階層的積木法
– メタ解法の新しいフレームワーク –

久保 幹雄
東京商船大学 流通情報工学
藤沢 克樹
東京工業大学 数理・計算科学

1　はじめに

メタ解法とは困難な組合せ最適化問題に対する高精度な解を短時間で求めるためのヒューリスティックスの新しいパラダイムである．

本論文では，良いメタ解法を設計するための新しいフレームワークを提案する．設計の基本になるのは Genetic Algorithm[6] や Evolution Algorithm[10] などのメタ解法の基礎となる「積木仮説」である．積木仮説とは，一言で言えば「良い解は良い構成要素（積木）から構成される」という漠然とした仮定である（ここで用いる積木仮説は，Genetic Algorithm や Evolution Algorithm で用いられてきた標準的なものを簡略化したものである．従来の積木仮説の標準形については § 2 で触れる）．

ここではさらに，積木の階層構造の概念を導入する．積木の階層における最下層は解を定義するための最小の集合（台集合）であり，最上層が解の集合に相当する．ここで提案するメタ解法の基礎になるのは「積木仮説」を拡張した「階層的積木仮説」である．ここで「階層的積木仮説」とは，一言で言えば「良い積木は，その積木からみて下層にある良い積木から構成される」という仮定である．

本論文では，この仮説に基づくメタ解法の新しいフレームワーク（階層的積木法）を提案する．階層的積木法は，ローカルサーチ [7]，タブーサーチ [1, 4, 5]，Simulated Annealing 法 [8, 9] などに代表される近傍をベースにした探索法（近傍探索法），Genetic Algorithm，貪欲ランダム化適応型探索法（greedy randomized adaptive search procedure: GRASP）[2] などの代表的なメタ解法を特殊ケースとして含む一般的なフレームワークであると同時に，適応型記憶計画 (adaptive memory programming) の哲学を数学的に実現した解法とも言える．本論文では，階層的積木法を代表的な組合せ最適化問題に対して適用する方法についても詳述する．

論文の以下の構成は次のようになっている．

§ 2 では，本論文で用いる主な用語と記号の定義をあげる．

§3では，提案手法（階層的積木法）のフレームワークを示すとともに，従来のメタ解法が階層的積木法の特殊ケースであることを示す．
§4では，階層的積木法の種々の組合せ最適化問題に対する適用方法について述べる．
§5では，階層的積木法の運搬経路問題への適用方法について述べる．
§6では，結論と今度の課題を示す．

2 諸定義

ここでは，本論文で用いる用語と記号の定義をあげるとともに，ここで用いる積木の概念と，従来使われてきたものとの違いについて述べる．

ある空でない有限集合（台集合）$\mathcal{U}$が与えられたとき，解の集合 $\mathcal{F}$ を $\mathcal{F} \subseteq 2^{\mathcal{U}}$ と定義する．ここで，$2^{\mathcal{U}}$ は集合 $\mathcal{U}$ のべき集合(部分集合の集合)を表す．通常，解の集合$\mathcal{F}$は$2^{\mathcal{U}}$において，ある条件を満たすものの中で極大な要素から構成される．解の集合 $\mathcal{F}$および写像 $f : \mathcal{F} \to R$ が与えられたとき

$$\max\{f(x) : x \in \mathcal{F}\}$$

を与える$x \in \mathcal{F}$を求める問題が，ここで対象とする組合せ最適化問題である．関数 $f(x)$ を目的関数と，目的関数を最大にする解$x \in \mathcal{F}$を最適解と呼ぶ．

解の集合$\mathcal{F}$を与えたとき，近傍Nは以下の写像と定義される．

$$N : \mathcal{F} \to 2^{\mathcal{F}}$$

近傍は何らかの意味で近い解の集合を表す．$x \in \mathcal{F}$と$y \in N(x)$の組 (x, y) を移動と呼ぶ．ローカルサーチ，Simulated Annealing 法，タブーサーチをはじめとする多くのメタ解法は近傍を基に設計される．以下では，近傍をベースにして設計されたメタ解法を総称して近傍探索法と呼ぶ．

$\mathcal{F}$に含まれる解xの部分集合全体からなる集合を積木集合 $\tilde{\mathcal{F}}$ (図 1) と呼び，その要素を積木と呼ぶ．言い換えれば，$\tilde{\mathcal{F}}$は$\mathcal{F}$を単調化した集合である．ここで，$\tilde{\mathcal{F}}$は解の集合および台集合を含む，すなわち$\mathcal{U} \subseteq \tilde{\mathcal{F}}, \mathcal{F} \subseteq \tilde{\mathcal{F}}$が成立する．

二つの積木$x, y \in \tilde{\mathcal{F}}$が$x \subseteq y$を満たすとき，$x$は$y$に対して下層の積木であるといい，逆に$y$は$x$の上層の積木であるという．解$x \in \mathcal{F}$は，それより上層の積木を持たないので最上層の積木であり，逆に台集合の要素$x \in \mathcal{U}$は，空集合を除いてそれより下層の積木を持たないので最下層の積木である．

ここで，上で定義した積木 (building block) の概念は，従来の研究（特に Genetic Algorithm や Evolution Algorithm に代表される遺伝的メカニズムを利用したメタ解法）で用いられている積木の概念と若干異なることに注意されたい．Genetic Algorithm では解を $0, 1$ のビット列として表現し（ただし Evolution Algorithm では任意の表現法を許す場合もある），部分列をスキーマ（schema, similarity template）と呼び，定義長が短くかつ適応度が高いスキーマを積木と呼んでいる [6]．また，Genetic Algorithm における積木仮説

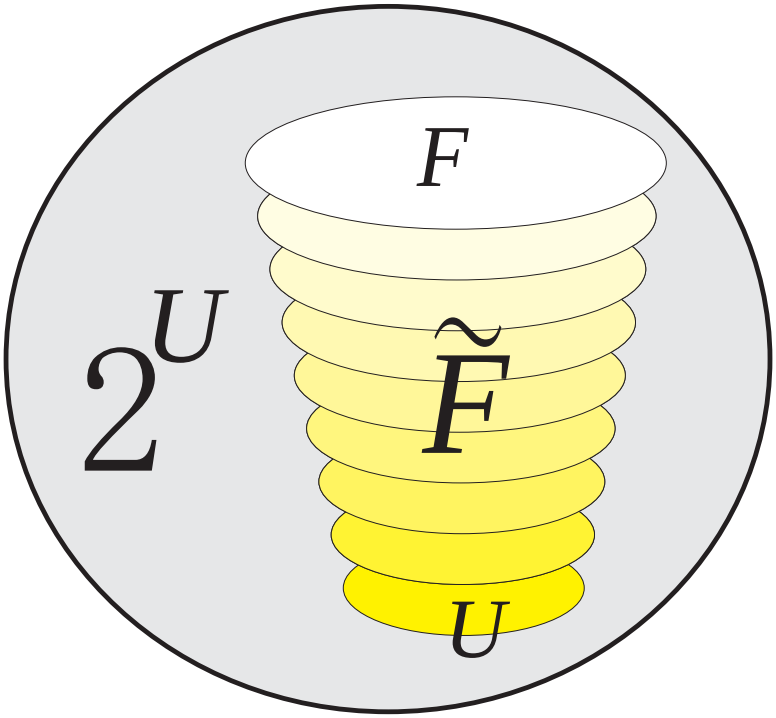

図 1: 積木の階層

(building block hypothesis) も，前に述べた「良い解は，良い積木を組み合わせたものである」という単純な形ではなく，Genetic Algorithm が効果的に解を改善できるための仮定であり，通常は以下のようなものを指す.

(積木仮説: Genetic Algorithm 版)
(Genetic Algorithm の定義でいうところの) 積木は他のスキーマと比べて (Genetic Algorithm における自然淘汰を表す操作で) 生き残る可能性が高く，さらに良い (すなわち定義長が短く適応度の高い) スキーマが生成される.

本論文で用いる仮説はより単純なものであり，多くの組合せ最適化問題に対しては妥当なものと考えられる.

(階層的積木仮説)
良い積木は，その積木からみて下層にある良い積木から構成される.

もちろん，この仮説が成立しない問題を作ることは容易であるが，§ 4 で示すように，我々の興味の対象となる実用的な組合せ最適化問題の多くはこの仮定を満たすものと考えられる.

3　階層的積木法

ここでは，階層的積木法の一般形を提示するとともに，従来のメタ解法との関連について述べる．

§ 3.1 では，積木集合の多様性の尺度（多様度）の概念を導入する．

§ 3.2 では，解集合に対する目的関数を積木集合に拡張した概念を導入する．

§ 3.3 では，積木集合上での移動操作を定義する．ここで用いるのは，通常の解集合に対する近傍探索法をもとにした移動（近傍探索移動 NEIGHBOR: § 3.3.1），下層の積木をもとに上層の積木を組み立てる移動（組立移動 BUILD: § 3.3.2），および上層の積木をもとに下層の積木の集合を生成する移動（分解移動 DECOMPOSE: § 3.3.3）の三つである．

§ 3.4 では，初期積木の生成方法について述べる．

§ 3.5 では，階層的積木法の一般的なフレームワークを示す.

3.1　積木集合の多様性の尺度

集中化は，§ 3.3.1 で述べる近傍探索法によって達成されるが，多様化の達成は比較的難しい．そのため，多様化は Genetic Algorithm における交叉やタブーサーチにおける長期メモリのように陰的に扱われていただけであった．最近になって多様化を陽的に組み込んだメタ解法が幾つか提案されている．例えば，Fleurent-Ferland [3] によるタブーサーチと Genetic Algorithm のハイブリッド法では解の集合に対するエントロピーの概念を導入している．しかし，彼らの提案している多様化の尺度は問題依存であり，一般の組合せ最適化問題に対して適用できる多様化の尺度ではない．

ここでは，探索に多様化を陽的に導入するための基礎として，解（積木）の多様性を測るための新しい尺度（多様度）を導入する．ここで提案する多様度は，問題の構造に依存せず任意の組合せ最適化問題に適用可能なものである．

与えられた積木の集合 $\mathcal{B} \subseteq \tilde{\mathcal{F}}$ がどれだけ多様性を持っているかを測るためには，積木 $x \in \mathcal{B}$ がどれだけ一様に台集合の要素を含んでいるから定量化すれば良い．台集合 $\mathcal{U}$ に番号を付け $\mathcal{U} = \{1, \cdots, |\mathcal{U}|\}$ とし，積木 $x \in \mathcal{B}$ を $|\mathcal{U}|$ 次元の特性ベクトルとみなす．すなわち，ベクトル x の第 i 成分が 1 のとき，台集合の第 i 番目の要素が積木 x に含まれ，0 のとき台集合の第 i 番目の要素が積木 x に含まれないものとする．$\mathcal{B}$ に含まれる全ての積木に対して，特性ベクトルの成分ごとの和をとった $|\mathcal{U}|$ 次元ベクトルを $X(\mathcal{B})$ とする．すなわち，

$$X(\mathcal{B})_i = \sum_{x \in \mathcal{B}} x_i \quad i = 1, \cdots, |\mathcal{U}|$$

とする．ここで $X(\mathcal{B})_i$ はベクトル $X(\mathcal{B})$ の第 i 成分である．

$N = \sum_{i=1}^{|\mathcal{U}|} X(\mathcal{B})_i$ とする．ベクトル $X(\mathcal{B})$ が，要素の合計が N の $|\mathcal{U}|$ 次元ベクトルの中で最も多様性が高いと考えられるベクトル $X^* = (N/|\mathcal{U}|, \cdots, N/|\mathcal{U}|)$ からどの程度はなれているかを多様化の尺度と考える．

定義 1: 積木の集合 $\mathcal{B} \subseteq \tilde{\mathcal{F}}$ の多様度 $D(\mathcal{B})$ は

$$D(\mathcal{B}) = \sum_{i=1}^{|\mathcal{U}|} (X(\mathcal{B})_i - N/|\mathcal{U}|)^2$$

と定義される．

例えば，$X(\mathcal{B}) = (3, 1, 2, 0)$ のとき，$N = 6, |\mathcal{U}| = 4$ であるので最も多様性が高いと考えられるベクトルは $X^* = (6/4, 6/4, 6/4, 6/4)$ となり，多様度 $D(\mathcal{B})$ は

$$D(\mathcal{B}) = (3 - 6/4)^2 + (1 - 6/4)^2 + (2 - 6/4)^2 + (0 - 6/4)^2 = 5$$

と計算される．

上で定義した多様性の尺度は，$X(\mathcal{B})_i, i = 1, \cdots, |\mathcal{U}|$ を確率変数とみたときの分散，もしくは点列の一様性を測定するための尺度である discrepancy の概念を変形したものであると考えられる．

3.2 積木の評価尺度

目的関数 f は解集合上で定義されていたが，ここではそれを積木集合上に拡張した関数を導入する．

積木集合 $\tilde{\mathcal{F}}$ の要素に対して評価尺度 $\tilde{f}$ を以下の写像として定義する．

$$\tilde{f} : \tilde{\mathcal{F}} \to R$$

ここで，解 $x \in \mathcal{F}$ に対しては $\tilde{f}(x) = f(x)$ である．一般には，最適解に含まれる可能性が高い積木は “良い”積木と考えられるので評価尺度を大きくし，逆に最適解に含まれる可能性が低い積木は “悪い”積木であると考えられるので評価尺度を小さくとれば良い．しかし，実際には解くべき問題の構造を考慮して $\tilde{f}$ を定義する必要がある．問題に依存した $\tilde{f}$ の定義方法については § 4 で詳述する．

さらに，積木 $x \in \tilde{\mathcal{F}}$ に対して多様化および集中化の尺度の概念を加味する方法を提案する．通常の評価尺度 $\tilde{f}(x)$ を持つ積木 $x \in \tilde{\mathcal{F}}$ および積木の集合 $\mathcal{B}$ を与えたとき，評価尺度を以下のように定義する．

$$\tilde{f}(x) + \alpha \times \sum_{i \in x} X(\mathcal{B})_i$$

ここで，α は多様化と集中化を制御するためのスカラーパラメータであり，$\alpha > 0$ ならば $\mathcal{B}$ と同じ積木を生成することを避けるために x の評価尺度を大きめにすることを表し，$\alpha < 0$ ならば $\mathcal{B}$ と同じ積木を生成するように x の評価尺度を小さめにすることを表す．

3.3 積木集合に対する移動操作

積木集合上では，複数の積木から別の積木を構成したり，一つの積木から複数の積木を構成したりする操作が重要になる．ここでは，積木集合 $\tilde{\mathcal{F}}$ 上での移動の概念に対応

するものとして，通常の近傍探索法における探索に対応する近傍探索移動 NEIGHBOR，下層の積木から上層の積木を生成する組立移動 BUILD，上層の積木から下層の積木を生成する分解移動 DECOMPOSE の三つの移動操作を導入する．

NEIGHBOR:

解 $x \in \mathcal{F}$ を与えたとき，何らかの近傍探索法によって $\mathcal{F}$ 内の探索を行い，探索の途中で得られた解の集合（もしくは一つの解）X を返す．

BUILD:

積木の部分集合 $\mathcal{B} \subseteq \tilde{\mathcal{F}}, \mathcal{B} = \{\beta_1, \cdots, \beta_m\}$ を与えたとき，$\{\beta_1, \cdots, \beta_m\}$ を合成して得られる上層の積木 $\beta \in \tilde{\mathcal{F}}, \cup_{i=1}^{m} \beta_i \subseteq \beta$ を返す．

DECOMPOSE:

解 $x \in \mathcal{F}$ を与えたとき，$\cup_{i=1}^{m} \beta_i \subseteq x$ を満たす積木の集合 $\{\beta_1, \cdots, \beta_m\}$ を返す．

以下では，各々の移動操作の実現方法について述べる．

3.3.1 近傍探索移動

近傍探索移動 NEIGHBOR は，一つの解 $x \in \mathcal{F}$ から別の解（の集合）を得るための操作である．この操作の実現のためには，近傍の設定と探索の進め方を決定する必要がある．近傍の決定については問題の構造に強く依存する．§ 4 で，個々の問題への適用方法について述べるときに触れるものとして，ここでは探索の進め方について述べる．

ローカルサーチ
Simulated Annealing 法
Genetic Algorithm
GRASP
タブーサーチ
等が挙げられるが，ここでは，計算時間と得られる解の精度からローカルサーチとタブーサーチを用いるものとする．

3.3.2 組立移動

組立移動 BUILD とは，積木の集合 $\mathcal{B} \subseteq \tilde{\mathcal{F}}$ が与えられたとき，解（の集合）もしくは $\mathcal{B}$ より上層の積木（の集合）を生成する操作である．

評価尺度 $\tilde{f}$ が大きい積木ほど，良い解（もしくは積木）を生成するものと考えられるが，生成される解（もしくは積木）が多様性を持つためには評価尺度 $\tilde{f}$ が小さい積木も用いる必要がある．ここでは，ランダム性を用いて，積木を選択し解（もしくは積木）を構築していく方法を採用する．ここで，評価尺度が上位の積木の選択確率を増すと得られる解（もしくは積木）に集中化をもたらし，選択確率を下げると多様化をもたらす．

この確率は，現在の積木の集合$\mathcal{B}$の多様化の度合いの尺度をもとに適宜決められるものとする．この操作を関数RANDOM_ARG_MAX で表す.

procedure 組立移動 BULID
1 $x := \emptyset$
2 $\mathcal{B} :=$ 積木の集合
3 **while** stopping-criterion $\neq$ yes **do**
4 $y^* :=$ RANDOM_ARG_MAX $\{\tilde{f}(y) : y \in \mathcal{B}, x \cup y \in \tilde{\mathcal{F}}\}$
5 もしそのようなy^*がなければ STOP.
6 $x := x \cup y^*$
7 $\mathcal{B} := \mathcal{B} \setminus \{y^*\}$
8 もし$x \in \mathcal{F}$ならSTOP.

例えば，確率的タブーサーチのアイディアを用いるとRANDOM_ARG_MAX は次のような操作となる．

最も評価尺度の大きい要素（積木）を選択するか否かを確率$p(\geq 0)$で表が出る硬貨を投げることによって決める．表が出たらその要素を採択し，裏が出たときには次の要素を調べ，再び同様に硬貨を投げて採択か否かを決める．この操作を何れかの要素が選択されるまで繰り返す.

ここで，選択確率pが1に近いときには集中化が促され，pが0に近いときには多様化が促される．

しかし，このような方法ではランダム性にだけ頼って多様化を行っているので同じ積木が何度も生成されてしまう可能性が高い．ここでは，強制的に異なった積木を生成するための仮想木法を提案する．

まず，評価尺度の大きい順にその要素を選択するか否かを表す列挙木を考える．この列挙木は陽的に表現する必要はなく“仮想的”に生成されるものであるが，操作の表現のためにいまは木が生成されているものと仮定して話をする．はじめの枝分かれは，評価尺度が最大の要素を採択するか否かを表しており，確率的タブーサーチと同様にパラメータpを用いて，採択する確率をp，採択しない確率を$1-p$とする．以下同様に，2番目以降の要素を採択する枝を確率pで，採択しない枝を確率$1-p$で決める．この列挙木の最終段階（葉）が生成される確率はその葉に至る枝の選択確率の積で表される．ここで，ある閾値θを用いて，葉が生成される確率がθ以上のものを生成された積木の集合とする．

3.3.3 分解移動

分解移動 DECOMPOSE とは，解$x \in \mathcal{F}$が与えられたとき，$\cup_{i=1}^{m} \beta_i \subseteq x$を満たす積木の集合$\{\beta_1, \cdots, \beta_m\}$を生成する操作である．

生成された積木の集合は，小さすぎても大きすぎても意味のある積木にならない．

3.4 初期積木の生成

階層的積木法は，積木集合の部分集合 $\mathcal{B} \subseteq \tilde{\mathcal{F}}$ を逐次改善していく方法である．したがって，初期の積木の集合を与える必要がある．この操作（INITIALIZE）は基本的には問題依存であるが，ここでは幾つかの指針を示す.

最も単純な方法は，台集合 $\mathcal{U}$ 自身を $\mathcal{B}$ とすることである．このとき，良好な解に含まれる可能性の低い台集合の要素は，あらかじめ $\mathcal{B}$ から外しておく方法も有効であると考えられる.（適用例：巡回セールスマン問題）

また，良い解に頻繁に含まれると考えられる積木を積極的に生成し，それを初期積木の集合に含めておく方法も有効である.（適用例）

3.5 階層的積木法のフレームワーク

以上の構成要素をもとに，階層的積木法一般形は以下のように書ける.

procedure 階層的積木法（一般形）
1 $\mathcal{B}$:= INITIALIZE
2 **while** stopping-criterion $\neq$ yes **do**
3 以下の3通りの移動操作の内の何れかを適用する
4 a) $x \in \mathcal{B} \cap \mathcal{F}$ に対して NEIGHBOR (x) を $\mathcal{B}$ に加える.
5 b) $x \in \mathcal{B} \cap \mathcal{F}$ に対して DECOMPOSE(x) を $\mathcal{B}$ に加える.
6 c) BUILD $(\mathcal{B})$ を $\mathcal{B}$ に加える.

上のアルゴリズムにおいて，積木集合に要素を加える際に，$\mathcal{B}$ の位数が上限に達しているときには，最も不適合な積木を $\mathcal{B}$ から除くことによって積木集合の要素数を一定数以下に保つものとする.

上のアルゴリズムの $4,5,6$ 行目の何れの操作も積木の集合 $\mathcal{B}$ に対する変更を行うが，互いに情報を交換する必要がないので，階層的積木法は並列計算機上で効率的に実装できる.

なお，並列計算機を用いない場合には以下の簡略化された階層的積木法のフレームワークを用いる.

procedure 階層的積木法（非並列型）
1 $\mathcal{B}$:= INITIALIZE
2 **while** stopping-criterion $\neq$ yes **do**
3 x := BUILD $(\mathcal{B})$
4 y := NEIGHBOR (x)
5 DECOMPOSE (y) を $\mathcal{B}$ に加える.

上のアルゴリズムにおいて，BUILD は解を返すように設計されているものとする.

4　適用例

ここでは，階層的積木法の種々の組合せ最適化問題への適用について述べる．

§ 4.1 では，巡回セールスマン問題に対する階層的積木法の適用例をあげるとともに，台集合に制限を付ける方法や積木の評価尺度を表す写像 $\tilde{f}$ の決定法について述べる．

§ 4.2 では，グラフ分割問題に対する階層的積木法の適用方法について述べる．ここでは，クリーク（完全部分グラフ）を積木と考える．

§ 4.3 では，グラフ彩色問題に対する階層的積木法の適用方法について述べる．

§ 4.4 では，二次割当問題に対する階層的積木法の適用方法について述べる．

§ 4.5 では，ジョブショップスケジューリング問題に対する階層的積木法の適用方法について述べる．

4.1　巡回セールスマン問題

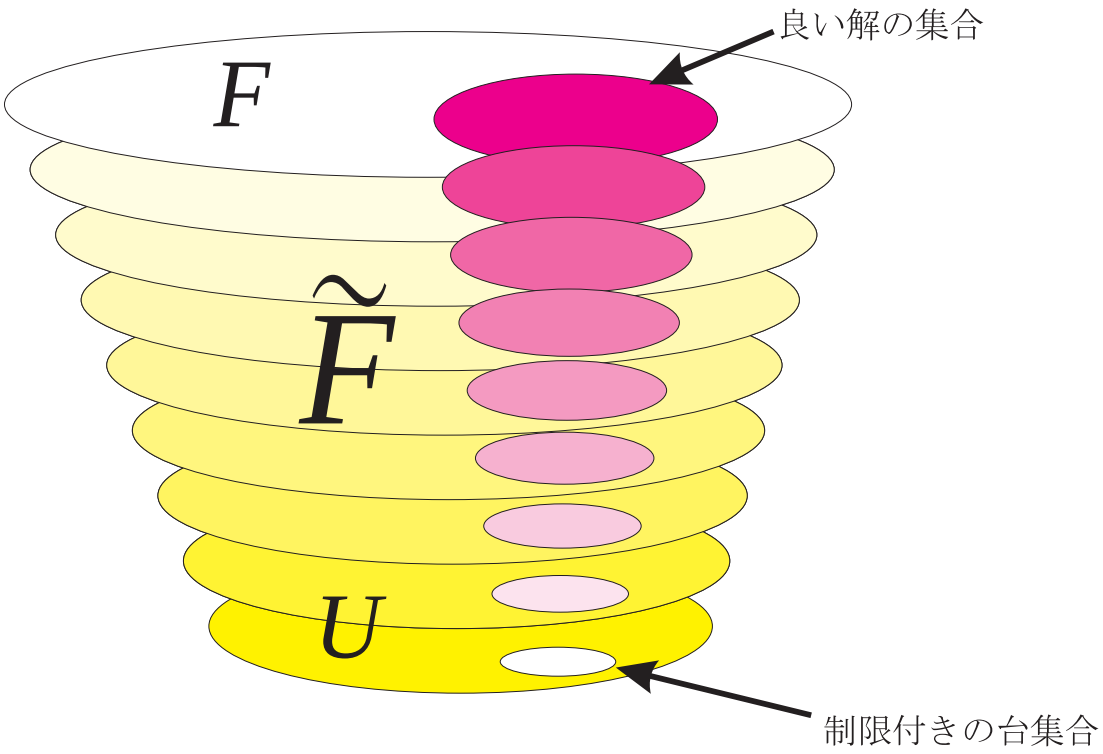

図 2: 積木の制限

巡回セールスマン問題とは，n 個の点 (都市) から成るグラフ $G = (V, E)$, 枝上の距離 (重み, 費用) 関数 $d : E \to Z^+$ が与えられたとき，全ての点をちょうど 1 回ずつ経由する巡回路 (Hamilton 閉路) で，枝上の距離の合計 (巡回路の長さ) を最小にするものを求める問題である．

ここで，台集合$\mathcal{U}$は枝の集合Eに対応し，$\mathcal{F}$の要素はHamilton閉路に対応する．積木集合$\tilde{\mathcal{F}}$の要素は枝の部分集合であるが，閉路を含むものは良い解を生成するための積木にはなり得ないので除外する．言い換えれば，閉路を含む枝集合$x \in \tilde{\mathcal{F}}$に対しては$\tilde{f}(x)$を$\infty$と設定する．

従来のローカルサーチに対する研究で有効に用いられてきた技法として，解に含まれない要素を事前に除去しておくことがあげられる．例えば，巡回セールスマン問題においては長い枝は良い解に含まれる可能性が低いと考え，事前に除去しておく方法が多くのヒューリスティックスで用いられている．この技法は，最近では候補集合（candidate set）と呼ばれ，巡回セールスマン問題に対する有効な近似解法を作るための基本となっている．ここで提案する手法においては，最下層の積木に対して評価尺度を与え，良い積木を構成しえない積木からは次のレベルの積木を構成しないことに対応する．

また，台集合$\mathcal{U}$に対しても写像dが定義されているので，それを用いて積木の評価尺度を設定する．最も簡単な評価尺度は平均枝長であり，積木$x \in \tilde{\mathcal{F}}$に含まれる枝の集合を$E(x)$と書くとき，以下のように定義される．

$$\tilde{f}(x) = \sum_{e \in E(x)} d_e / |E(x)|$$

他にも点のクラスターを考慮した尺度が考えられる．

候補集合（制限された台集合：図2）において，点iに隣接している点の集合を$\delta(i)$と書く．すなわち，制限された台集合（枝集合）を$\hat{E}$とすると

$$\delta(i) = \{j \in V : (i,j) \in \hat{E}\}$$

である．このとき，候補集合上で点iに隣接している点への平均距離を$\bar{d}_i$と書く．すなわち，

$$\bar{d}_i = \sum_{j \in \delta(i)} d_{ij}$$

である．以上の定義より，クラスターを考慮した積木xの評価尺度は次のように定義される．

$$\tilde{f}(x) = \frac{1}{|E(x)|} \sum_{(i,j) \in E(x)} d_{ij} / (\bar{d}_i + \bar{d}_j)$$

4.2 グラフ分割問題

無向グラフ$G = (V, E)$が与えられたとき(点数$n = |V|$は偶数とする), 点集合Vの一様分割(L, R)とは，$L \cap R = \emptyset$,$L \cup R = V$, $|L| = |R| = n/2$を満たす点の部分集合の対である．グラフ分割問題とは，LとRの間にある枝の本数を最小にする一様分割(L, R)を求める問題である.

ここで，台集合$\mathcal{U}$は点集合Vに対応し，$\mathcal{F}$の要素は点の部分集合Lで$|L| = n/2$となるものに対応する．積木$\tilde{\mathcal{F}}$の要素は点の部分集合$S \subseteq V$であり，積木が良い解を生成するためには，カット$(S, V \setminus S)$に含まれる枝が少ないことが望ましい．したがって，完全部

分グラフ（クリーク）を積木として保持する方法を用いる．最大クリーク問題を解くことになるが、近傍探索法によって容易に優れた近似解を検出できる．k-way 分割におけるマッチングの利用も階層的積木の生成と考えられる．

4.3 グラフ彩色問題

無向グラフ $G = (V, E)$ の k 分割とは，点集合 V の k 個の部分集合への分割 $\Upsilon = \{V_1, \cdots, V_k\}$ で，$V_i \cap V_j = \emptyset, \forall i \neq j$, $\cup_{j=1}^{k} V_j = V$ を満たすものを指す．このとき，V_i を色クラス (color class) と呼ぶ．k 分割は，全ての色クラス V_i が安定集合のとき k 彩色と呼ばれる．グラフ彩色問題とは，与えられた無向グラフ $G = (V, E)$ に対して，最小の k (彩色数) を導く k 彩色 $V_1, \cdots, V_k$ を求める問題である．

彩色数 k を固定したグラフ彩色問題においては，台集合 $\mathcal{U}$ は $V \times \{1, \cdots, k\}$ であり，$\mathcal{F}$ は k 彩色に対応する．

グラフ彩色問題においては安定集合 (stable set) を積木として保持する．あるグラフ G 上で、最大クリーク問題を解くこととその補グラフ $\tilde{G}$ 上で、最大安定集合問題を解くことは同値である．よって最大クリーク問題用の近傍探索法などを利用することができる．

4.4 二次割当問題

二次割当問題とは，集合 $V = \{1, \cdots, n\}$ および $n \times n$ 行列 $F = [f_{ij}]$,$D = [d_{k\ell}]$ が与えられたとき，

$$\sum_i \sum_j f_{ij} d_{\pi(i)\pi(j)}$$

を最小にする順列 $\pi : V \to \{1 \cdots, n\}$ を求める問題である．

ここで，台集合 $\mathcal{U}$ は $V \times V$ に対応し，$\mathcal{F}$ の要素は V から V への一対一写像，すなわち V 上の順列 π に対応する．よって適当な長さの順列を積木として保持する．近傍探索法などによって多くの近似解を算出し、それらの共通部分を積木とする方法が考えられる．

4.5 ジョブショップスケジューリング問題

作業の集合 $J = \{0, 1, \cdots, n\}$ (0 と n は開始と終了を表すダミー作業) および機械の集合 $M = \{1, \cdots, m\}$ が与えられている．作業間には先行順序が存在し，先行関係を有向枝 A で表す．機械 $k \in M$ 上には，その機械上で処理される作業の集合 J_k がある．J_k 上での完全有向グラフ $G_k = (J_k, E_k)$ を考え，その枝集合を離接枝と呼ぶ．G_k 上の任意の 2 点 i, j に対して，(i, j) もしくは (j, i) のいずれかの離接枝を選ぶ．このとき，得られるグラフが閉路を持たないように選ぶと，J_k 上の全順序が一意に定まる．これが機械 k 上で処理される作業の順序を表す順列に対応する．作業を点の集合，有向枝，離接枝を枝の集合した有向グラフ $D = (J, \cup_{k \in M} E_k \cup A)$ を離接グラフと呼ぶ (図 3 参照)．ジョブショップ

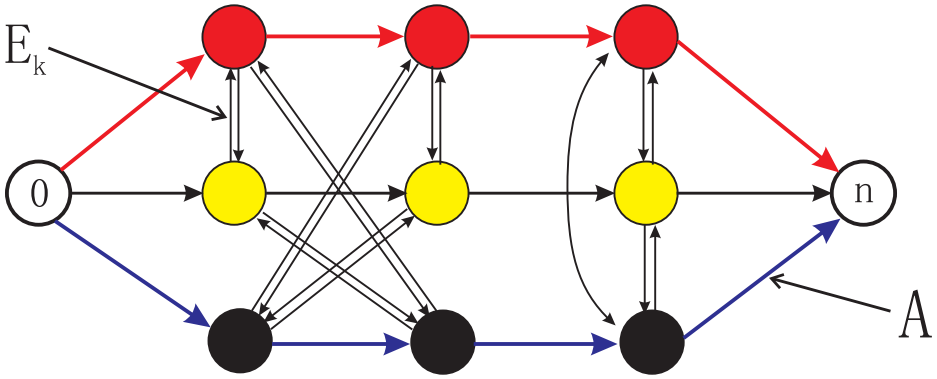

図 3: 離接グラフ

スケジューリング問題とは，離接グラフ上で作業 $0, n$ 間の最長路の長さが最小になるように，二つの作業間の離接枝から一つを選択する問題である. (離接枝が閉路を形成するときには，最長路の長さを無限大と定義する).

ここで，台集合 $\mathcal{U}$ は離接枝の集合 $\cup_{k\in M} E_k$ に対応し，$\mathcal{F}$ の要素は各々の機械 $k \in M$ に対する J_k 上の順列に対応する．この問題においては一つの機械上での離接枝の集合を積木として保持する.

5　運搬経路問題における階層的積木法

階層的積木法を以下の仮定を持つ運搬経路問題の一般形に対して適用する.

- 単一のデポにある複数の運搬車で全ての顧客の需要を満たす.
- 顧客の需要量は既知である.
- 各地点間の移動時間，移動費用は既知とする.
- 顧客上での作業の開始時刻は，あらかじめ与えられた最早開始時刻，最遅開始時刻の間でなくてならない (時間枠条件)．ここで，時間枠は最早到着時刻と最遅到着時刻の組で表される．時間枠付きの問題が通常の運搬経路問題と大きく異なる点は，運搬車が最早到着時刻より早く到着したときに“待ち”が生じることである．また，最遅到着時刻より遅い時刻での到着は許さない.
- 一つのルートに含まれる顧客の需要量の重量合計は，そのルートに割り当てられた運搬車の積載重量上限を超えない (積載重量条件)．ここでルートとは，デポを出発した運搬車が何人かの顧客を訪問し再びデポに戻る巡回路を指す.
- 運搬車ルートの総移動時間（これは顧客上での作業時間を含む）は，稼働時間の上限を超えない．ここで運搬車ルートとは，運搬車の一日のスケジュールを表し，幾

つかのルートを合わせたものである．

- 運搬車の総移動費用を最小化する運搬車ルートを求めることを問題の目的とする．

この問題は，時間枠条件および運搬車が一日に何回かデポを出発する条件（回転条件）を有する運搬経路問題と考えられ，多くの実務的な問題に対するコンサルティングから生まれたものである．

5.1 積木の階層

本論文で扱う運搬経路問題においては，以下のような積木の階層が自然に定義できる．

第 1 層 (台集合 $\mathcal{U}$): 枝（点の対）の集合 E

第 2 層（オプション）: パスの集合 P

第 3 層: ルート（巡回路）の集合 T

第 4 層: 運搬車ルートの集合 R

第 5 層: 解集合 $\mathcal{F}$

5.2 近傍探索移動 NEIGHBOR

ここでは，近傍探索移動 NEIGHBOR を設計する．設計の基本になるのはタブーサーチであり，様々な付加条件に柔軟に対応するために，制約を目的関数に適応的に組み込むペナルティ関数法（適応型ペナルティ関数法）を用いる．

タブーサーチを設計するための基本材料は近傍と属性である．

移動 (k, v) の際の目的関数値の減少量を $\Delta(k, v)$ と書く．

以上の要素を組み込んだタブーサーチは，疑似コードを用いて以下のように書くことができる．

procedure 近傍探索移動 NEIGHBOR
1 iter := 0
2 $LS(k, v) := 0$ for all $k \in K, v \in V$
3 $LTM(k) := 0$ for all $k \in K$
4 **while** stopping-criterion $\neq$ yes **do**
5 iter := iter + 1
6 $(k^*, v^*) = \arg\max_{k,v}\{\Delta(k, v) : LS(k, v) < \text{iter}\}$
7 $LS(k^*, v^*) := \text{iter} + TL$
8 $LTM(k^*) := LTM(k^*) + 1$

9　　移動 (k^*, v^*) を行う.
10 探索中に見つかった最良の実行可能解と最良の目的関数を持つ解を返す.

5.3　組立移動 BUILD

組立移動は以下の四つに分けられる.

第 1 組立移動 BUILD1:
枝集合（第 1 階層積木）からパス（第 2 階層積木）を生成する操作.

第 2 組立移動 BUILD2:
パス（第 2 階層積木）の集合からルート（第 3 階層積木）を生成する操作.

第 3 組立移動 BUILD3:
ルート（第 3 階層積木）の集合から運搬車ルート（第 4 階層積木）を生成する操作.

第 4 組立移動 BUILD4:
運搬車ルート（第 4 階層積木）から解を生成する操作.

階層的積木法においては，全ての組立移動操作は台集合上での集合の和演算で定義される．すなわち運搬経路問題の場合には，台集合は枝集合であるので，組立移動は枝集合の和をとる演算と定義される．このとき，二つの積木の和集合が再び積木になっているか否かを判定する必要がある.

5.3.1　第 1 組立移動 BUILD1

枝集合を限定する操作を行う．各顧客から近い (移動時間の少ない) 顧客との間の枝のみに限定する.

5.3.2　第 2 組立移動 BUILD2

パスの集合 P が与えられており，評価尺度の順に並べられているものとする．パスからルート（巡回路）を生成するためには，二つのパスの和集合で定義される枝集合が積木になっているか否かを判定する必要がある．枝集合がルートになるためには，以下の条件を満たす必要がある.

閉路条件: 閉路を含まない.

積載重量条件: 枝集合に含まれる顧客の需要量の合計が運搬車の積載重量上限を超えない.

稼働時間条件: 総移動時間が稼働時間の上限を超えない.

時間枠条件: 時間枠を破ることなく点（顧客）集合を訪問できる．

適当なデータ構造の下で積最重量条件と稼働時間条件は$O(1)$で判定可能である．
ここで解くべき問題は巡回セールスマン問題に対する貪欲解法の変形である．

5.3.3 第3組立移動 BUILD3

ルート（第3階層積木）の集合から運搬車ルート（第4階層積木）を生成する操作．
二つのルートを併せて運搬車ルートを生成するときに，以下の条件を判定する必要がある．

稼働時間条件: 総移動時間が稼働時間の上限を超えない．

ここで解くべき問題は，稼動時間をアイテムの大きさ，最大稼動時間をビンの大きさとみなしたときのビンパッキング問題と同値である．

5.3.4 第4組立移動 BUILD4

運搬車ルート（第4階層積木）から解を生成する操作．
各顧客は一つの運搬車ルートによってサービスを受ける必要がある．したがって，ここで解くべき問題は集合分割問題となる．

5.4 分解移動 DECOMPOSE

第5層 (実行可能解集合 $\mathcal{F}$) 上の近傍探索法によって得られた複数の近似解を，頻繁に使用されているルート，パスなどの積木の構成要素に分解し、それぞれ下層において保持する．

6 結論と今後の課題

本論文では、階層的積木法というメタ解法の新しいフレームワークを提案した．適用する問題によって階層の数や積木の要素は異なるが，探索の途中において解そのものを保持しておくにとどまらず，解の構成要素である積木を階層的に保持しておくのが特徴の一つである．BUILD や DECOMPOSE などの操作は問題依存であるために細部は規定できないが，その分柔軟性に富んでいるといえる．
階層的積木法においても，その性能を大きく左右するのは近傍探索移動 NEIGHBOR であるが，ここには既存のメタ解法をほぼそのまま利用することができる．
また近年メタ解法を設計する際には，集中化と多様化という概念が重要になって来ている．ここで，集中化とは良い解の周辺を集中的に探索することであり，多様化とは今までに探索していない新しい領域を強制的に探索させることである．多くのメタ解法は

この2つの相反する概念を取り込んではいたが，2つの概念を陽的に制御しているメタ解法はあまり見られない．そこで筆者らは，2つの概念を陽的に制御する集中化・多様化制御法を提案して開発を行い、階層的積木法と有機的に組み合わせることを目標としている．保持している積木の中から集中化や多様化に適した要素を選出して，最上層で行われる NEIGHBOR おいて陽的に集中化・多様化を制御するのがねらいである．

参考文献

[1] R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6, 1994.

[2] T.A. Feo, M.G.C Resende, and S.H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Opns. Res.*, 42(5):860–878, 1994.

[3] C. Fleurent and J. A. Ferland. Genetic and hybrid algorithms for graph coloring. *to appear in Annals of Operations Reseacrh*, 1994.

[4] F. Glover. Tabu search: fundamentals and usage. Working paper, University of Colorado, Boulder, 1995.

[5] F. Glover, M. Laguna, E. Taillard, and D. de Werra. Tabu Search. special issues of the *Annals of Operations Research*, Vol. 41, J.C. Baltzer.

[6] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[7] D. S. Johnson. Local optimization and the traveling salesman problem. In *Proc. 17-th Colloquium on Automata, Languages, Programming*, pages 446–461. Springer-Verlag, 1990.

[8] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part I, graph partitioning. *Operations Research*, 37:865–892, 1989.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[10] Z. Michalewicz. *Evolutionary Computation and Heuristics*. Kluwer Academic Publishers, 1996.