設計変更における効率的組立順序探索アルゴリズム*

村 山 長** 大 場 史 憲**

An Efficient Algorithm for Generating Assembly Sequences in Modification Design

Takeshi Murayama and Fuminori Oba

It is important to generate and evaluate assembly sequences in product design stages for reducing manufacturing costs. This paper proposes a method for generating the assembly sequences efficiently. In the method, the problems of generating the assembly sequences are decomposed into subproblems, and the decompositions are represented as AND/OR graphs. To generate the sequences, an algorithm for searching the AND/OR graphs, which are reduced by using heuristics, is developed. This algorithm is efficient especially to generate the assembly sequences for the design objects got by modifying the original ones whose assembly sequences have ever been generated. The efficient generation of assembly sequences is achieved by using the assembly sequences of the original design objects. Based on the method, a system is developed, and an example is given by using the system to demonstrate the effectiveness of the method.

Key words : assembly sequence generation, modification design, problem decomposition, AND/OR graph, assembly, search, AO* algorithm, space reduction

1. 緒 言

最近、組立可能性の検証や組立計画を目的として、コンピュータを用いて組立順序を生成する研究が行われている^{1)~6)}。これらの研究では組立順序の算出時に、部品や部分組立品の組付け（または、分解）の可能性の検証が行われる。製品の構成部品数が多いほどこの検証回数は多くなるが、この検証に要する計算量が大きい[†]ため、それに伴って組立順序の生成に要する計算時間も非常に大きくなる。このため、この検証回数を減少させ、組立順序の算出を効率化する研究が行われている。例えば、伊藤らはヒューリスティクスを用いて効率よく組立順序を求めている³⁾。Leeは部分組立品を抽出することにより、不必要的検証を行う回数を減少させ、組立順序の算出を効率化している⁴⁾。また、安定性を評価することにより、効率化する研究も行われている⁵⁾⁶⁾。

一方、製品設計では、修正設計や改善のための設計変更など、設計対象の部分的変更がよく行われる。本論文では、このような変更された設計対象の組立順序を求める場合に、設計変更前の設計対象の組立順序情報を有効に利用することにより、組立順序の算出を効率化する方法を提案する。ここでは、組立順序算出のためのグラフ探索方法に焦点をあてる。ただし、既存のいくつかの研究と同様に、組立と分解の可逆性を仮定し、組立手段を限定しない。

2. 組立順序生成問題の分解

提案する方法では、組立順序の算出問題を部分問題に分解して処理する。そして、設計変更された設計対象の組立順序を求

める場合、設計変更前の設計対象の組立順序生成時に処理した部分問題の解を利用することにより、組立順序の算出を効率化することを考える。ただし、ここでは二つの物体の組付けを繰り返すことにより組立を行うものとして、問題を分解する。

ある物体 A_i （組立品または部分組立品）の組立順序を求める問題 S_i は、以下の三つの部分問題に分解できる。

- (1) A_i を構成する部品群を二つの部品群 A_j, A_k に分け、それらのうちの一方を他方に組付けられるかを調べる部分問題 $C_{j,k}$
- (2) A_j の組立順序を求める部分問題 S_j
- (3) A_k の組立順序を求める部分問題 S_k

この三つの部分問題がすべて解をもてば、その解を統合することにより、問題 S_i の一つの解（物体 A_i の組立順序）が求まる。ただし、物体 A_i の分割方法は複数存在するため、それに対応して問題の分解方法も複数存在する。また、部分問題 S_j, S_k は、

Fig.1 Decomposition of problem of generating assembly sequences

* 原稿受付 平成6年1月4日

** 正会員 広島大学工学部（東広島市鏡山1-4-1）

† 例えば、内山らの研究⁷⁾では、1個の部品の分解可能性の検証に約10分の計算時間を要している。

問題 S_i と同様に、それぞれより小さい部分問題に分解できる。また、部分問題 $C_{j,k}^i$ も、さらに小さい部分問題（組付け時の衝突チェックを行う問題など）に分解できる。

このような組立順序を求める問題の分解は図1のようにAND/ORグラフ⁸⁾で表現できる。このグラフで、各節点は、問題、部分問題、または、終端節点（1個の部品 A_m に対応する節点） E_m を表す。また、コネクタは上述のような部分問題間のAND関係を表すものであり、このコネクタでつながった三つの子節点が表す部分問題がすべて解をもてば、その親節点の問題は解をもつ。たとえば、コネクタ CO_1 により、「ある設計対象 A_1 の組立順序を求める」という問題 S_1 は、二つの部分組立品 A_2, A_3 のうち一方を他方に組付けられるかを調べる部分問題 $C_{2,3}^1$ と、 A_2 の組立順序を求める部分問題 S_2 および A_3 の同様の部分問題 S_3 に分解され、部分問題 $C_{2,3}^1, S_2, S_3$ の解を統合することにより、問題 S_1 の一つの解が求まる。この問題 S_1 は複数のコネクタとOR関係として接続されているが、これは問題 S_1 の分解方法が複数あることを表す。

このようなAND/ORグラフを探索し、問題 S_i から終端節点まで解をもつ節点のパスを求めることによって解グラフが得られる。このような各解グラフのトップダウンパスが設計対象 A_1 の一つの分解順序を表す。また、その逆が組立順序となる。もし、解グラフが得られなければ、その設計対象は組立不可能である。

このグラフの探索中に、 A_j と A_k との組付け可能性を調べる問題 $C_{j,k}^i$ は、組立と分解の可逆性の仮定に基づいて、 A_j （または A_k ）の分解可能性を調べることによって解かれるが、その処理方法は既に報告しているので⁹⁾、ここでは、説明を省略する。なお、この処理方法では、ある物体の分解時にはその物体だけを移動させ、他の部品や部品群を移動させないものとして、分解可能性を検証している。このため、ある物体の分解可能性を検証する時に、他の分解可能性の検証問題における他の物体の移動を考慮に入れる必要はなく、各分解可能性の検証問題は独立に処理できる。

組立順序の算出問題をこのような部分問題に分解することにより、設計変更された設計対象の組立順序を求める場合、設計変更前の設計対象のAND/ORグラフに存在する部分問題は新たに処理する必要はなく、それ以外の新たな部分問題のみを処理すればよいことになる。

3. 探索空間の縮小化

3.1 問題の分解とカットセット

AND/ORグラフにおいて問題 S_i の分解は、図2のように、物体 A_i （組立品または部分組立品）を部品接続グラフで表し、これを二つの部品群に分割する初等的カットセットを求めることによって行える。図2では、初等的カットセット $\{d, f\}, \{c, f, h\}$ により、問題 S_i が部分問題に分解されることが示されている。

各問題 S_i に対して、すべての初等的カットセットを求めてAND/ORグラフを作成すれば、すべての解グラフを求めることができるが、構成部品数の多い設計対象では、AND/ORグラフの規模が大きくなり、探索に要する時間が膨大になる。

そこで、ここでは探索空間を縮小するため、組付けに関するヒューリスティックな優先関係に基づいて初等的カットセットをあらかじめ設定された数だけ求め、それに対するコネクタと子節点を生成して解グラフを探索し、その探索途中で、組立順序（解グラフ）が得られないことが判明すれば、さらにコネク

(a) Part-connectivity graph and cut sets

(b) Decomposition of problem by using cut sets

Fig.2 An example of problem decomposition by using cut sets

タと子節点を設定数だけ追加し、探索を続ける方法をとる。

3.2 優先関係グラフ

ここでは組付けの優先関係を図3のような有向グラフ（優先関係グラフ）で表す。ここで、節点は部品間の接続関係を、枝はそれらの優先関係を表し、枝の終点の接続関係が始点の接続関係に優先する。すなわち、図3のレベル値の小さいものほど優先関係が高いことを示す。製品の分解時は、この優先関係の低い部品間の接続ほど早く解除される可能性が高くなる。

図1や図2(b)のように、問題 S_i のコネクタは、問題の分解を表すと同時に、物体 A_i の分解も表す。このため、優先関係の低い接続から構成される初等的カットセットは、現実に物体を分解する可能性が高く、 S_i の解を得るのに有効なコネクタと子節点を生成すると考えられる。例えば、図3では、接続 $\{c, d, f, h\}$ が最も優先関係が低いいため、この接続により構成されるカットセット（図2(a)）によって生成されるコネクタと子節点（図2(b)）により、 S_1 の解が得られる可能性が高い。

このような優先関係を、ここでは、

- (1) 部品の種類に関する情報、
- (2) 部品間の接続の種類に関する情報、
- (3) 製品の機能的な階層関係に関する情報、

に対して、ヒューリスティックなルールを適用することにより算出する。図4, 5, 6にルールによる算出例を示す。

図4は、「締結部品（図4ではピン③）によって複数の部品（①, ②）が固定されている場合、その固定される部品（①, ②）間の組付けが締結部品の組付けより先に行われる」というルールが適用された例である。図5は、「ある部品（①）が複数の部品（②, ③）と接続関係を持ち、その接続が固定接続と運動拘束（一方の部品が運動可能で、その運動を他方の部品が

Fig.3 Precedence graph

Fig. 4 An example of precedence graph generation based on parts' classes

Fig. 5 An example of precedence graph generation based on connective relations' classes

Fig. 6 An example of precedence graph generation based on functional hierarchy

Fig. 7 An example of fixation by bolt and nut

ガイドしている関係)である場合、固定接続関係にある部品(②)との組付けが、運動拘束関係にある部品(③)との組付けより先に行われる」というルールが適用された例である。また、図6は、製品の機能的な階層関係を基にした優先関係の生成例であり、「機能ユニットの組立を機能ユニット間の組付けより先に行う」というルールに基づいている。この他、以下のルールを現在用いているが、後述するシステムでは、ルールの追加、修正も可能である。

- (1) カバーが存在する場合は、カバーとそれを固定する締結部品の組付けは他部品の組付けより後に行う。
- (2) 運動可能な部品群が存在する場合は、その部品群を構成する部品間の組付けをしてから、部品群を組付ける。

- (3) 運動可能な部品群の運動範囲を制限するために付けられた部品群は、その運動可能な部品群を組付けた後に組付ける。
- (4) 図7のようにボルト、ナット、ワッシャにより固定されている場合は、ワッシャ①、ボルト②、ワッシャ③、ナット④の順に組付ける。ただし、これらのうち一部の部品がぬけていても、この順序は同じである。

このような各ルールによって得られた優先関係を合成し、設計対象に対する優先関係グラフを得る。得られた優先関係グラフには、優先関係の矛盾を表すループが存在する場合がある。このため、各ルールに対してあらかじめ重要度を与えておき、ループが存在する場合は、重要度の最も低いものを削除することにより、ループを解除する。図4, 5, 6に示した例の中では、図6で適用したルールの重要度が最も低く設定されている。

3.3 優先関係グラフによる問題の分解

優先関係グラフを用いた問題 S_i の分解(設定数のコネクタと子節点の生成)は以下のプロセスで行われる。

Step1: 問題 S_i と対応する優先関係グラフを求める。

Step2: 優先関係グラフを図3のように階層化し、最もレベル値の大きい節点の集合をPとする。

Step3: Pの部分集合から構成される初等的カットセットをすべて求め、そのカットセットの集合をCSとする。ただし、問題 S_i を親節点とするコネクタが既にいくつか存在する場合は、そのコネクタに対応する初等的カットセットをCSから取り除く。また、CSに含まれるカットセットの数があらかじめ設定された数より少ない場合は、次のレベルの節点をPに追加して、CSを求めなおす。CSに含まれるカットセットの数が設定された数より多い場合は、設定された数だけ任意に選択する。

Step4: 求めた初等的カットセットを基に、問題 S_i を分解し、設定数のコネクタと子節点を生成する。

4. 組立順序の算出方法

4.1 探索アルゴリズム

組立順序の算出のため、AO*アルゴリズム⁸⁾を改良した解グラフの探索アルゴリズムを開発した。これは、上述の設定数のコネクタと子節点の生成と、組付け可能性の検証問題 C_{ijk}^{jk} の処理とを繰り返すことにより、図8のようにAND/ORグラフを順次展開し、組立順序を探索するものである。

このアルゴリズムでは、図8のように、AND/ORグラフの展開の各時点で、各節点にコストが与えられる。節点 C_{ijk}^{jk} は物体 A_i が物体 A_j と A_k の組付けにより生成される場合の組付け可能性の検証問題を表すが、その問題の処理の結果、組付け不可能と判定された場合は、そのコスト $q(C_{ijk}^{jk})$ を ∞ とおき、可能な場合は、この検証問題の処理において得られる組付け作業の複雑さの評価値がコストとして与えられる。その評価値算出方法は、既に文献9)で報告しているのでここでは省略する。

一方、節点 S_i のコスト $q(S_i)$ は、探索のある時点における、 S_i に対応する物体 A_i の組立順序の評価値の下界値を表す。すなわち、その時点までに展開されたAND/ORグラフにおいて、 S_i が一つのコネクタで子節点の組 (C_{ijk}^{jk}, S_j, S_k) と結合されている時(A_i が A_j と A_k との組付けにより生成される時)、それらのコストの和 $q(C_{ijk}^{jk}) + q(S_j) + q(S_k)$ が $q(S_i)$ となる。ただし、コストが ∞ の子節点があれば、 $q(S_i)$ も ∞ となる。また、 S_i を親節点とするコネクタが複数あれば、コネクタごとに求めた子節点のコスト

Fig. 8 AND/OR graph search by the proposed algorithm

の和の中で最小のものが $q(S_i)$ となる。この時、そのコストの和が最小のコネクタに印↓が付けられる。例えば、図 8(c)の S_1 では、コネクタ1,2に対する子節点のコストの和がそれぞれ、 $3+0+8=11$ 、 $25+0+0=25$ であるので、その小さい方(11)が S_1 のコストとなり、コネクタ1に印↓が付けられる。また、 S_1 が、 S_7 、 S_8 のように、その時点でのAND/ORグラフにおいて、まだ子節点への展開がなされていない時、上記のようなコストが定まらないので、その下界値を $q(S_i)=0$ と置く。また、 S_i が組立品でなく単一部品に対応する時、その節点を E_i と記述し、そのコストを $q(E_i)=0$ と置く。

このようにして、探索のある時点での各節点 S_i のコストは、その時点でのAND/ORグラフの最下位の節点から順に求められ、グラフの展開に応じてその値が更新され、また、それに伴い、印↓が付け換わる。各時点において、この印↓を S_i から辿って得られる部分グラフ G' は、 S_i にコストの下界値を与えるものであり、その時点における解グラフの候補となる。

以下に、解グラフの探索アルゴリズムを示す。

Step1 (初期設定): 出発節点 S_1 (設計対象 A_1 の組立順序を求める問題)のみからなるグラフ G と解グラフの候補 G' を作り、 $q(S_1)=0$ と置く(図 8(a))。ただし、 G はその時点で展開されているAND/ORグラフを表し、後述の手続きでこれにコネクタと節点が順次追加される。

Step2 (グラフの展開): G' の展開可能な節点 S_i の中でコストが最小のものを、次に展開すべき節点 S_a とする。節点 S_a を親節点とし、3.3節の手順で、あらかじめ設定された数のコネクタと子節点を生成する(図 8の例ではコネクタの数は2個)。ここで、節点 S_a のすべての初等的カットセットに対応するコネクタと子節点が生成された時、これを意味する

ラベルall-generatedを S_a につける。

Step3 (子節点のコストの算出): 生成された子節点に前述の方法でコストを与える。すなわち、子節点のうち $C_{i,j,k}$ には組付け作業の評価値を、 S_i, E_i には0を、コストとして与える。この時、組付け可能性の検証問題 $C_{i,j,k}$ については、そのコスト算出のため、既に提案した手法に従ってさらに小さい部分問題(衝突チェックなど)に分解して処理するが、その結果、この問題 $C_{i,j,k}$ が解をもつ場合(組付け可能と判定された場合)はsolvedのラベルをつけ、解をもたない場合(組付け不可能と判定された場合)はunsolvableのラベルをつける。また、節点 E_i にも便宜上solvedのラベルをつける。

Step4 (上位節点のコストの更新): Step3で求めた子節点のコストを用いて、それより上位の各節点 S_i のコストを下から順に前述の方法で再計算する。例えば、図 8(c)では新たに生成された子節点が $C_{6,7}, S_6, S_7, C_{5,8}, S_8$ であるため、 S_3, S_1 の順にコストが再計算される。再計算によって得られたコストが ∞ でない場合は、前述のように S_i を親節点とするコネクタの一つに印↓が付けられる。もし、この印↓をつけたコネクタの子節点がすべてsolvedであれば、その節点 S_i が解けた(その節点 S_i が対象とする物体 A_i の組立順序が得られた)ことになるので、それにsolvedのラベルをつける。例えば、図 8(c)で、もし、 S_7 がsolvedになれば、 S_3 は解けたことになるため、 S_3 にsolvedのラベルが付けられる。この時、 S_3 の解(S_3 が対象とする物体 A_3 の組立順序)は、 S_3 から発する印↓の付いたコネクタを辿ることによって得られる。この組立順序は S_3 にコストの下界値を与える最もよいものを表す。

再計算によりある節点 S_i のコストが ∞ となり、その節点にall-generatedのラベルがついている場合は、その節点 S_i を親節点とするコネクタは既にすべて生成されており、しかも、どのコネクタによってもその節点の解は得られない。このため、その節点にはunsolvableのラベルをつける。また、節点 S_i のコストが ∞ であっても、all-generatedのラベルがついていない場合は、その節点 S_i は未生成のコネクタによって解が与えられる可能性がある。このため、その節点 S_i のコストを0に置き換える。例えば、図 8(c)で、もし S_7 がunsolvableになり、コネクタ3, 4では S_3 の解が得られないことが分かった場合は、コストの再計算により S_3 のコストが ∞ になるが、 S_3 にall-generatedのラベルが付いていないので、 S_3 のコストは0に書き換えられる。

Step5 (グラフの更新): グラフ G にStep2で生成したコネクタと子節点を加え、 G を更新する。また、 S_i から発する印↓の付いたコネクタをたどることによって、新しい解グラフの候補 G' を求める。

Step6 (終了判定): 以下の手続きで探索を続けるかを判定する。

6-1: 出発節点 S_1 にsolvedのラベルが付けられた場合は、問題 S_1 の解(設計対象 A_1 の組立順序)が得られたことになるため、探索を終了する。組立順序は、Step5で求めた G' として表現される。

6-2: S_1 にunsolvableのラベルが付けられた場合は、その設計対象の組立は不可能であることを意味するため、探索を終了する。

6-3: solved, unsolvableのいずれのラベルも S_1 に付けられていない場合は、Step2に戻り G' をさらに展開する。ただし、既にラベルが付けられた節点は展開可能な節点か

ら除外する。

AO*アルゴリズムでは葉節点が順次展開されるが、本手法では、Step2で中間節点もコスト値によっては展開され、子節点が追加される。これは、その時点までに生成された節点では解が得られない場合に起こる。このため、組立不可能な設計対象に対しては、出発節点 S_1 にunsolvableのラベルがつけられるまで、中間節点にコネクタと子節点が追加され、出発節点に対してはすべてのコネクタと子節点が生成される。これは、組立不可能な場合は探索空間が縮小されないが、組立可能性は必ず検証できることを意味する。なお、このStep2で生成される子節点の中で、グラフGの中に既に存在しているものがあれば、それを既存の節点と等置するものとする(図8(c)の S_5)。

4.2 設計変更前のAND/ORグラフの利用

既に組立順序が得られている設計対象を設計変更し、再び、組立順序を求める場合は、設計変更前の設計対象のAND/ORグラフ G^* を有効に利用する。これは、Step2でグラフが展開され、子節点 S_i が新たに生成された時に、その子節点が表す問題と同じものが G^* にあれば、図9のように G^* におけるその節点(この図では S_3^*)より下位の部分グラフSGを探索中のグラフGにコピーすることによって行える。ここで、問題が同じとは、問題の種類および問題が対象とする物体の構成部品がすべて同じで、その構成部品に設計変更がなされていない場合をいう。

また、Step3で組付け可能性を検証する問題 $C_{i,k}^*$ を処理する場合も、この問題の部分問題について、同様に設計変更前に同じ問題を処理していればその解を利用する。ただし、組付け可能性を検証する問題 $C_{i,k}^*$ から分解されるすべての部分問題(組付け方向を決定する問題、衝突をチェックする問題、移動距離を求める問題など)に対してこのような処理を行うと、かえってグラフの探索に時間がかかる可能性がある。このため、問題 $C_{i,k}^*$ の部分問題の中では、計算時間のかかる部品間の衝突チェ

Fig.9 Use of AND/OR graph having been obtained before

ックに対してのみ、このような処理を行う。

なお、設計変更された部分によっては、設計変更前のAND/ORグラフをほとんど利用できない場合がある。この場合は、生成される組立順序は設計変更前のものと大幅に変わることになる。

5. 計算例

本方法の有効性を示すため、NEWS 3460上でPROLOGを用いてプロトタイプ・システムを構築した。ここでは、図10(a)に示す簡単な設計対象に対するシステムの実行例を示す。ただし、この設計対象で、部品②、③、④、⑤から構成される部品群

Fig. 10 A design object and its precedence graph

Fig. 11 A part of search graph generated by the system

Table 1 Computing time

Number of connectors generated at one step	Use of G*	Computing time s
1	with using G*	1440
	without using G*	3600
2	with using G*	3960
	without using G*	11160

は運動可能で、部品①はその運動方向を拘束しており、部品②、③は部品④、⑤を締結している。図10(b)は、この設計対象の部品接続グラフであり、図10(c)はルールの適用により生成された優先関係グラフである。この優先関係グラフを用いて、まず、この設計対象の組立順序を算出し、次に、部品①の形状を変更して、再び組立順序を算出した。その結果、図11の探索グラフが得られた。ただし、これは探索時にコネクタを二つずつ生成した場合の探索グラフである。この図で太線で描かれた部分が解グラフである。また、色をつけた部分SGは、設計変更前のAND/ORグラフの一部をコピーすることによって生成された部分を表す。なお、この図には示されていないが、 $C_{2,3}^{1,3}$ の部分問題である“部品③と部品②間の衝突チェック”、“部品③と部品④間の衝突チェック”、“部品③と部品⑤間の衝突チェック”も、設計変更前の解を利用することができた。

表1に探索時に生成するコネクタが一つの場合と二つの場合、また、設計変更前のAND/ORグラフG*を利用する場合と利用しない場合の計算時間の比較を示す。この表に示すように、設計変更前のAND/ORグラフG*を利用することにより、計算時間は40%以下に減少した。また、最短計算時間は、コネクタを1個ずつ生成し、設計変更前のAND/ORグラフG*を利用した場合であり、1440sであった。なお、既存の手法で同じ解グラフを算出する場合、少なくともこの解グラフに含まれる問題は処理される。このため、既存の手法による組立順序の算出時間（ただし、組付け可能性の検証方法や計算機環境が同じ場合）は、この解グラフに含まれる問題の処理に要する計算時間よりも必ず大きくなる。この計算時間は3600sであったが、これは本手法の最短計算時間の2.5倍である。

本実行例は簡単なものであったが、筆者らはこの他、20個

の部品から構成される歯車ポンプに対しても、提案した方法に基づいて組立順序を算出した。この歯車ポンプに対する計算例では、設計変更前のAND/ORグラフを利用することにより、処理する部分問題の数が約14%に減少することを確認している。

6. 結 言

本研究で得られた結論を以下に示す。

- (1) 既に組立順序が得られている設計対象を設計変更し、再び、組立順序を算出する場合、その処理を効率化する方法を示した。
- (2) 組立順序の探索空間を縮小するため、組付けに関するヒューリスティクスを用いる方法を示した。
- (3) 縮小された探索空間内で、コストを最小化する組立順序を求める探索アルゴリズムを提案した。
- (4) 実行例により本方法の有効性を示した。

参 考 文 献

- 1) D. F. Baldwin, T. E. Abell, M. M. Lui, T. L. DeFazio and D. E. Whitney : An Integrated Computer Aid for Generating and Evaluating Assembly Sequences for Mechanical Products, IEEE Trans. Robotics and Automation, 7, 1, (1991) 78.
- 2) E. Molloy, H. Yang and J. Browne : Design for Assembly within Concurrent Engineering, Ann. CIRP, 40, 1, (1991) 107.
- 3) 伊藤公俊, 曾布川拓司: 機械のアセンブリ計画の自動生成 (第1報) -問題の定式化-, 精密工学会誌, 56, 7 (1990) 1262.
- 4) S. Lee : Backward Assembly Planning with Assembly Cost Analysis, Proc. 1992 IEEE Int. Conf. on Robotics and Automation, (1992) 2382.
- 5) S. Lee and C. Yi : Subassembly Stability and Reorientation, Proc. 1993 IEEE Int. Conf. on Robotics and Automation, (1993) 521.
- 6) 金井 理, 高橋秀智, 小野塚守: 統合化組立工程設計支援システムの開発 (第6報), 1993年度精密工学会春季大会学術講演会講演論文集, (1993) 1035.
- 7) 内山直樹, 荒井栄司, 井越昌紀: 機械製品の組立可能性の検証のための分解経路の生成, 日本機械学会論文集(C編), 59, 560 (1993) 1300.
- 8) N. J. Nilsson (白井良明, 辻井潤一, 佐藤泰介 訳): 人工知能の原理, 日本コンピュータ協会, (1983) 94.
- 9) 村山 長, 大場史憲: Design for Assembly のための組立経路の探索, 日本機械学会講演論文集, No.930-27 (1993) 372.