数理解析研究所講究録
第1641巻 2009年 114-124

適応的計算幾何
Adaptive Computational Geometry

安 熙甲　　岡本 吉央
Hee-Kap Ahn *　　Yoshio Okamoto †

概 要

適応的計算幾何の研究を行う．適応的アルゴリズムは整列問題に対して精力的に行われて来ているが，本論ではその枠組みを幾何問題へ一般化する．そのため，幾何問題を配列の置換問題 (すなわち再配置問題) と見なし，「整列度」を入力配列から所望の出力配列への距離として定義する．アルゴリズムが**適応的**であるとは，与えられた入力配列が所望の出力に近いとき早く実行終了し，しかも整列度自体の情報を何も用いないこととする．ケース・スタディとして，2次元凸包計算と2次元 kd 木構成を考察する．

1 はじめに

計算幾何は数値的な問題 (すなわち，1次元の問題) を高次元へ拡張したものであると見なすことができる．典型例は2次元凸包計算であり，数字配列の整列問題を2次元へ一般化したものと見なせる．

本研究の動機は整列問題のアルゴリズム解析に対して「整列度」を考慮する先行研究にある．すなわち，与えられた入力がほとんど整列しているとき整列アルゴリズムの実行が早く終了することを期待するのである．Mehlhorn [23] はそのような性質を持つ整列アルゴリズムに対して「適応的整列アルゴリズム[1]」という用語を導入した．適応的整列アルゴリズムの最悪時解析に対する形式的な枠組みを導入したのは Mannila [22] であり，そのサーベイが Estivill-Castro and Wood [15] である．

*浦項工科大学校，韓国 (Pohang University of Science and Technology, Korea)
†東京工業大学，日本 (Tokyo Institute of Technology, Japan)

[1] Knuth [18] の日本語版 (p. 370) では「適合性のあるソート法」と訳されている．

適応的整列アルゴリズムにはいくつか特徴がある．1つ目は，整列度が高いときに実行が早く終了することである．2つ目は，アルゴリズムが整列度に関するどんな情報も用いないことである．それが「適応的」であると呼ばれる理由である．

本研究では**適応的計算幾何**の研究をする．適応的アルゴリズムの枠組みを幾何問題に適用するため，幾何問題を置換問題として捉えたい．すなわち，(点，線分などの) 幾何的対象の配列が与えられたとき，所望の解答を表現する配列の置換 (すなわち再配置) を出力するのである．整列問題は自然に置換問題と見なすことができ，2次元凸包問題も置換問題と見なすことができる．(実際，凸包アルゴリズムの下界は整列問題への還元によって与えられる．) 更に，内部幾何アルゴリズムに関する最近の研究は幾何問題のいくつかを置換問題として取り扱っている [7, 9, 10].

様々な「整列度」が先行研究では提案されている [15]．本研究では，最も古くから最も頻繁に用いられる反転の数を考える．集合 X 上の線形順序 $<$ と X の置換 π (すなわち，X 上の全単射) が与えられたとき，**反転**とは順序対 $(i,j) \in X^2$ で，$i < j$ と $\pi(i) > \pi(j)$ を満たすもののことである．π における反転の総数を $\mathsf{inv}_<(\pi)$ で表すが，線形順序 $<$ が文脈から明らかな場合は $\mathsf{inv}(\pi)$ という省略記法を用いる．置換 π が順序 $<$ に従うための必要十分条件が $\mathsf{inv}_<(\pi) = 0$ であることに注意する．したがって，反転の数は整列度として適切であると見ることができる．

ケース・スタディとして，次の2つの幾何問題を考える．最初の問題は2次元凸包計算である．すなわち，平面上に与えられた点集合に対して，その凸包を計算するのである．第2の問題は2次元 kd 木構成である．すなわち，平面上に与えられた点集合に対して，その kd 木を計算するのである．この2つの問題が $O(n(1 + \log(1 + k)))$ 時間で解けることを示す．ただし，k は与えられた n 個の点から成る配列が所望の出力に対して持つ反転の数である．$k \leq \binom{n}{2}$ なので，この計算量は $O(n \log n)$ である．したがって，n に関して最悪時最適である．

関連研究　適応的整列問題を議論している論文は多く存在する．Estivill-Castro and Wood [15] のサーベイは一読の価値がある．適応的整列問題は整数整列 [26] や入出力効率性に関しても (キャッシュを意識する場合とキャッシュを意識しない場合ともに) [8] 議論されている．

適応性の枠組みを整列以外の問題に適用している論文もいくつかある．Demaine, López-Ortiz, and Munro [11] は整列集合に対する集合演算を考

察し，問題の困難さを表すある指標に関する適応的アルゴリズムを与えている．これはデータベースの応用が動機となっていて，この方向性での研究を続けた論文もある [4, 3].

他の研究の方向は実を言うと適応的計算幾何である．適応的計算幾何に初めて言及した論文は Levcopoulos, Lingas, and Mitchell [19] である．しかし，彼らは問題を置換問題として見ることはせず，また，彼らの用いた整列度は入力と出力を比較しているわけではない．それは Baran and Demaine [1] や Barbay and Chen [2] も同じである．すなわち，整列問題の味わいを持つ多次元問題の基礎理論を築くために適応的計算幾何を研究するのは本研究が初めてである．

皆さんは適応的アルゴリズムと固定パラメータアルゴリズム [25] の違いに疑問を持つかもしれない．固定パラメータアルゴリズムはパラメータ自身を用いてもよいが，適応的アルゴリズムはパラメータを知る必要がない．

記法　要素数 n の配列 A の添字は $1,\ldots,n$ であるとする．A の第 i 要素を $A[i]$ で表す ($i \in \{1,\ldots,n\}$)．$A[i],\ldots,A[j]$ から成る A の部分配列を $A[i..j]$ で表す．A の部分集合 A' に対して差 $A \setminus A'$ は $A \setminus A'$ の要素を A で並んでいた順番のまま並べた配列を表わす．2つの配列 A と B の (この順での) 連結を $A \circ B$ で表す．点の集合 (または配列) P に対して，$\mathrm{conv}(P)$ で P の凸包を表し，$\partial\mathrm{conv}(P)$ で $\mathrm{conv}(P)$ の境界を表す．

2　2次元凸包

非形式的に言うと，**2次元凸包問題**では一般の位置にある n 個の点の集合 P が与えられたとき (ただし，点集合が一般の位置にあるとは，どの3点も1直線上になくどの2点の x 座標も同じではないこととする)，P の凸包の境界上の点を特定する必要がある．より形式的に言うと，この問題では平面上で一般の位置にある n 個の点の配列 P が与えられて，P を次のように並び替えた配列 Q を出力する．まず，$Q[1]$ は P の最も左にある点で，h を $\partial\mathrm{conv}(P)$ 上の点の数とするとき，$Q[1..h]$ はこれらの点を時計回り順に並べたものとする．そして，$Q[h{+}1..n]$ は $\mathrm{conv}(P)$ の内部の点を x 座標に従って並べたものであるとする．これによって，P 上の線形順序 $<_P$ を「Q は順序 $<_P$ に従って整列したもの」として定義できる．図1の

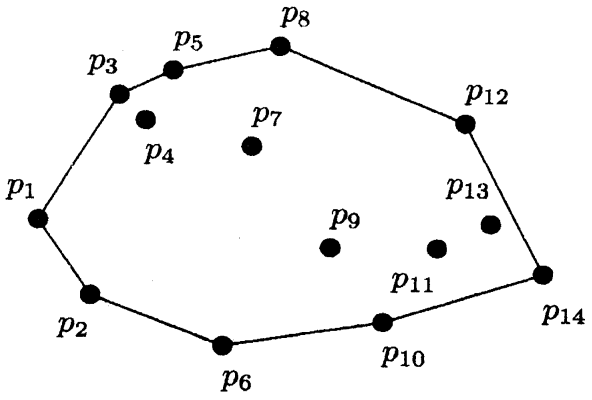

図 1: 2 次元凸包.

例において，出力配列は $[p_1, p_3, p_5, p_8, p_{12}, p_{14}, p_{10}, p_6, p_2, p_4, p_7, p_9, p_{11}, p_{13}]$ となる.

適応的凸包アルゴリズムを設計する際に，少なくとも次のような 2 つの困難が見当たる．まず 1 点目として，2 点 p, q を見るだけで $p <_P q$ かどうかを決定できず，それは p, q の周りに他の点がどのように置かれているかに依存する．2 点目は 1 点目に関連するが，分割統治法に従って進み，P の部分集合 P' が得られたとき，$<_{P'}$ が $<_P$ の P' への制限であるとは限らない．すなわち，線形順序が継承されないのである．この 2 つの問題点は整列問題には現れなかったことに注意する.

本研究で提案するアルゴリズムはこれらの問題点を克服し，適応的であることが証明できる．ConvexHull(P) への呼び出しによって，conv(P) の上側鎖 $U(P)$ を x 座標の昇順，conv(P) の下側鎖 $U(P)$ を x 座標の降順，conv(P) の内部にある点の集合 $I(P)$ を x 座標の昇順で計算できる．所望の出力は連結 $U(P) \circ L(P) \circ I(P)$ である.

アルゴリズム： ConvexHull(P)

Step 1: P が所望の出力の通り並んでいるとき，$U(P), L(P), I(P)$ を特定して終了.

Step 2: そうでないとき，P の鉛直二等分線 s を計算する．s の左側にある P の点の集合を P_A とし，右側にある P の点の集合を P_B とする.

Step 3: conv(P_A) と conv(P_B) の上側共通接線 u と下側共通接線 ℓ を計算する．u を張る (唯一の) 2 点を $a_u \in P_A$ と $b_u \in P_B$ とする．同様に，ℓ を張る (唯一の) 2 点を $a_\ell \in P_A$ と $b_\ell \in P_B$ とする．P_L を a_u, a_ℓ が張る直線の左側にある P の点の集合とし，P_R を b_u, b_ℓ が張

る直線の右側にある P の点の集合とし，$P_M = P \setminus (P_L \cup P_R)$ とする．$P_L \subseteq P_A$ と $P_R \subseteq P_B$ に注意する．

Step 4: $U(P_L)$, $L(P_L)$, $I(P_L)$, $U(P_R)$, $L(P_R)$, $I(P_L)$ を得るために再帰的に ConvexHull(P_L) と ConvexHull(P_R) を実行する[2]．$U(P) = U(P_L) \circ U(P_R)$ かつ $L(P) = L(P_R) \circ L(P_L)$ とする．

Step 5: P_M の点を x 座標に関して整列し，配列 $S(P_M)$ を得る．そして，$I(P_L), S(P_M), I(P_R)$ を併合し，その結果を $I(P)$ とする．終了．

このアルゴリズムは Kirkpatrick and Seidel [17] の分割統治アルゴリズムに似ている．彼らのアルゴリズムでは上側包と下側包を別々に計算するが，ここでそうしてしまうと適応的にならない．むしろ，ここでは一斉に計算する．アルゴリズムの正当性は簡単に示される．

それでは，実行時間を見積もる．ここからは n で入力点の数，k で入力点配列 P と所望の出力間の反転の数 (すなわち，所望の出力が Q のときの $\mathsf{inv}_{<_P}(Q)$) とする．

Step 1 は次のようにすれば $O(n)$ で実行できる．まず，P の最も左にある点 p と最も右にある点 q を $O(n)$ 時間で特定する．所望の出力にて p は最初に来なくてはならず ($p = P[1]$)，q はどこか，例えば h' 番目の位置に来なくてはならない ($q = P[h']$) とする．次に，部分配列 $P[1..h']$ が凹鎖 C_u であるかどうかを連続する 3 点の曲がり方をすべて見ることで確認する．これは $O(n)$ 時間で行える．その次に，P の下側包の左から 2 つ目の点 p' を $O(n)$ 時間で特定し，$p' = P[h]$ を満たす添字 $h > h'$ を発見する．そして，部分配列 $P[h'..h]$ と p が凸鎖 C_ℓ であるか $O(n)$ 時間で確認する．ここで，$P[h{+}1..n]$ の点は $\mathrm{conv}(P)$ の内部になくてはならず，x 座標に関して整列されていなくてはならない．まず，整列されているかどうかは $O(n)$ 時間で確認できる．そして各点 $r \in P[h{+}1..n]$ に対して左から右へ順番に r が凹鎖 C_u と凸鎖 C_ℓ の間にあるか確認する．すべてが整列しているので，これは $O(n)$ 時間で行える．最後に，$U(P) = P[1..h']$, $L(P) = P[h'{+}1..h]$, $I(P) = P[h{+}1..n]$ とする．これで Step 1 が $O(n)$ 時間で実行できた．

Step 2 は中央値発見問題に帰着され，$O(n)$ 時間で解くことができる [6]．

Step 3 において，上側共通接線と下側共通接線を計算する部分は (Kirkpatrick and Seidel [17] と同様に) 2 次元線形計画法に帰着され，$O(n)$ 時間

[2]境界に些細な部分があるけれども，簡単な修正で対処できる．

で解くことができる [24]. また, P_L と P_R を見つけることも簡単に $O(n)$ 時間で可能である. ここで $|P_L|, |P_R| \leq n/2$ に注意する.

Step 4 では再帰呼び出しを行う. 重要な観察は $\partial\mathrm{conv}(P)$ 上にある P の点は $\partial\mathrm{conv}(P_L)$ か $\partial\mathrm{conv}(P_R)$ 上にあり, $\partial\mathrm{conv}(P_L)$ 上にある P_L の点 (と $\partial\mathrm{conv}(P_R)$ 上にある P_R の点) は $\partial\mathrm{conv}(P)$ 上にあるということである. ゆえに, $\mathrm{ConvexHull}(P_L)$ に対する所望の出力 Q_L と $\mathrm{ConvexHull}(P_R)$ に対する所望の出力 Q_R は Q の部分配列である. このようにして, 先に述べた困難を克服することができる. 入力配列 P が $|P| = n$ と $\mathsf{inv}_{<_P}(Q) = k$ (Q は所望の出力) を満たすとき $\mathrm{ConvexHull}(P)$ の最悪時間計算量を $t(n,k)$ で表すと, Step 4 は高々 $t(|P_L|, k_L) + t(|P_R|, k_R)$ 時間しかかからない. ただし, k_L, k_R はそれぞれ P_L, P_R と Q (の P_L, P_R への制限) の間の反転の数である. P_L, P_R, P_M は P の部分列で互いに共通要素を持たないので, 次の補題が成り立つ.

補題 2.1. *k_L, k_R, k_M がそれぞれ P_L, P_R, P_M と Q の間の反転の数であるとすると, $k_L + k_R + k_M \leq k$ が成り立つ.*

Step 5 では P_M の点を整列する. 例えば Estivill-Castro and Wood [14] の適応的整列アルゴリズムを用いれば P_M の整列が $O(|P_M|(1+\log(1+k_M)))$ 時間でできる. 更に, $I(P_L), I(P_R), S(P_M)$ はどれも整列済みであるので, 併合も標準的な方法によって $O(n)$ 時間でできる.

それでは, 今までの議論を総括して全体の実行時間を見積もる. Step 1, 2, 3 はある定数 a と十分大きな n に対して an 時間かかるとし, Step 5 はある定数 b と十分大きな n に対して $b|P_M|(1 + \log_2(1 + k_M))$ 時間かかるとする. P_L, P_R, P_M の点の数をそれぞれ n_L, n_R, n_M と表わすことにすると ($n_L + n_R + n_M = n$ に注意), 次の再帰式が得られる. すなわち, 十分大きな任意の n と $k \geq 1$ に対して

$$t(n,k) \leq an + t(n_L, k_L) + t(n_R, k_R) + bn_M(1 + \log_2(1 + k_M)).$$

(小さな n に対しては $t(n,k) = O(1)$ となり, $k \leq 0$ に対しては $t(n,k) = O(n)$ となることに注意する[3].) ここから, ある定数 c と十分大きなすべての n に対して $t(n,k) \leq cn(1 + \log_2(1 + k))$ が成り立つことを示す.

帰納法により,

$$\begin{aligned} t(n,k) \leq\ & an + cn_L(1 + \log_2(1 + k_L)) + cn_R(1 + \log_2(1 + k_R)) \\ & + bn_M(1 + \log_2(1 + k_M)) \end{aligned}$$

[3]すなわち, これは Step 1 の計算量である.

が得られる．定数cは$2b \leq c$を満たすように選ぶ．$n_L = \alpha n$かつ$n_R = \beta n$とすると，$0 \leq \alpha \leq 1/2$，$0 \leq \beta \leq 1/2$，$n_M = (1-\alpha-\beta)n$が成り立つ．ここで，αとβは自由に選べるパラメータではなく，入力によって定まる数であることに注意する．これで再帰式は

$$\begin{aligned} t(n,k) &\leq an + c\alpha n(1+\log_2(1+k_L)) + c\beta n(1+\log_2(1+k_R)) \\ &\quad + \frac{c}{2}(1-\alpha-\beta)n(1+\log_2(1+k_M)) \\ &= an + c(\alpha+\beta+\frac{1-\alpha-\beta}{2})n \\ &\quad + cn\log_2(1+k_L)^{\alpha}(1+k_R)^{\beta}(1+k_M)^{(1-\alpha-\beta)/2} \\ &\leq an + cn + cn\log_2(1+k_L)^{\alpha}(1+k_R)^{\beta}(1+k_M)^{(1-\alpha-\beta)/2} \end{aligned}$$

となる．

ここで，最後の対数の真数$(1+k_L)^{\alpha}(1+k_R)^{\beta}(1+k_M)^{(1-\alpha-\beta)/2}$がいつ最大になるのか知りたい．更に対数を取ると，これの最大化はα, βを変数とする次の線形計画問題になる．

$$\begin{aligned} &\text{maximize} && \alpha\log_2(1+k_L) + \beta\log_2(1+k_R) + \frac{1-\alpha-\beta}{2}\log_2(1+k_M) \\ &\text{subject to} && 0 \leq \alpha, \beta \leq 1/2. \end{aligned}$$

これは直接解くことができて，次の4つに場合分けされる．変数αの係数を$A = \log_2(1+k_L) - \frac{1}{2}\log_2(1+k_M)$とし，$\beta$の係数を$B = \log_2(1+k_R) - \frac{1}{2}\log_2(1+k_M)$とする．

Case 1: $A \geq 0$かつ$B \geq 0$のとき，$\alpha = \beta = 1/2$は最適解であり，最適値は$(\log_2(1+k_L) + \log_2(1+k_R))/2$である．

Case 2: $A \geq 0$かつ$B < 0$のとき，$\alpha = 1/2, \beta = 0$は最適解であり，最適値は$(2\log_2(1+k_L) + \log_2(1+k_M))/4$である．

Case 3: $A < 0$かつ$B \geq 0$のとき，$\alpha = 0, \beta = 1/2$は最適解であり，最適値は$(2\log_2(1+k_R) + \log_2(1+k_M))/4$である．

Case 4: $A < 0$かつ$B < 0$のとき，$\alpha = \beta = 0$は最適解であり，最適値は$(\log_2(1+k_M))/2$である．

上の4つの場合のそれぞれに対して，$t(n,k)$ の見積もりを行う．Case 1 のときは

$$\begin{aligned} t(n,k) &\leq (a+c)n + cn\log_2(1+k_L)^{1/2}(1+k_R)^{1/2} \\ &\leq (a+c)n + cn\log_2 \frac{(1+k_L)+(1+k_R)}{2} \\ &\leq (a+c)n + cn\log_2 \frac{2+k}{2} \\ &\leq (a+c)n + cn\log_2(\frac{3}{4}(1+k)) \\ &= (a+c)n + (\log_2 \frac{3}{4})cn + cn\log_2(1+k) \end{aligned}$$

が得られる．ここで第2の不等式で相加平均と相乗平均の関係，第3の不等式で補題，最後から2つ目の不等式で $k \geq 1$ に対する $\frac{2+k}{2} \leq \frac{3}{4}(1+k)$ という不等式を用いた．したがって，c が $a+(1+\log_2 \frac{3}{4})c \leq c$ を満たすように選べば，望み通り $t(n,k) \leq c(n+\log_2(1+k))$ が得られる ($\log_2 \frac{3}{4} \approx -0.415037$ に注意)．

Case 2 のときは

$$\begin{aligned} t(n,k) &\leq (a+c)n + cn\log_2(1+k_L)^{1/2}(1+k_M)^{1/4} \\ &\leq (a+c)n + cn\log_2(1+k_L)^{1/2}(1+k_M)^{1/2} \end{aligned}$$

となり，Case 1 と同じように進めればよい．Case 3 も同様である．Case 4 のときは

$$\begin{aligned} t(n,k) &\leq an + cn + cn\log_2(1+k_M)^{1/2} \\ &\leq an + cn + cn\log_2(1+k_M/2) \end{aligned}$$

となり，同じように進めればよい．このようにして，4つの場合すべてに対して $t(n,k) \leq cn(1+\log_2(1+k))$ となることが分かった．この議論をまとめて次の定理が得られる．

定理 2.2. 上のアルゴリズム ConvexHull は平面上で一般の位置にある点集合に対して $O(n(1+\log(1+k)))$ 時間で凸包を計算する．ただし，n は与えられた点の数で，k は入力と所望の出力の間の反転の数である．

3 kd木の構成

平面上で与えられた点集合の kd 木 [5] は点集合の置換であると見なせる．すなわち，根が最初に来て，左部分木が再帰的に次に来て，その後

に右部分木が再帰的に来るのである [9]. 平面上の n 個の点の配列が与えられたとき，それが kd 木を表現しているかどうかは $O(n)$ 時間で確認できる (詳細は省略). この手続きと kd 木を構成する通常の再帰的アルゴリズムを組み合わせることで，次の定理が得られる (詳細は再び省略).

定理 3.1. 平面上で一般の位置にある n 個の点の配列が与えられたとき，その kd 木を $O(n(1+\log(1+k)))$ 時間で計算できる. ただし，k は入力と所望の出力の間の反転の数である.

4 展望

整列問題に対しては $O(n(1+\log(1+k/n)))$ 時間アルゴリズムが多く提案されている [12, 13, 20, 21, 22, 23]. そして，タイトな下界も知られている [16]. この下界は本研究が扱った問題 (2 次元凸包計算，2 次元 kd 木構成) にも適用されるが，本研究のアルゴリズムはこの限界に達していない. 最適アルゴリズムの発見が望まれる. 出力依存適応的凸包アルゴリズムについてはどうだろうか？他の整列度についてはどうだろうか？置換問題と見なせる他の幾何問題についてはどうだろうか？多くの問題がまだ解かれておらず，この方向の研究が興味深く進んでいくことを期待する.

謝辞 本研究の主要部分は第 2 著者が 2008 年 5 月に浦項工科大学校を訪問したときに行なわれた. 浦項工科大学校の厚い支援に感謝する. また，実りある議論に対して Sang-Sub Kim, Iris Reinbacher, Wan-Bin Son にも感謝する.

参考文献

[1] I. Baran and E.D. Demaine. Optimal adaptive algorithms for finding the nearest and farthest point on a parameric black-box curve. International Journal of Computational Geometry and Applications **15** (2005) 327–350.

[2] J. Barbay and E.Y. Chen. Convex hull of the union of convex objects in the plane: an adaptive analysis. Proc. 20th CCCG (2008) 47–51.

[3] J. Barbay, A. Golynski, J.I. Munro, and S.S. Rao. Adaptive searching in succinctly encoded binary relations and tree-structured documents. Theoretical Computer Science **387** (2007) 284–297.

[4] J. Barbay and Kenyon. Adaptive intersection and t-threshold problems. Proc. 13th SODA (2002) 390–399.

[5] J.L. Bentley. Multidimensional binary search trees used for associative searching. Communications of the ACM **18** (1975) 509–517.

[6] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest, R.E. Tarjan. Time bounds for selection. Journal of Computer and System Sciences **7** (1972) 448–461.

[7] P. Bose, A. Maheshwari, P. Morin, J. Morrison, M. Smid, J. Vahrenhold. Space-efficient geometric divide-and-conquer algorithms. Computational Geometry: Theory and Applications **37** (2007) 209–227.

[8] G.S. Brodal, R. Fagerberg, and G. Moruz. Cache-aware and cache-oblivious adaptive sorting. Proc. 32nd ICALP (2005) 576–588.

[9] H. Brönnimann, T.M. Chan, E.Y. Chen. Towards in-place geometric algorithms and data structures. Proc. 20th SCG (2004) 239–246.

[10] H. Brönnimann, J. Iacono, J. Katajainen, P. Morin, J. Morrison, G.T. Toussaint. Space-efficient planar convex hull algorithms. Theoretical Computer Science **321** (2004) 25–40.

[11] E.D. Demaine, A. López-Ortiz, J.I. Munro. Adaptive set intersections, unions, and differences. Proc. 11th SODA (2000) 743–752.

[12] A. Elmasry. Priority queues, pairing, and adaptive sorting. Proc. 29th ICALP (2002) 183–194.

[13] A. Elmasry, M.L. Fredman. Adaptive sorting: an information theoretic perspective. Acta Informatica **45** (2008) 33–42.

[14] V. Estivill-Castro, D. Wood. Practical adaptive sorting. Proc. ICCI'91 (1991) 47–54.

[15] V. Estivill-Castro, D. Wood. A survey of adaptive sorting algorithms. ACM Computing Surveys **24** (1992) 441–476.

[16] L.J. Guibas, E.M. McCreight, M.F. Plass, J.R. Roberts. A new representation of linear lists. Proc. 9th STOC (1977) 49–60.

[17] D.G. Kirkpatrick, R. Seidel. The ultimate planar convex hull algorithm? SIAM Journal on Computing **15** (1986) 287–299.

[18] D.E. Knuth. The Art of Computer Programming, Second Edition.

Vol. 3, Sorting and Searching. 1998, Addison Wesley, Reading.

[19] C. Levcopoulos, A. Lingas, and J.S.B. Mitchell. Adaptive algorithms for constructing convex hulls and triangulations of polygonal chains. Proc. 8th SWAT (2002) 80–89.

[20] C. Levcopoulos, O. Petersson. Splitsort—An adaptive sorting algorithm. Information Processing Letters **39** (1991) 205–211.

[21] C. Levcopoulos, O. Petersson. Adaptive Heapsort. Journal of Algorithms **14** (1993) 395–413.

[22] H. Mannila. Measures of presortedness and optimal sorting algorithms. IEEE Transactions on Computers **34** (1985) 318–325.

[23] K. Mehlhorn. Data Structures and Algorithms, Vol. 1, Sorting and Searching. Springer-Verlag, Berlin Heidelberg, 1984.

[24] N. Megiddo. Linear programming in linear time when the dimension is fixed. Journal of the ACM **31** (1984) 114–127.

[25] R. Niedermeier. Invitation to Fixed-Parameter Algorithms. Oxford University Press, 2006.

[26] A. Pagh, R. Pagh, and M. Thorup. On adaptive integer sorting. Proc. 12th ESA (2004) 556–579.