Algoritmy 95

Int. Conference Algorithms'95, Nizke Tatry, Slovak Republik, pp.221-226, 1995.

Algoritmy se strukturální složitostí menší než optimální aneb K čemu je paralelní programování?

Václav Skala[1]
Katedra informatiky a výpočetná techniky
Západočeská univerzita
Americká 42, Box 314
306 14 Plzeň
Skala@kiv.zcu.cz

1. Úvod

Složitost algoritmu je významným faktorem při návrhu jednotlivých algoritmů. Vzhledem k rostoucím požadavkům na rychlost zpracování se stále více objevují algoritmy využívající možnost paralelního programování. Náklady na zpracování úlohy však velmi rychle rostou a skutečné porovnání výkonnosti systémů ukazuje, že urychlení s počtem procesorů neroste lineárně. V mnoha případech se nevyužívá možnosti předzpracování, které vede k podstatnému urychlení v průběhu zpracování. V mnoha případech vede předzpracování nejen k vlastnímu urychlení, ale též i ke snížení strukturální složitosti algoritmů. Je nutné poznamenat, že menší složitost algoritmu ještě negarantuje rychlejší vlastní výpočet.

Algoritmy ořezávání přímky konvexním n-úhelníkem a test bod uvnitř n-úhelníka jsou základními algoritmy v oblasti počítačové grafiky.

2. Bod uvnitř konvexního n-úhelníka

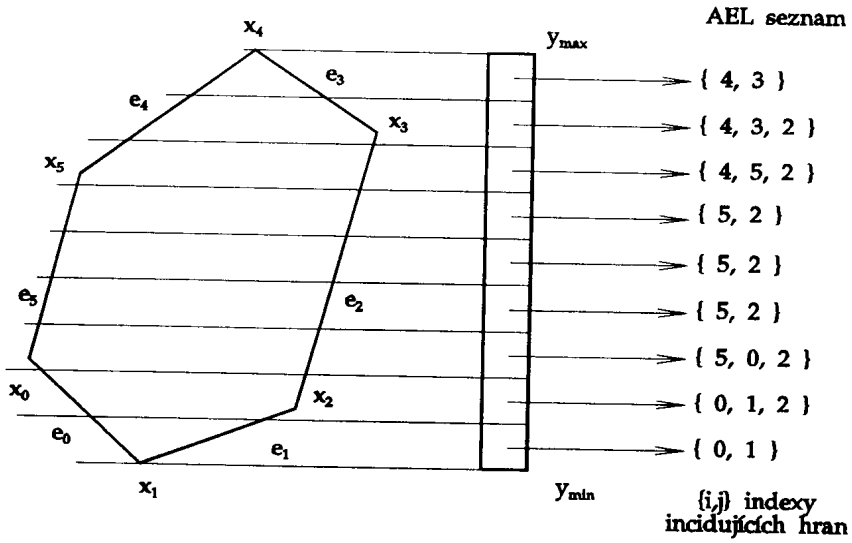
Algoritmus bod uvnitř konvexního n-úhelníka je známou úlohou, která se velmi často objevuje v literatuře. Známá řešení jsou složitosti O(N) a O(lg N), kde N je počet hran daného konvexního n-úhelníka a v [2],[4] lze nalézt porovnání výkonnosti jednotlivých algoritmů. Vzniká však otázka, zda je možné využít informace, že konvexní n-úhelník se nemění a je pro mnoho testovaných bodů neměnný. V tomto případě je evidentně výhodné předzpracování, které vede až k algoritmu s **očekávanou složitostí O(1)**, viz [3].

Princip algoritmu je v podstatě jednoduchý, a to:

- konvexní n-úhelník "nařežeme" na tenké vodorovné pásy tak, že v ideálním případě pás protíná právě dvě hrany, viz obr.1,
- ze známé y-ové souřadnice určíme index vodorovného pásu v němž daný bod leží a x - ová souřadnice pak slouží k určení, zda daný bod leží uvnitř, či vně daného n-úhelníka.

[1]Podporováno v rámci grantu ZČU č.151a společného projektu CHARM s CMEST-IST, Lisabon, Portugalsko

Vzhledem k tomu, že index vodorovného pásu odpovídajícího souřadnici y je možné určit v jednom kroku nezávisle na počtu hran daného n-úhelníka včetně určení, zda bod je či není uvnitř n-úhelníka, je očekávaná složitost algoritmu O(1).



Obr.1.

Situace je však poněkud komplikovanější, neboť vzhledem ke konečné "tloušce" vodorovných pásů se může stát, že do pásu spadnou až čtyři hrany n-úhelníka, které je nutné testovat. Toto je však vyjímečná situace. Vlastní algoritmus je pak po vybudování datové struktury při předzpracování velmi jednoduchý, viz alg.1.

```
function POINT_IN_POLYGON (x: vector): boolean;
begin { O(1) processing time }
      POINT_IN_POLYGON := false;
      if y ∉ < ymin , ymax > then EXIT; { bod vně n-úhelníka, nad nebo pod}
      { urči odpovídající pruh, tj. index i }
      i := (ymax - y) * δinv; {δinv je nenulová šíře pásu }
      { nyní se otestují pouze dvě hrany z Active Edge List AEL }
      { Fi,j (x) by mělo být realizováno jako makro; i-index pásu, j-člen seznamu}
      if F i,1 (x) < 0 then EXIT; { EXIT z procedury }
      if F i,2 (x) < 0 then EXIT;
      POINT_IN_POLYGON := true; { bod x je uvnitř }
end { POINT_IN_POLYGON };
```

Algoritmus 1

Uvedený algoritmus je zjednodušený a podrobnosti včetně porovnání s ostatními algoritmy lze nalézt v [3]. Dosažené výsledky při testování pak ukazují konstatní výpočetní čas nezávisle na počtu hran daného konvexního n-úhelníka.

3. Algoritmus ořezávání přímky v E^2

Algoritmus ořezávání přímky konvexním n-úhelníkem v E^2 je typickým problémem počítačové grafiky, jehož složitost byla, je a bude předmětem mnoha studií. Vlastní algoritmy používané v praxi jsou více či méně výkonné, většinou se složitostí O(N). Nicméně bylo z výpočtové geometrie známo, že optimální algoritmus by měl mít složitost O(lg N). Takový algoritmus byl odvozen a porovnání jeho výpočetní výkonnosti lze nalézt v [2], [4]. Z povahy řešeného problému a způsobu použití vyplývá, že ořezávací n-úhelník je víceméně konstatní pro mnoho přímek, či úseček, které se ořezávají.

Na základě principu duality, viz [1], je známo, že test bod uvnitř konvexního n-úhleníka je ekvivalentní testu, zda přímka protíná konvexní n-úhelník. Je tedy nasnadě, že test, zda přímka n-úhelník protíná, či ne, lze provést s očekávanou složitostí O(1) při využití principu duality a testu uvedeného výše. Vzniká tedy základní otázka, zda je možné nalézt též hrany, které jsou přímkou protnuty s očekávanou složitostí O(1).

Každou přímku lze vyjádřit jako

$$ax + by + c = 0$$

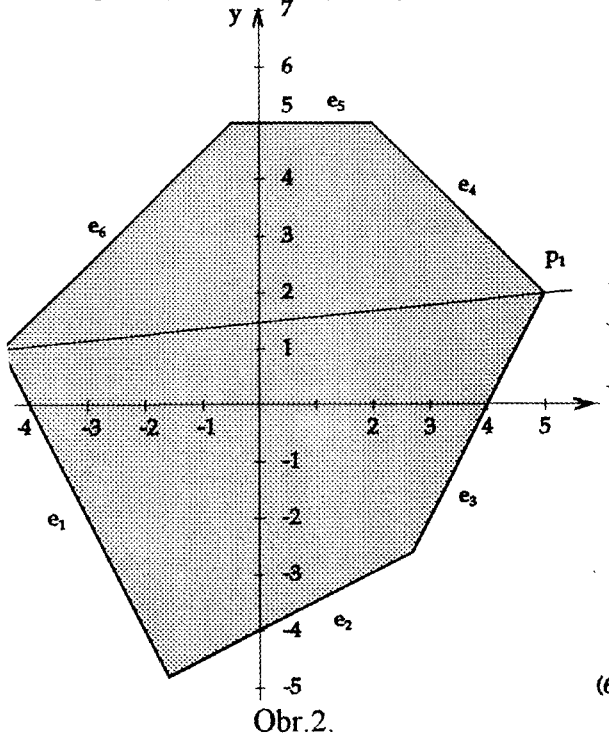
nebo

$$y = kx + q \quad k \neq \infty$$

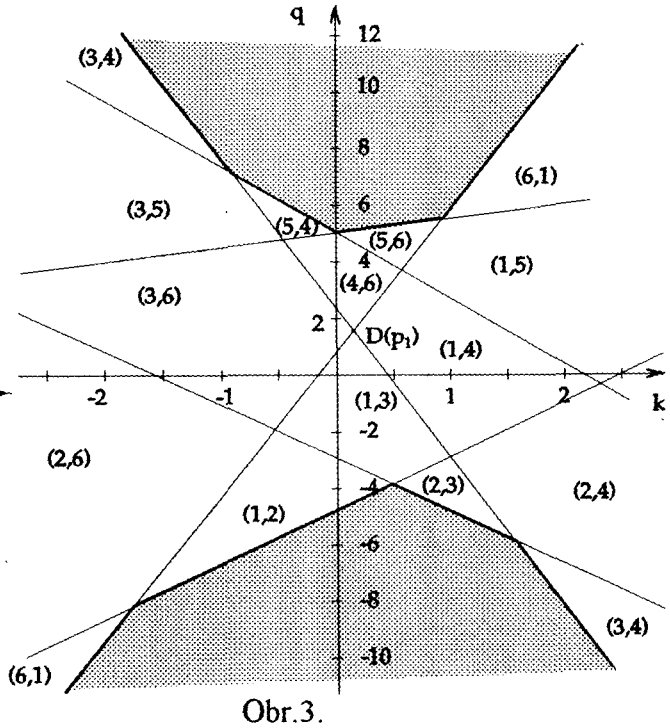
Základní představa činnosti algoritmu spočívá ve třech krocích, a to:

- převední konvexního n-úhelníka do duálního prostoru a vytvoření vhodné datové struktury,
- převedení ořezávané přímky do duální reprezentace, tj. určení (k,q) - obr.3,
- zjištění "oblasti", která je společná pro všechny přímky protínající dané dvě hrany, v níž se daný bod (k,q) nachází.

Tento postup, který je v principu velmi jednoduchý, lze demonstrovat na obr.2 a obr.3.



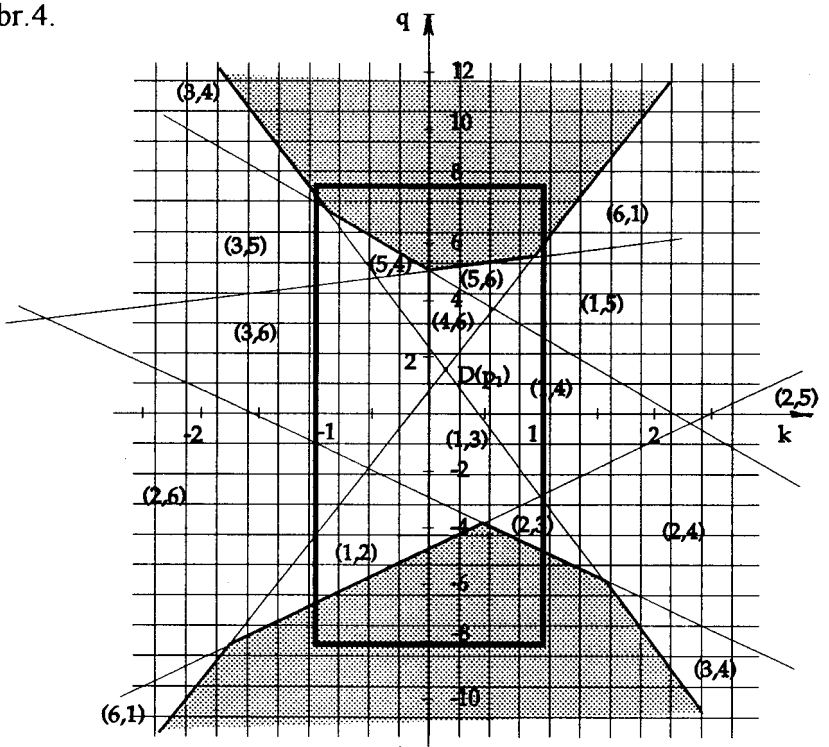
Obr.2.



Obr.3.

Z obrázku je zřejmé, že daný přístup naráží na dvě podstatné komplikace. První spočívá v principiální neomezenosti jednotlivých oblastí v duálním prostoru, druhá pak v tom, jak na základě znalosti polohy bodu duálního obrazu přímky, tj. znalost(k,q), určit oblast, v níž tento bod leží.

Použijeme-li obdobný přístup, tj. dělení prostoru, jako v případě bod uvnitř n-úhelníka, pak je zřejmé, že ze znalosti hodnot (k,q) lze přímo určit "obdélník" v němž daný bod leží. S tímto obdélníkem je pak spojen seznam všech "oblastí", se kterými daný obdélník inciduje, viz obr.4. Lze tedy říci, že k určení, které hrany daná přímka protíná, potřebujeme konstatní počet kroků, a tedy algoritmus bude mít očekávanou složitost O(1), viz obr.4.



Obr.4.

Posledním problémem je tedy eliminace neomezenosti jednotlivých oblastí v duálním prostoru. Lze říci, že každý n-úhelník lze uzavřít do kosočtverce "posazeného na vrchol". Je zřejmé, že každou přímku lze vyjádřit jako

$$y = kx + q \quad |k| \leq 1$$

nebo

$$x = my + p \quad |m| < 1$$

Lze tedy duální reprezentaci s neomezenými oblastmi nahradit pomocí dvou duálních reprezentací, kde již budeme uvažovat oblast <-1,1> x <-k,k> a <-1,1> x <-q,q> . Podle zpracovávané přímky se pak rozhodneme, kterou reprezentaci pro testování vybereme.

Z uvedeného tedy vyplývá, že ze znalosti koeficientů a,b,c rovnice zpracovávané přímky lze určit hrany protínané přímkou s **očekávanou složitostí O(1)**.

4. Dosažené výsledky

Pro ilustraci chování uveďme výsledky testů jednotlivých algoritmů. Alg_1 je nejrychlejší O(N) algoritmus, Alg_2 je nejrychlejší O(lg N) algoritmus a Alg_3 je navrhovaný algoritmus s očekávanou složitostí O(1). Přesné srovnání viz [3].

N	3	4	5	10	20	50	100
Alg_1	1.04	1.32	1.65	3.18	6.15	15.22	30.32
Alg_2	0.99	0.99	1.10	1.05	1.26	1.42	1.53
Alg_3	1.15	1.10	1.32	1.31	1.27	1.10	1.10

Bod uvnitř pravidelného n-úhelníka
Tabulka 1

N	3	4	5	10	20	50	100
Alg_1	0.66	0.88	0.88	1.37	2.31	4.01	7.58
Alg_2	0.33	0.28	0.27	0.28	0.38	0.38	0.43
Alg_3	0.11	0.16	0.17	0.11	0.11	0.11	0.11

Bod vně pravidelného n-úhelníka
Tabulka 2

N	3	4	5	10	20	50	100
Alg_1	0.77	1.33	1.64	3.19	6.15	9.17	12.24
Alg_2	0.89	0.99	0.94	1.20	1.27	1.32	1.43
Alg_3	1.16	1.16	1.10	1.16	1.15	1.10	1.10

Bod uvnitř nepravidelného n-úhelníka
Tabulka 3

N	3	4	5	10	20	50	100
Alg_1	0.61	0.82	0.93	1.87	3.57	4.45	5.93
Alg_2	0.38	0.44	0.49	0.61	0.66	0.66	0.66
Alg_3	0.11	0.11	0.11	0.11	0.10	0.11	0.11

Bod vně nepravidelného n-úhelníka
Tabulka 4

Zvýšením počtu pásů q-krát se zvýší dále efektivita testu Bod uvnitř n-úhelníka dle tabulky 5.

q	1	2	4	10
zrychlení	1.0	1.3	1.4	1.45

Tabulka 5

Algoritmus ořezávání přímky konvexním n-úhelníkem vykazuje předpokládané chování, tj. potvrzuje se očekávaná složitost O(1), a konkrétní výsledky porovnání s algoritmy se složitostí O(N) a O(lg N) jsou předmětem samostatného grantu.

5. Alternativa k paralelnímu zpracování?

Z uvedeného je zřejmé, že existují úlohy u nichž vhodné předzpracování vede k urychlení řešení daného problému. Ne vždy však předzpracování musí probíhat v "klasickém" pojetí a použití duálních reprezentací může být výhodné. Teoretické základy duálních reprezentací lze nalézt např. v [5]. Nicméně změna složitosti algoritmu pod optimální složitost se promítá v našem případě do zvýšených paměťových požadavků, a to s náročností O(M) u prvního algoritmu, u druhého pak $O(M^2)$, přičemž hodnota M je závislá na geometrických vlastnostech n-úhelníka [3]. Za podstatný rys uvedeného přístupu lze považovat zejména to, že zřejmě existuje poměrně široká třída problémů řešitelných uvedeným způsobem a že je možné nahradit velmi drahé paralelní zpracování převodem na algoritmy s větší paměťovou náročností, jejichž realizace je nepoměrně levnější.

6. Poděkování

Je mi milou povinností poděkovat všem studentům a kolegům za jejich připomínky, zejména pak Ing.P.Lederbuchovi a studentům zaměření Počítačová grafika a CAD systémy za nezávislé ověření algoritmů.

7. Literatura

[1] Kolingerová,I.: Řešení úloh počítačové grafiky pomocí duální transformace, Disertační práce ZČU-KIV, 1994.

[2] Skala,V.: An Efficient Algorithm for Line Clipping by Convex Polygon, Computers & Graphics (Pergamon Press), Vol.17, No.4, pp.417-421, 1993.

[3] Skala,V.: Point-in-Polygon Algorithm with O(1) Complexity, Preprint No.68, Univ.of West Bohemia, 1994.

[4] Skala,V.: $O(lg_2N)$ Line Clipping Algorithm in E^2, Computers & Graphics (Pergamon Press), Vol.18, No.4, pp.517-524, 1994.

[5] Stolfi,J.:Primitives for Computational Geometry, Technical Report DEC, 1989.

Abstract

New algorithms Point-in-Polygon and Line Clipping algorithms are very often used especially in computer graphics applications. Algorithms usually have O(N) or O(lg N) complexities. New algorithms with O(1) expected complexity has been developed using pre-processing. The interesting feature of presented algorithms is that memory requirements depend on geometrical properties of polygons. The presented approach can be considered as an alternative method to parallel processing.