP 困難な組合せ最適化問題の最も有力な解法の一つである分枝限定法 [11] に基づき, 並列化問題に対する最適化アルゴリズムを提案すると共に, 適用例による評価を行う.

4.1 最適化アルゴリズム

分枝限定法とは, 直接解くのが困難であるような複雑な組合せ最適化問題を, よりやさしい小規模な部分問題に分解し, それらをすべて解くことによって等価的に元の問題を解こうとするものである.

提案する最適化アルゴリズムでは, 部分問題において元の並列化問題で与えられた積項の部分集合 $\{T_j \mid j \in I_n \subseteq J_n\}$ を, Q 個 ($Q \leq P$) のプロセッサに配分して式 (10) の目的関数 $E(\cdot)$ の値を評価する. $I_n = J_n$ かつ $Q = P$ が成り立つとき, 添字集合 I_n の分割 $D(I_n; Q)$ と対応する $E(\cdot)$ の値は, 明らかに並列化問題の許容解である.

$$E(D(I_n; Q)) = \max_{1 \leq p \leq Q} \left\{ Op \left(Fac \left(\sum_{j \in I_{np}} T_j \right) \right) \right\} \quad (10)$$

ただし,

$$\begin{aligned} D(I_n; Q) &= \{I_{n1}, \dots, I_{np}, \dots, I_{nQ}\}, \\ I_n &= I_{n1} \cup \dots \cup I_{np} \cup \dots \cup I_{nQ}, \\ I_{np} &\neq \phi, \quad I_{np} \cap I_{nq} = \phi \quad (p \neq q). \end{aligned}$$

次に, 集合の分割アルゴリズム [24] を応用した部分問題の生成方法について説明する. これは, 与えられた積項の添字集合 J_n のすべての部分集合 $I_n \subseteq J_n$ に対して, それらの分割を重複なく, 見落としがないように順次発生させるものである.

提案する分枝操作では, 添字集合 I_n の任意の分割 (部分問題) $D(I_n; Q)$ から, 添字集合 $\hat{I}_n = I_n \cup \{j\}$ ($j \notin I_n, j \in J_n$) の以下に示す $(Q+1)$ 種類の異なる分割を派生させる.

$$D(I_n; Q) = \{I_{n1}, I_{n2}, \dots, I_{nQ}\} \quad (11)$$

であるとき,

$$\begin{aligned} D(\hat{I}_n; \hat{Q}) &= \{\{I_{n1} \cup \{j\}\}, I_{n2}, \dots, I_{nQ}\}, \\ D(\hat{I}_n; \hat{Q}) &= \{I_{n1}, \{I_{n2} \cup \{j\}\}, \dots, I_{nQ}\}, \\ &\vdots \\ D(\hat{I}_n; \hat{Q}) &= \{I_{n1}, I_{n2}, \dots, \{I_{nQ} \cup \{j\}\}\}, \\ D(\hat{I}_n; \hat{Q}) &= \{I_{n1}, I_{n2}, \dots, I_{nQ}, \{j\}\}. \end{aligned} \quad (12)$$

ただし, $j \notin I_n$, かつ, $j \in J_n$ である.

式 (12) では任意の分割を改めて $D(\hat{I}_n; \hat{Q})$ と表している. このとき, 式 (13) (14) に示す条件を共に満たす分割のみ新たな部分問題として有効である.

$$\hat{Q} \leq P \quad (13)$$

$$P - \hat{Q} \leq |J_n| - |\hat{I}_n| \quad (14)$$

ただし, $|\hat{I}_n|, |J_n|$ はおのおの $\hat{I}_n, J_n$ の要素数を表す.

式 (13) の意味するところは明らかである. また, 式 (14) が満たされないと, 最終的に積項がまったく配分されないプロセッサが生じてしまう. したがって, これらの条件を満たさない分割は許容解とは成りえず無条件に破棄することができる.

例えば, 提案する分枝操作にしたがって $J_n = \{1, 2, 3, 4\}$, $P = 3$ とした場合の部分問題をすべて生成すると Fig. 3 に示す分枝図が得られる. Fig. 3 において, 破線の枝は式 (13) (14) の条件を満たさないために除かれる分割を示している. Fig. 3 には冗長な部分問題が存在せず, 並列化問題に対して必要最小限の分枝図が構成されていることが分かる.

ところで, 目的関数 $E(\cdot)$ の値を求めるためには計算式の簡単化を行う必要があり, 生成されたすべての部分問題に対してこれを行うことは計算時間において不利である. したがって, 最適化アルゴリズムでは, 目的関数の下界値と上界値を利用して部分問題の評価の高速化を図っている.

ある積和式に含まれる異なる変数の総数を $Atom(\cdot)$ により表し, 式 (10) の目的関数 $E(\cdot)$ に対する下界値 $L(\cdot)$ と上界値 $U(\cdot)$ をおのおの以下のように定義する.

$$L(D(I_n; Q)) = \max_{1 \leq p \leq Q} \left\{ Atom \left(\sum_{j \in I_{np}} T_j \right) - 1 \right\} \quad (15)$$

Fig. 3 Branching tree

$$U(D(I_n; Q)) = \max_{1 \leq p \leq Q} \left\{ Op \left(\sum_{j \in I_{np}} T_j \right) \right\} \quad (16)$$

このとき, 簡単な数式の因数分解を例に考えれば分かるように, 明らかに式 (17) の関係が常に成り立つ.

$$L(D(I_n; Q)) \leq E(D(I_n; Q)) \leq U(D(I_n; Q)) \quad (17)$$

したがって, 暫定解に対する目的関数 $E(\cdot)$ の値を Z とするとき, 部分問題 $D(I_n; Q)$ が式 (18) を満たせば終端し, 式 (19) を満たせば新たな部分問題を派生させればよく, これらの場合, 共に $D(I_n; Q)$ に対して $E(\cdot)$ の値を求める必要がない.

$$Z \leq L(D(I_n; Q)) \quad (18)$$

$$U(D(I_n; Q)) < Z \quad (19)$$

提案する最適化アルゴリズムでは, Fig. 3 に示したような分枝図の各節点 (部分問題) を左回りに辿る深さ優先探索法 [11] を採用する. また, 分枝限定法においては, 優れた暫定解が早期に求まるほど探索すべき領域は縮小する. したがって, 先行制約条件がないスケジューリング問題に対する近似アルゴリズムとして知られる LPT 法 (Longest Processing Time) [23] を改良して, 筆者らが先に提案した並列化問題に対する近似アルゴリズム GCF/LPT 法 (Greatest Common Factor/LPT) [13] を用いて求めた近似解を暫定解の初期値とする.

活性な部分問題を保存するためにスタック S [25] を用いてアルゴリズムを記述すると以下になる. スタックが空になるとアルゴリズムの計算は終了し, その時点の暫定解 $D(J_n; P)$ が並列化問題の最適解である.

【最適化アルゴリズム】

$Z = E(D(J_n; P));$ // $D(J_n; P)$: solution with GCF/LPT.
 $I_n = \{1\}; Q = 1;$ push($D(I_n; Q), S$);
 while(not empty(S)){

```

 $D(I_n; Q) = \text{top}(\mathbf{S});$ 
if ( $U(D(I_n; Q)) \geq Z$ ) {
  if ( $L(D(I_n; Q)) < Z$ ) {
    if ( $E(D(I_n; Q)) < Z$ ) {
      if ( $|I_n| = |J_n|$ ) {
         $D(J_n; P) = \text{pop}(\mathbf{S});$   $Z = E(D(J_n; P));$ 
      } // renew incumbent
    } else pop( $\mathbf{S}$ ); // termination
  } else pop( $\mathbf{S}$ ); // termination
}
 $D(I_n; Q) = \text{pop}(\mathbf{S});$ 
make new  $D(\hat{I}_n; \hat{Q})$ 's from  $D(I_n; Q)$  as Eq. (12);
for each of the  $D(\hat{I}_n; \hat{Q})$ 's do {
  if ( $(\hat{Q} \leq P)$  and  $(P - \hat{Q} \leq |J_n| - |\hat{I}_n|)$ )
    push( $D(\hat{I}_n; \hat{Q}), \mathbf{S}$ );
} // branching operation
}
output  $D(J_n; P)$  and  $Z$ ; // optimal solution

```

4.2 適用例による評価

二つの回転関節と一つの直動関節を持つ極座標ロボット・アーム [16] の逆動力学計算において、第 2 関節トルクの計算式を数式処理システムを用いて式 (2) に示したような閉じた形で導出した。この計算式に対して最適化アルゴリズムを適用した結果を **Table 1** に示す。Table 1 の各列は、第 1 列目よりプロセッサ数 P 、GCF/LPT 法により求めた近似解に対する目的関数 $E(\cdot)$ の値 (暫定値の初期値)、最適解に対する目的関数 $E(\cdot)$ の値、最適化アルゴリズムにより生成された部分問題の総数が許容解の総数 (第 2 種スターリング数) に対して占める比率、最適化アルゴリズムの計算時間を示している。ただし、最適化アルゴリズムの計算時間には、GCF/LPT 法による暫定解の初期値の計算時間も含まれている。また、使用した計算機は Sun SPARC Station 20 Model 50、プログラムは g++ によりコンパイルした。

Table 1 の結果から、すべてのプロセッサ数 P において、従来の GCF/LPT 法により求まる近似解よりも優れた最適解が得られることが分かる。また、生成された部分問題の総数が許容解の総数に比べて非常に少ないことから、最適化アルゴリズムにおける探索効率が極めて高いことが確認できる。

この例題において用いたロボット制御則と、得られた最適解の具体的な一例を付録に示す。

同様に、三つの回転関節を持つ多関節ロボット・アーム [16] について、式 (3) に示したような線形化補償器の制御則を導出した。第 3 関節トルクの計算式に対して最適化アルゴリズムを適用した結果を **Table 2** に示す。Table 1、Table 2 の結果より、最適化アルゴリズムがロボット・アームの幾何学的な形状の違いや制御則の種類にかかわらず有効であることが分かる。

さらに、Table 1 に示した例題において、最適化アルゴリズムにおける上界値 $U(\cdot)$ 、下界値 $L(\cdot)$ 、目的関数 $E(\cdot)$ の各計算回数と暫定値 Z の更新回数を **Table 3** に示す。Table 3 より、提案した最適化アルゴリズムにおいては、式 (18) に示した下界値 $L(\cdot)$ を用いた部分問題の終端が、目的関数 $E(\cdot)$ の計算回

Table 1 Spherical robot-arm (RRP)

P	GCF	Opt	$D(\cdot; \cdot)$ [%]	time [sec]
2	21	14	3.732	69.85
3	15	10	4.393×10^{-2}	84.22
4	12	9	8.641×10^{-3}	332.07
5	11	8	1.570×10^{-3}	299.58
6	10	7	6.234×10^{-5}	18.67

Table 2 Articulated robot-arm (RRR)

P	GCF	Opt	$D(\cdot; \cdot)$ [%]	time [sec]
2	18	16	5.824	56.64
3	16	12	2.030×10^{-1}	150.24
4	12	10	9.854×10^{-3}	47.54
5	11	9	2.331×10^{-3}	18.28
6	9	8	1.939×10^{-4}	1.47

Table 3 Effect of search method

P	$U(\cdot)$	$L(\cdot)$	$E(\cdot)$	Z
2	1223	1213	1017	6
3	3137	3133	1589	5
4	14846	14842	5507	3
5	17212	17207	4665	3
6	1705	1704	388	3

数の削減に大きく貢献していることが確認できる。

5. 準最適化アルゴリズムと評価

大規模な並列化問題にも適用可能な準最適化アルゴリズムを提案すると共に、その有効性を評価する。

5.1 準最適化アルゴリズム

並列化問題は NP 困難であるので、問題の規模が大きくなると求解に要する時間は指数関数的に増加し、いわゆる組合せ論的爆発によって、前節で提案した最適化アルゴリズムのように分枝限定法を用いて探索の効率化を図った場合でも、厳密な最適解を求めることが困難となる。このように大規模な組合せ最適化問題に対する一般的な解法としては、ランダム法や欲張り法のような基本的な近似アルゴリズムのほか、アニーリング法、タブー探索法、遺伝アルゴリズムといったメタ戦略と呼ばれる新しい試みも成されている [26]。しかしながら、対象とする問題固有の知識をうまく活用することが近似解法の成功にとって最も重要なことである。

本論文では、式 (4) のように積和式で表されたロボット・アームの制御則においては、一般に各積項に含まれる乗算回数に大きなばらつきがあることに着目し、最適化アルゴリズムと近似アルゴリズム (GCF/LPT 法) [13] を組み合わせた以下のような準最適化アルゴリズムを提案する。

【準最適化アルゴリズム】

- (1) 与えられた積項の集合 $\{T_j \mid j \in J_n\}$ から含まれる乗算回数が多い順にいくつかの積項を取り出し、部分集合 $\{T_j \mid j \in \hat{J}_n \subseteq J_n\}$ を定義する。
- (2) 最適化アルゴリズムを用いて積項 T_j ($j \in \hat{J}_n$) を P 個のプロセッサに配分する。
- (3) GCF/LPT 法を用いて残りの積項 T_j ($j \in J_n - \hat{J}_n$) を P 個のプロセッサに配分する。 □

分枝限定法において計算を最後まで実行せずに途中で打ち切り、その時点での暫定解を近似解として利用するような単純な方法では、計算を打ち切る適切な時点と判断するのが難しい。

提案した準最適化アルゴリズムは、ロボット・アームの制御則の特性を考慮して問題の定義領域を分割し、2種類のアルゴリズムの適用範囲の比によって準最適解の精度と求解に要する計算時間を調節することが可能である。また、並列化問題の解に大きな影響を与える領域に対してのみ最適化アルゴリズムを適用しているため、以下でも示すように、アルゴリズムの計算時間に比べて得られる解の精度が高く効率が良い。

5.2 適用例による評価

最適化アルゴリズムとの比較において、提案した準最適化アルゴリズムを評価する。評価のための尺度として、得られた準最適解の精度 R_a ($R_a \geq 1$) と求解に要する計算時間の比 R_t ($R_t \leq 1$) を導入する。

$$R_a = \frac{\text{準最適解に対する目的関数 } E(\cdot) \text{ の値}}{\text{最適解に対する目的関数 } E(\cdot) \text{ の値}} \quad (20)$$

$$R_t = \frac{\text{準最適化アルゴリズムの計算時間}}{\text{最適化アルゴリズムの計算時間}} \quad (21)$$

ただし、最適化アルゴリズムの計算時間は、GCF/LPT 法による暫定解の初期値の計算時間も含む。

さらに、準最適化アルゴリズムにおいて、最適化アルゴリズムを適用する積項 T_j ($j \in \hat{J}_n$) が全積項 T_j ($j \in J_n$) に占める割合を X_r ($X_r = |\hat{J}_n|/|J_n|$) とする。 $X_r = 1$ ($\hat{J}_n = J_n$) であるとき、準最適化アルゴリズムは最適化アルゴリズムに等しく、明らかに次の関係が成り立つ。

$$R_a = R_t = 1, \quad (X_r = 1) \quad (22)$$

前節で用いた極座標ロボット・アームの逆動力計算の例題に対して準最適化アルゴリズムを適用した結果を Fig. 4 に示す。Fig. 4 は、プロセッサ数 P を変えて、パラメータ X_r に対する R_a と R_t の変化を示している。 X_r が 1 に近づくのに伴って R_a と R_t が 1 に漸近しており、最適化アルゴリズムを適用する積項数を増やすほど、アルゴリズムの計算時間は長くなるが得られる解の精度は向上することが分かる。これは直観的にも予想しうる結果である。

ところで、Fig. 4 では $P = 2$ の場合、 X_r を増やすことにより得られる準最適解の精度 R_a が一時的に悪化している。この原因は、準最適化アルゴリズムが、分解法 [27] のように、原問題の部分問題について単独に最適化を行うためであると考えられる。すなわち、部分問題によっては、最適化アルゴリズムによる最適解が局所的なものとなり、大域的には不適切な解とな

Fig. 4 Effect of quasi-optimization algorithm

る可能性がある。しかし、提案した準最適化アルゴリズムにおいては、ロボット制御則の特性を考慮して原問題を適切に分割している。したがって、Fig. 4 の結果からも分かるように、このようなケースは希であると言える。

Fig. 4 において特に注目すべき点は、 X_r の増加に伴い、 R_t よりも R_a の方が急激に 1 に収束していることである。これはわずかなアルゴリズムの計算時間の増加によって、求まる解の精度が大幅に改良される可能性があることを意味しており、提案した準最適化アルゴリズムの妥当性を示すものである。

6. お わ り に

本論文では、ロボット制御則の計算の並列化問題に対して、最適解を厳密に求めることができる最適化アルゴリズムと、大規模な問題に対しても適用可能な準最適化アルゴリズムを提案した。対象とした並列化問題は NP 困難であるが、NP 完全の理論は NP 困難な問題に対して最悪の場合に問題規模の指数関数程度の計算量を要することを推測しているだけであって、すべての問題例に対して困難性を指摘しているものではない。また、繰り返し実行されるロボット制御則の並列処理においては、ある程度の計算時間を要しても最適解に近い解が望まれる。この意味において、今回、最適化アルゴリズムと準最適化アルゴリズムを提案した意義は大きい。

本論文において考えた並列処理の手法は、積和式で表される任意の計算式に対して適用することができ、ロボット制御則の種類に依存しない。また、並列計算機の各プロセッサにおける逐次処理の計算効率を高めることによって全体の処理時間を短縮するものであり、プロセッサ間の結合方法などハードウェアに大きく影響されることがなく、多くの分散メモリ型並列計算機において有効であると考えられる。したがって、提案した並列化アルゴリズムに基づき、数式処理システムを用いて導出したロボット・アームの制御則から、特定の並列計算機に対するプログラムを生成するような自動並列化コンパイラの開発に応用することも可能である。

プロセッサ間の通信コストやロボットの制御則などによっては、あるプロセッサの計算結果を他のプロセッサが利用した方が有利となるような場合も考えられる。今後の課題としては、

これらとの比較において本論文における並列処理手法の有効性と限界について検討することがある。

なお、本研究は、文部省科学研究費助成金、一般研究(C) 07650460 の助成を受けたものである。

参考文献

- [1] 古田：“ロボットの動的制御”，計測と制御，vol.30, no.5, pp.376-382, 1991.
- [2] J.Y.S. Luh and C.S. Lin: “Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator,” IEEE Trans. on Systems, Man, and Cybernetics, vol.SCM-12, no.2, pp.214-234, 1982.
- [3] H. Kasahara and S. Narita: “Parallel Processing of Robot-Arm Control Computation on a Multimicroprocessor System,” IEEE J. of Robotics and Automation, vol.RA-1, no.2, pp.104-113, 1985.
- [4] W.S. Wang and C.H. Liu: “Implementation and Experimental Study of a Multiprocessor System for Real-Time Model-Based Robot Motion Control,” IEEE Trans. Industrial Electronics, vol.41, no.2, pp.163-172, 1994.
- [5] D.J. Kuck, Y. Muraoka and S.C. Chen: “On the Number of Operations Simultaneously Executable in Fortran-Like Programs and Their Resulting Speedup,” IEEE Trans. Computers, vol.C-21, no.12, pp.1293-1310, 1972.
- [6] S. Lakshmivarahan and S.K. Dhall: Analysis and Design of Parallel Algorithms, Part 2, McGraw-Hill, 1990.
- [7] 安浦：“並列計算機と並列計算モデル”，情報処理，vol.33, no.9, pp.1024-1032, 1992.
- [8] 丸田，辻井：シグナルプロセッサとその応用，昭晃堂，1988.
- [9] 田川，太田，藤原，羽根田：“ロボットの逆動力学計算に対する最適化システム”，計測自動制御学会論文集，vol.29, no.2, pp.220-226, 1993.
- [10] R.J. Fateman: “Optimal Code for Serial and Parallel Computation,” Communications of the ACM, vol.12, no.12, pp.694-695, 1969.
- [11] 茨木俊秀：組合せ最適化—分枝限定法を中心として—，産業図書，1983.
- [12] 茨木俊秀：アルゴリズムとデータ構造，6章，昭晃堂，1991.
- [13] 田川，神吉，太田，羽根田：“ロボットの動的制御における計算の並列化アルゴリズムとオブジェクト指向による実現”，システム制御情報学会論文誌，vol.7, no.12, pp.505-511, 1994.
- [14] J.Y.S. Luh, M.W. Waiker, and R.P.C. Paul: “On-Line Computational Scheme for Mechanical Manipulators,” Trans. ASME, vol.102, no.2, pp.69-76, 1980.
- [15] J.J.E. Slotine and W. Li: “On the Adaptive Control of Robot Manipulators,” Int. J. Robotics Research, vol.6, no.3, pp.49-59, 1987.
- [16] M.W. Spong and M. Vidyasagar: Robot Dynamics and Control. John Wiley & Sons, 1986.
- [17] J. Koplik and M.C. Leu: “Computer Generation of Robot Dynamics Equations and the Related Issues,” J. Robotic Systems, vol.3, no.3, pp.301-319, 1986.
- [18] J.F. Nethery and M.W. Spong: “Robotica: A Mathematica Package for Robot Analysis,” IEEE Robotics & Automation Magazine, vol.1, no.1, pp.13-20, 1994.
- [19] 齊藤，梅野，阿部，桑原，前川：“REDUCEを用いたロボットアームの運動学および動力学方程式自動生成システム”，電気学会論文誌 C, vol.109, no.2, pp.89-96, 1989.
- [20] H. Mayeda, K. Yoshida and K. Osuka: “Base Parameters of Manipulator Dynamic Models,” IEEE Trans. Robotics and Automation, vol.6, no.3, pp.312-321, 1990.
- [21] K.B. Irani and K.W. Chen: “Minimization of Interprocessor Communication for Parallel Computation,” IEEE Trans. Computers, vol.C-31, no.11, pp.1067-1075, 1982.
- [22] M.A. Breuer: “Generation of Optimal Code for Expressions via Factorization,” Communications of the ACM, vol.12, no.6, pp.333-340, 1969.
- [23] R.L. Graham: “Bounds on Multiprocessing Timing Anomalies,” SIAM J. Applied Mathematics, vol.17, no.2, pp.416-429, 1969.
- [24] 仙波一郎：組合せアルゴリズム，5章，サイエンス社，1991.
- [25] A.V. Aho, J.E. Hopcroft and J.D. Ullman: Data Structures and Algorithm, Chapter 2. Addison-Wesley, 1983.
- [26] 茨木：“組合せ最適化とスケジューリング問題：新解法とその動向”，計測と制御，vol.34, no.5, pp.340-346, 1995.
- [27] 西川，玉置，牧：“ジョブショップ・スケジューリング問題に対する分解法”，計測自動制御学会論文集，vol.27, no.5, pp.607-613, 1991.

付録：最適解の具体例

極座標ロボット・アームの逆動力学計算において，第2関節トルクの計算式は以下のように与えられる。

$$f_2 = \sum_{j \in J_2} T_j, \quad J_2 = \{1, 2, \dots, 16\} \quad (23)$$

$$\begin{aligned} T_1 &= a_1 * \dot{q}_1 * *2 * c_2 * s_2, \\ T_2 &= a_2 * \ddot{q}_2, \\ T_3 &= -a_3 * \dot{q}_1 * *2 * c_2 * s_2, \\ T_4 &= a_4 * \dot{q}_1 * *2 * c_2 * s_2, \\ T_5 &= a_5 * \ddot{q}_2, \\ T_6 &= -a_6 * \dot{q}_1 * *2 * c_2 * s_2, \\ T_7 &= a_7 * \ddot{q}_2 * \dot{q}_3 * q_3, \\ T_8 &= -a_8 * \dot{q}_1 * *2 * q_3 * *2 * c_2 * s_2, \\ T_9 &= a_8 * \ddot{q}_2 * q_3 * *2, \\ T_{10} &= -a_{10} * \dot{q}_1 * q_3 * c_2, \\ T_{11} &= a_{11} * \ddot{q}_2 * \dot{q}_3, \\ T_{12} &= -a_{11} * \dot{q}_1 * *2 * q_3 * c_2 * s_2, \\ T_{13} &= a_{11} * \ddot{q}_2 * q_3, \\ T_{14} &= -a_{14} * \ddot{q}_1 * c_2, \\ T_{15} &= a_{15} * q_3 * c_2 * s_1, \\ T_{16} &= a_{16} * c_2 * s_1. \end{aligned}$$

ただし， a_k は重力加速度や各リンクの慣性モーメント等をまとめた定数パラメータ， q_n ($n = 1, 2, 3$.) は各関節の変位， c_n ， s_n は三角関数 $\cos(q_n)$ ， $\sin(q_n)$ を表すものとする。

並列化するプロセッサの個数を $P = 3$ として最適化アルゴリズムを適用すると，各積項の配分方法が以下のように求まり，目的関数の最小値は $E = 10$ となる。

$$\begin{aligned} J_{21} &= \{1, 3, 4, 6, 8, 12\}, \\ J_{22} &= \{2, 5, 7, 9, 11, 13\}, \\ J_{23} &= \{10, 14, 15, 16\}. \end{aligned} \quad (24)$$

ここで，式 (24) に示した積項の配分方法によって決まる三つの積和式をそれぞれ簡単化すると，各プロセッサごとに行われる計算処理は以下になる。ただし，筆者らがさきに提案した数式の簡単化アルゴリズムに基づいて，自動的に導入される中間変数を v_m としている [9]。本文中の式 (8) に示した2自由度のロボット・アームの例題では，簡単のために，これらの中間変数は省略してある。

1番目のプロセッサでは， f_{21} が次のように計算される。

$$\begin{aligned} v_{10} &:= a_1 - a_3 + a_4 - a_6; \\ v_1 &:= -a_{11} - a_8 * q_3; \end{aligned}$$

$$\begin{aligned} v_2 &:= q_3 * v_1 + v_{10}; \\ f_{21} &:= \dot{q}_1 * \dot{q}_1 * c_2 * s_2 * v_2; \end{aligned} \quad (25)$$

ただし, v_{10} は定数値となることに注意されたい.
2 番目のプロセッサでは, f_{22} が次のように計算される.

$$\begin{aligned} v_{10} &:= a_2 + a_5; \\ v_1 &:= a_7 * q_3 + a_{11}; \\ v_2 &:= a_{11} + a_8 * q_3; \\ v_3 &:= q_3 * v_2 + v_{10}; \\ f_{22} &:= \dot{q}_2 * v_3 + \dot{q}_2 * \dot{q}_3 * v_1; \end{aligned} \quad (26)$$

ただし, v_{10} は定数値となることに注意されたい.
3 番目のプロセッサでは, f_{23} が次のように計算される.

$$\begin{aligned} v_1 &:= a_{15} * s_1 - a_{10} * \dot{q}_1; \\ v_2 &:= a_{16} * s_1 - a_{14} * \dot{q}_1 + q_3 * v_1; \\ f_{23} &:= c_2 * v_2; \end{aligned} \quad (27)$$

式 (25) (26) (27) に示した各計算式に含まれる演算回数は, オフラインで行うことができる定数値 v_{10} の計算を除いて, それぞれ 8 回, 10 回, 9 回であり, $E = 10$ となることが確認できる.

田川聖治 (Kiyoharu Tagawa)

1962 年 7 月 20 日生. 1991 年神戸大学大学院修士課程修了. 1993 年より神戸大学工学部電気電子工学科助手. ロボットの計算機制御に関する研究に従事. 計測自動制御学会, システム制御情報学会, 電気学会, IEEE の会員. (日本ロボット学会正会員)

神吉良英 (Yoshihide Kanki)

1971 年 2 月 9 日生. 1995 年神戸大学大学院修士課程修了. 同年, 株式会社島津製作所に入社. 在学中, ロボット制御則の並列処理に関する研究に従事. 電子情報通信学会の会員.

羽根田博正 (Hiromasa Haneda)

1939 年 8 月 10 日生. 1972 年カリフォルニア大学 (パークレイ校) 大学院博士課程修了. 1982 年より神戸大学工学部電子工学科教授. 計算機援用設計, 解析に関する研究に従事. Ph.D. 電子情報通信学会, 電気学会, 情報処理学会, システム制御情報学会, IEEE (フェロー), ACM 等の会員.

福居毅至 (Tuyoshi Fukui)

1971 年 9 月 5 日生. 1995 年神戸大学工学部電気工学科卒業. 同年, 同大学院博士前期課程に入学. 現在, 分枝限定法に基づくスケジューリング手法に関する研究に従事.

太田有三 (Yuzo Ohta)

1949 年 11 月 10 日生. 1977 年大阪大学大学院博士課程修了. 1987 年より神戸大学工学部電子工学科助教授. 計算機援用システム解析設計に関する研究に従事. 工学博士. 計測自動制御学会, システム制御情報学会, 電子情報通信学会, 電気学会, 情報処理学会, IEEE 等の会員.