数理解析研究所講究録 1297 巻 2002 年 107-115

配置コストをもつ長方形詰込み問題に対する局所探索法の高速化

京都大学・情報学研究科　今堀 慎治　(Shinji Imahori)
柳浦 睦憲　(Mutsunori Yagiura)
茨木 俊秀　(Toshihide Ibaraki)
Graduate School of Informatics,
Kyoto University

Abstract: 長方形詰込み問題とは, 様々な大きさの長方形を二次元平面上に互いに重ならないように配置する問題であり, NP 困難であることが知られている. 我々は, 以前この問題に一般的な配置コストを導入し, 局所探索法に基づくアルゴリズムを提案した. この問題は非常に汎用的であり, 様々な配置問題やスケジューリング問題を扱うことが可能である. また, 提案アルゴリズムの特徴として, 順列対表現を用いて解を表現し, 低次の多項式時間で順列対から配置を求めるという点が挙げられる. 本研究では, このアルゴリズムをもとに, 局所探索における様々な近傍において, 解一つ当りの評価を高速に行う手法を提案する. 配置問題や実際のスケジューリング問題を用いた数値実験を行い, 提案手法が従来のアルゴリズムと比較して優れていることを確認した.

1 はじめに

長方形詰込み問題とは, 様々な大きさの長方形を二次元平面上に互いに重ならないように配置する問題であり, 代表的な組合せ最適化問題の一つであるとともに, 現実への応用の面からも重要な問題である. しかし, 一口に長方形詰込み問題と言っても, 様々な制約条件や目的関数を持つ問題が考えられ, 従来の研究ではその一部に特化したアルゴリズムが提案されてきた. このため, 我々は以前, 文献 [1] でこの問題に一般的な配置コストを導入し, 局所探索法に基づくアルゴリズムを提案した. [1] の問題は, 様々な配置問題やスケジューリング問題を自然に定式化できる, 非常に汎用性の高い問題である.

長方形詰込み問題において解を表現する方法にはこれまでに様々な方法が提案されてきた. 代表的なものとして, 順列を用いて解を表現し, この順列をもとにある基準で配置を求めるアルゴリズム [2, 3] や, 二次元のグリッド上に長方形を配置することで解を表現し, これをもとに実際の配置を求めるアルゴリズム [5] などがある. [1] のアルゴリズムの特徴としては, 順列対表現 [4] を用いて解を表現し, 低次の多項式時間で順列対から配置を求めるという点が挙げられる. 文献 [1] の結果は, 全ての長方形を覆う長方形の面積を最小化する問題に対する, 高橋のアルゴリズム [7] (長方形数を n とするとき, 計算時間が $O(n \log n)$) を含む結果である.

本稿では, 文献 [1] のアルゴリズムをもとに, 局所探索における様々な近傍において, 解一つ当りの評価を高速に行う手法を提案する. 提案手法により, たとえば前述の面積最小化問題を扱う場合, 解一つ当りの評価を, $O(1)$ もしくは $O(\log n)$ 時間で行うことができる. また, 提案手法が実用的

にも優れていることを示すため, 配置問題や実際のスケジューリング問題を用いた数値実験を行い, 提案手法が従来のアルゴリズムと比較して優れていることを確認した.

以下, 2節で一般的な配置コストをもつ長方形詰込み問題を定義し, 3節で我々の用いている解表現法である, 順列対表現の説明および特徴を述べ, 4節で順列対から配置を求めるアルゴリズムの説明をする. 5節では局所探索法およびメタ戦略について述べ, 続く6節で高速化手法を提案する. 7節では提案手法の有効性を様々な問題を用いた計算実験によって評価し, 最後の8節はまとめの節である.

2 問題の定義

長方形集合 $I = \{1, 2, \ldots, n\}$ と, 各 $i \in I$ に対し m_i 種類のモードが与えられる. 各長方形 $i \in I$ の各モード $k\,(k = 1, 2, \ldots, m_i)$ について,

$w_i^{(k)}$: 長方形 i のモード k での幅,

$h_i^{(k)}$: 長方形 i のモード k での高さ,

$p_i^{(k)}(x)$: 長方形 i のモード k での x 軸コスト関数,

$q_i^{(k)}(y)$: 長方形 i のモード k での y 軸コスト関数,

が与えられる. 配置は, 各長方形について1つのモードを選択し, さらに x と y の座標値を与えることで定まる. 配置 π における各長方形のモードを $\mu(\pi) = (\mu_1(\pi), \mu_2(\pi), \ldots, \mu_n(\pi))$ とする. また, 長方形 i の左下隅の座標を $(x_i(\pi), y_i(\pi))$ と記し, x 軸コストの最大値と y 軸コストの最大値を

$$p_{\max}(\pi) = \max_{i \in I} p_i^{(\mu_i(\pi))}(x_i(\pi)), \tag{1}$$

$$q_{\max}(\pi) = \max_{i \in I} q_i^{(\mu_i(\pi))}(y_i(\pi)), \tag{2}$$

と定義する. このとき, 全長方形を二次元平面上に互いに重なりなく配置し, 目的関数

$$g(p_{\max}(\pi), q_{\max}(\pi)) + c(\mu(\pi)) \tag{3}$$

を最小化する問題を考える. ただし関数 g は, $p_{\max}(\pi), q_{\max}(\pi)$ に関して単調非減少であると仮定する. また, $g(p_{\max}(\pi), q_{\max}(\pi))$ は $O(1)$ 時間で, $c(\mu(\pi))$ は $O(n)$ 時間で計算可能と仮定する. 本研究で提案するアルゴリズムは, 配置コスト関数 $p_i^{(k)}(x)$ と $q_i^{(k)}(y)$ として任意の区分線形関数 (不連続, 非凸でもよい) を扱うことができるため, 非常に汎用的である. また, モードの導入により, たとえば長方形の 90° 回転などが実現できるようになっており, さらに汎用性が高くなっている. この定式化で扱うことのできる問題の一部を7節で紹介する.

3 解の表現方法

本研究では, 順列対 (sequence-pair) 表現 [4] を用いて解を表現する. 順列対表現では, n 個の長方形の順列の対 $\sigma = (\sigma_+, \sigma_-)$ を考える. ここで, $\sigma_+(k) = i$ は, 順列 σ_+ において k 番目の長方形が

iであることを意味する (σ_- も同様). $\sigma=(\sigma_+,\sigma_-)$ より二項関係 $\preceq_\sigma^x$ と $\preceq_\sigma^y$ を,

$$\sigma_+^{-1}(i)\le\sigma_+^{-1}(j)\ \text{かつ}\ \sigma_-^{-1}(i)\le\sigma_-^{-1}(j)\Longleftrightarrow i\preceq_\sigma^x j,$$

$$\sigma_+^{-1}(i)\ge\sigma_+^{-1}(j)\ \text{かつ}\ \sigma_-^{-1}(i)\le\sigma_-^{-1}(j)\Longleftrightarrow i\preceq_\sigma^y j,$$

と定義する. 二項関係 $\preceq_\sigma^x$ と $\preceq_\sigma^y$ は, 定義よりそれぞれ半順序関係になる. また, 任意の対 i と j $(i\neq j)$ に対し「$i\preceq_\sigma^x j$ あるいは $j\preceq_\sigma^x i$」と「$i\preceq_\sigma^y j$ あるいは $j\preceq_\sigma^y i$」のちょうど一方が成立するという性質がある. 与えられた $\sigma=(\sigma_+,\sigma_-)$ と $\mu=(\mu_1,\mu_2,\ldots,\mu_n)$ に対し

- $\mu_i(\pi)=\mu_i$,
- $i\preceq_\sigma^x j$ かつ $i\neq j\implies x_i(\pi)+w_i^{(\mu_i)}\le x_j(\pi)$,
- $i\preceq_\sigma^y j$ かつ $i\neq j\implies y_i(\pi)+h_i^{(\mu_i)}\le y_j(\pi)$,

を満たす配置 π すべての集合を $\Pi_{\sigma,\mu}$ と定義する. このとき, 「任意の σ と μ に対し, $\Pi_{\sigma,\mu}\neq\emptyset$」かつ「任意の配置 π に対し, $\pi\in\Pi_{\sigma,\mu}$ となる (σ,μ) が存在する」という性質が成り立つ.

我々のアルゴリズムでは, (σ,μ) の探索には後述する局所探索法, およびそれを基本としたメタ戦略を用いる. また, $\Pi_{\sigma,\mu}$ の中で目的関数 (3) を最小にする配置 π は, 次節で述べるように動的計画法によって効率よく求めることができる.

4 動的計画法

順列対 σ とモード μ が与えられたとき, 目的関数 (3) を最小にする配置 $\pi\in\Pi_{\sigma,\mu}$ を決定する問題を考え, 動的計画法に基づくアルゴリズムを与える. なお, $\preceq_\sigma^x$ と $\preceq_\sigma^y$ の性質から, $p_{\max}(\pi)$ と $q_{\max}(\pi)$ をそれぞれ独立に最小化すれば (3) を最小化できることが示せるので, ここでは, $p_{\max}(\pi)$ の最小値とそれを実現するような x 座標を求めるアルゴリズムのみを示す. y 方向についてもほぼ同様である.

まず, 計算のため,

$$J_i^{\mathrm{f}}=\{j\in I\mid j\preceq_\sigma^x i\},$$

$$J_i^{\mathrm{b}}=\{j\in I\mid i\preceq_\sigma^x j\},$$

$f_i(x)$: $x_i(\pi)+w_i\le x$ を満たす配置 $\pi\in\Pi_{\sigma,\mu}$ における $\max_{j\in J_i^{\mathrm{f}}}p_j(x_j(\pi))$ の最小値,

と定義する. このとき, $f_i(x)$ は, 漸化式

$$f_i(x)=\begin{cases}\min_{x_i\le x-w_i}p_i(x_i), & J_i^{\mathrm{f}}=\{i\}\ \text{のとき}\\ \min_{x_i\le x-w_i}\max\{p_i(x_i),\max_{j\in J_i^{\mathrm{f}}\setminus\{i\}}f_j(x_i)\}, & \text{それ以外}\end{cases}\tag{4}$$

と計算できる. $p_{\max}(\pi)$ の最小値は上の漸化式で求めた $f_i(x)$ を用いて $\max_{i\in I}\min_x f_i(x)$ と定まり, 各長方形の x 座標は,

$$x_i(\pi)=\begin{cases}\max\{x_i\mid p_i(x_i)=\min_{x_i'}\{p_i(x_i')\mid f_i(x_i'+w_i)=\min_x f_i(x)\}\}, & J_i^{\mathrm{b}}=\{i\}\ \text{のとき}\\ \max\{x_i\mid p_i(x_i)=\min_{x_i'}\{p_i(x_i')\mid f_i(x_i'+w_i)=\min_x\{f_i(x)\mid x\le r_i\}\}\}, & \text{それ以外}\end{cases}\tag{5}$$

と計算できる. ただし, $r_i = \min_{j \in J_i^b \setminus \{i\}} x_j(\pi)$ である. ここで求めた各長方形の x 座標は, 上述の $p_{\max}(\pi)$ の最小値を実現し, 各長方形のコスト関数を局所的に最小化している.

この計算にかかる時間は, 二分探索木をうまく用いる工夫を加えることで $O(\tau n \log n)$ 時間となり, 配置コスト関数が単調非減少であるときなど, 実用上重要ないくつかの特殊ケースに対しては, $O(n \log n)$ 時間である. (ただし, τ はコスト関数の複雑度によって定まる値である. アルゴリズムの詳細および計算時間の解析については文献 [1] を参照のこと.) この結果は, 村田らのアルゴリズム ($O(n^2)$ 時間) [4] より優れており, 高橋のアルゴリズム ($O(n \log n)$ 時間) [7] の一般化となっている.

5 局所探索法

局所探索法とは, 現在の解 (σ, μ) に対し, 少しの変形を加えることによって得られる解の集合 $N(\sigma, \mu)$(近傍と呼ばれる) 内に, (σ, μ) よりも良い解があれば, 現在の解をそれに置き換えるという操作を, 近傍内に改善解が存在しなくなるまで反復する方法である. この手法により得られる解, すなわち $N(\sigma, \mu)$ 内に改善解が存在しない解 (σ, μ) を局所最適解と呼ぶ.

局所探索法の動作を定めるには, 初期解の生成法, 近傍の定義, 解の評価法, 移動戦略, 終了の基準, などを決める必要がある. 本研究では, 初期解としてランダムに生成した順列対およびランダムに定めるモードを用い, 近傍には後で述べるシフト近傍を用いる. 解の評価には目的関数値を用いるが, この基準のみで解の評価を行った場合効率的な探索が困難であるため, 目的関数値が同じであっても, コスト関数値の総和 $\sum_{i \in I}(p_i(x_i(\pi)) + q_i(y_i(\pi)))$ が小さい解があればそちらに移動することにする. 移動戦略には, 代表的なものとして即時移動戦略と最良移動戦略があるが, 今回は前者を採用する. 終了の基準については, 最適解が得られるか, もしくはあらかじめ定めた計算時間に到達したとき探索を終了することにしている. 局所探索法や後述するメタ戦略については文献 [8] が詳しい.

5.1 近傍

局所探索を行うための近傍には, シフト近傍を用いている. シフト近傍は, σ_+ と σ_- の一方, もしくは両方の順列において, 一つの長方形の位置を移動することで得られる解の集合であり, それぞれシングルシフト近傍, ダブルシフト近傍と呼ぶ. また, ダブルシフト近傍の変形として, 両方の順列において長方形の位置を移動する際, 移動する長方形の他にもう一つの長方形を選択しておき, 長方形の移動先をあとで選択した長方形の付近に限定するという近傍も考えており, これを限定ダブルシフト近傍と呼ぶ. ダブルシフト近傍と比較して, シングルシフト近傍や限定ダブルシフト近傍は近傍のサイズ (近傍内の解の個数) が小さいため, 探索能力は劣るが高速に近傍内を探索できるという利点がある. 本研究では, さらに, 移動する長方形の選択の際に次節で定義するクリティカルパスを利用することで探索の効率化を行っている.

5.2 クリティカルパス

以下, x 軸方向についてのみ定義するが, y 軸方向も同様である. 配置 $\pi \in \Pi_{\sigma,\mu}$ に対して, 有向グラフ $G = (V, E)$ および長方形の部分集合 $S, T, \tilde{S}, \tilde{T}$ を,

$$V = I,$$

$$(i,j) \in E \iff x_i(\pi) + w_i = x_j(\pi) \ \text{かつ} \ i \preceq^x_\sigma j,$$

$$S = \{i \in I \,|\, p_i(x_i(\pi) - \varepsilon) \geq p_{\max}(\pi)\},$$

$$\tilde{S} = \{i \in S \,|\, p_i(x_i(\pi)) = p_{\max}(\pi)\},$$

$$T = \{i \in I \,|\, p_i(x_i(\pi) + \varepsilon) \geq p_{\max}(\pi)\},$$

$$\tilde{T} = \{i \in T \,|\, p_i(x_i(\pi)) = p_{\max}(\pi)\},$$

と定義する (ただし ε は十分小さな任意の正数). このとき, 始点を $s \in S$, 終点を $t \in T$ とし, $s \in \tilde{S}$ もしくは $t \in \tilde{T}$ が成り立つ G 上の有向パスをクリティカルパスと定義する. 4節で提案したアルゴリズムによって得られる配置においては, 必ず1本以上のクリティカルパスが存在し, これをこわさない限り $p_{\max}(\pi)$ の, すなわち目的関数値の改善は望めない.

5.3 メタ戦略

単純な局所探索法を用いて解を探索する場合, 計算時間は高速だが, 解の精度としては必ずしも満足のいくものが得られるわけではない. 近年, 局所探索法と比較して多少時間はかかっても, より精度の高い解を求める解法として, メタ戦略に対する要求が高まっており, 多くの組合せ最適化問題に対して大きな成功をおさめている.

本研究ではメタ戦略の中から, 反復局所探索法 (ILS 法) とタブー探索法 (TS 法) を試みている. ILS 法は多数の初期解それぞれに局所探索を適用し, 得られた最良の解を出力するものだが, 初期解の作成をランダムに行うのではなく, それまでの探索で得られた良い解をもとに作成することで, 探索の集中化を行うものである. TS 法は, 局所最適解 (すなわち, 近傍内に改善解が存在しない解) からも解の移動を強制するものであるが, 解のサイクリングの防止や, 探索の多様化のためにそれまでの探索の履歴を用いる点に特徴がある.

6 高速化

ダブルシフト近傍において, 移動する長方形 i を決定し, それを両方の順列において移動する場合, 1つの i に対して調べる解の個数は $O(n^2)$ となる. これらの解を以下の手法を用いて評価する. ここでは $p_{\max}(\pi)$ の最小値を求めるアルゴリズムのみを示すが, $q_{\max}(\pi)$ についても同様である.

まず, 移動する長方形 i を取り除いた $n-1$ 個の長方形の集合を $\tilde{I}$, 順列対を $\tilde{\sigma}$ とし, すべての $1 \leq p, q \leq n$ について,

$$J^{\mathrm{f}}_{p,q} = \{j \in \tilde{I} \,|\, \tilde{\sigma}_+(j) < p, \tilde{\sigma}_-(j) < q\},$$

$$J^{\mathrm{b}}_{p,q} = \{j \in \tilde{I} \,|\, \tilde{\sigma}_+(j) \geq p, \tilde{\sigma}_-(j) \geq q\},$$

$f_{p,q}(x)$: $\forall j \in J^{\mathrm{f}}_{p,q}$ に対して $x_j(\pi) + w_j \leq x$ を満たす $\pi \in \Pi_{\tilde{\sigma},\mu}$ の中での $\max_{j \in J^{\mathrm{f}}_{p,q}} p_j(x_j(\pi))$ の最小値,

$b_{p,q}(x)$: $\forall j \in J^{\mathrm{b}}_{p,q}$ に対して $x_j(\pi) \geq x$ を満たす $\pi \in \Pi_{\tilde{\sigma},\mu}$ の中での $\max_{j \in J^{\mathrm{b}}_{p,q}} p_j(x_j(\pi))$ の最小値,

と定義する. このとき, $f_{p,q}(x)$ は, p または q が 1 のとき 0 となり, そうでないときは漸化式

$$f_{p,q}(x) = \begin{cases} \max\{f_{p-1,q}(x), f_{p,q-1}(x)\}, & \sigma_+^{-1}(p-1) \neq \sigma_-^{-1}(q-1) \text{ のとき} \\ \min_{t \leq x-w_j}\{\max(p_j(t), f_{p-1,q-1}(t))\}, & \sigma_+^{-1}(p-1) = \sigma_-^{-1}(q-1) = j \text{ のとき} \end{cases}$$

により計算される. $b_{p,q}(x)$ の計算もほぼ同様に行うことができ, p または q が n のとき 0 となり, そうでないときは漸化式

$$b_{p,q}(x) = \begin{cases} \max\{f_{p+1,q}(x), f_{p,q+1}(x)\}, & \sigma_+^{-1}(p) \neq \sigma_-^{-1}(q) \text{ のとき} \\ \min_{t \leq x-w_j}\{\max(p_j(t), f_{p+1,q+1}(t))\}, & \sigma_+^{-1}(p) = \sigma_-^{-1}(q) = j \text{ のとき} \end{cases}$$

により計算される. また, $(\tilde{\sigma}, \mu)$ に対する x 軸コストの最大値を最小化したものを $p_{\max}(\tilde{\pi})$ と書き, 4 節のアルゴリズムを用いて計算しておく. これらを用いて, 取り除いた長方形 i を順列 $\tilde{\sigma}_+$ の α 番目, $\tilde{\sigma}_-$ の β 番目に挿入したときの $p_{\max}(\pi)$ の最小値は,

$$\max\{p_{\max}(\tilde{\pi}), \min_t \max(f_{\alpha,\beta}(t), p_i(t), b_{\alpha,\beta}(t+w_i))\} \tag{6}$$

と計算することができる.

この手法を用いると, 4 節のアルゴリズムで順列対から配置を求める計算, すなわち解を 1 つ評価する計算時間の $O(n/\log n)$ 倍の時間で $O(n^2)$ 個の解の評価が可能となり, 解一つあたりの評価時間は $O(1/n\log n)$ 倍となる. とくに, 前述の特殊ケースに対しては, 全体の計算時間が $O(n^2)$, すなわち解一つあたりの計算時間が $O(1)$ となる.

シングルシフト近傍や限定ダブルシフト近傍を考える場合は, 移動する長方形 1 つに対して, 調べる解の個数は $O(n)$ となる. これらの解の探索の際, 全ての p, q に対し前述の漸化式を用いて $f_{p,q}(x)$, $b_{p,q}(x)$ を計算すると, ダブルシフト近傍内の解を探索するのと同じオーダの計算時間がかかるため近傍を縮小する意味がなくなる.

そこで, 長方形 i を取り除いた $(\tilde{\sigma}, \mu)$ に対して 4 節のアルゴリズムを適用したときの計算結果をうまく使うことで, $f_{p,q}(x), b_{p,q}(x)$ の必要最小限のもののみ計算するという効率化を行っている. この工夫によって, 4 節のアルゴリズムを用いて 1 つの解を評価するのと同じオーダの計算時間で, 近傍内の解で取り除く長方形が i であるもの全てを評価しており, 従来のアルゴリズムと比較して $O(n)$ 倍の高速化を実現している.

7 計算実験

提案手法の性能評価のため, いくつかの数値実験を行った. 我々のアルゴリズムとしては, 近傍にシングルシフト近傍と限定ダブルシフト近傍を用いた反復局所探索法を採用しており, 探索の終了条件は, 局所探索を 100 回繰り返した時点で終了としている. 実験環境は, PC (Pentium III 1 GHz, 1 GB memory) 上で C 言語を用いている. 今回の実験では, 面積最小化問題と大きな構造物の生産スケジューリング問題の 2 つを調べた.

表 1: 面積最小化問題に対する既存のアルゴリズムと提案手法の比較

	SA-BSG		SA-SP		ILS-PRE		ILS	
問題例	充填率	計算時間	充填率	計算時間	充填率	計算時間	充填率	計算時間
ami49	97.10	69.0	96.29	176.0	96.30	100.0	93.98	1.8
rp100	97.08	68.2	88.54	248.7	95.76	200.0	94.00	12.8
pcb146	94.87	100.2	94.42	678.7	95.63	300.0	95.67	16.3
pcb500	94.10	334.6	90.82	7802.9	92.27	1000.0	94.58	197.0

7.1 面積最小化問題

与えられた長方形を, 90° 回転を許し, 互いに重ならないように全て配置したとき, それら全てを覆う長方形の面積の最小化を目的とする問題である. この問題は, 多くの研究 [4, 5, 7] で扱われている代表的な配置問題であり, 我々の定式化で扱う際は, 各長方形に縦置き, 横置きの 2 つのモードを与える. 問題例としては, 長方形数が 49 から 500 までの 4 つの問題例を扱った. これらの中には, 一般的に用いられているベンチマーク問題と, 各長方形の幅と高さをそれぞれ [1,100] の整数からランダムに選ぶことによって生成した問題例とが存在する. いずれの問題例に対しても最適解は分かっていない.

これらの問題例に対し, 我々のアルゴリズム (ILS) の性能を次の 3 つのアルゴリズムと比較する: (1) 解表現方法として限界スライスライングリッド (BSG) を, 探索方法としてアニーリング法を用いた中武らのアルゴリズム ([5], SA-BSG), (2) 解表現方法として順列対表現を, 探索方法としてアニーリング法を用いた村田らのアルゴリズム ([4], SA-SP), (3) 解表現方法として順列対表現を, 探索方法として反復局所探索法を用いた我々の従来のアルゴリズム ([1], ILS-PRE). (1), (2) のアルゴリズムは, 面積最小化問題に対する専用アルゴリズムである. また, (3) のアルゴリズムでは局所探索の近傍としてシフト近傍の他に交換近傍など複数のものを組合せた近傍を用いており, あらかじめ定めた計算時間に到達したとき探索を終了した.

面積最小化問題の 4 つの問題例について計算実験を行った結果を表 1 に示す. 表中, 問題例の名前に含まれる数字はその問題例の長方形数を表す. 充填率 (%) は 100·(与えられた長方形の面積の総和)/(全てを覆う長方形の面積) であり, 100 に近いほど性能が良い. また, 計算時間は, アルゴリズムが消費した CPU 時間 (秒) を表す.

この結果より, 提案手法は他の手法と比較して非常に高速に, 比較的良い解を求めることが可能であることが確認された. 特に, サイズの大きい問題例では, 現実的な計算時間で他の手法より良い性能を示している. しかし, 今回実験に用いた近傍のみでは, より長い計算時間を許したとしても, ある一定の精度以上の解を発見するのは困難であることが観測されている. このため, 今回用いた近傍とは異なる, 様々な近傍を組合せることで解の精度をより良くすることは今後の課題と言える.

7.2 大きな構造物の生産スケジューリング問題

大きな構造物を生産する工場におけるスケジューリング問題を考える. この工場では, 扱う構造物が大きいため, 一旦それらを工場内に配置すると, 作業が終了するまで移動することができない. そのため, 構造物の配置を含めてスケジューリングを行う必要がある. 各構造物は非常に細長い形

をしており, 各構造物について必要な一次元の作業領域と作業期間および作業を開始するべき期間が与えられる. この問題を我々の定式化で扱うために, 各構造物に対して, 必要な作業領域を高さ, 作業期間を幅とする長方形を考え, これを二次元平面上に配置する. 各長方形の x 方向のコスト関数 $p_i(x)$ は, 開始するべき期間からはずれると, 線形のコストがかかるように設定している. また, y 方向のコスト関数は, 利用可能な作業領域からはずれると, 十分に大きい線形のコストがかかるように設定している. 問題例としては, 構造物の数が 50 から 100 までの 5 つの問題例を用い, 特に構造物数が 78 の問題例は, 現実のスケジューリング問題からの問題例である. 今回扱った全ての問題例について, 実行可能なスケジュールが存在することが分かっている.

これらの問題例に対し, 野々部らのアルゴリズム [6] との比較を行った. ただし, [6] のアルゴリズムは資源制約付きプロジェクトスケジューリング問題に対する汎用アルゴリズムである. この実験に対する詳細な計算結果は省略するが, 提案手法では全ての問題例に対し, 野々部らのアルゴリズムが必要とする計算時間よりも短い時間で実行可能なスケジュールを発見することができ, こちらの問題に対しても, 提案手法が良い性能を示すことが確認された.

8 おわりに

我々は以前, 汎用的な問題である配置コストをもつ長方形詰込み問題を提案し, この問題に対する局所探索法に基づく近似解法の提案を行った. 本研究では, このアルゴリズムに対する高速化手法を提案し, 面積最小化を目的とした配置問題や, 現実に現れるスケジューリング問題を用いて, その効果を検証した. 数値実験の結果より, 我々の提案する高速化手法が十分に有効であること, そして, 各問題に対する専用アルゴリズムと比較しても満足の行く性能を発揮することが確認できた.

今回は, 2 つの問題に対する数値実験を行うにとどまったが, 我々の定式化で扱うことのできるより多くの問題に対して数値実験を行うことで, 提案手法の汎用性と性能の高さを示したいと考えている. また, ここで紹介した近傍以外にも, 様々な近傍に対して我々の提案した高速化手法は有効である. そのような近傍を, 様々なメタ戦略と組合せたアルゴリズムの開発, および数値実験による有効性の検証も今後の課題である.

謝辞

計算実験を行うにあたって, 面積最小化問題の問題例と計算実験の結果を提供して下さった北九州市立大学の 梶谷 洋司氏, 大阪大学の 坂主 圭史氏, 大きな構造物の生産スケジューリング問題の問題例と計算実験の結果を提供して下さった京都大学の 野々部 宏司氏に深く感謝致します.

参考文献

[1] S. Imahori, M. Yagiura and T. Ibaraki, "Local Search Algorithms for the Rectangle Packing Problem with General Spatial Costs," submitted for publication.

[2] D. Liu and H. Teng, "An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles," *European Journal of Operational Research*, 112, pp.413–420, 1999.

[3] A. Lodi, S. Martello, D. Vigo, "Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems," *INFORMS Journal on Computing*, 11, pp.345-357, 1999.

[4] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair," *IEEE Transactions on CAD*, 15, pp.1518–1524, 1996.

[5] S. Nakatake, K. Fujiyoshi, H. Murata and Y. Kajitani, "Module placement on BSG-structure and IC layout applications," *Proceedings of International Conference on Computer Aided Design*, pp.484–491, 1996.

[6] K. Nonobe and T. Ibaraki, "Formulation and tabu search algorithm for the resource constrained project scheduling problem," in: *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers, pp.557–588, 2001.

[7] T. Takahashi, "An Algorithm for Finding a Maximum-Weight Decreasing Sequence in a Permutation, Motivated by Rectangle Packing Problem (in Japanese)," *Technical Report of IEICE*, VLD96-30, pp.31–35, 1996.

[8] 柳浦睦憲, 茨木俊秀, 組合せ最適化 —メタ戦略を中心として—, 朝倉書店, 2001.