Visibilidade em Polígonos utilizando algoritmos paralelos

Pedro Mitsuo Shiroma

20 de agosto de 2003

Resumo

A visibilidade de polígonos a partir de um ponto é um dos problemas clássicos na geometria computacional, onde deseja-se determinar, dado um conjunto de obstáculos, qual a região que é visível a partir de um ponto qualquer. Este trabalho enfoca a em três resultados em algoritmos paralelos para resolver este problema, utilizando o modelo CREW PRAM, uma arquitetura *mesh-connected* com múltiplo *broadcast* e um vetor sistólico. Em seguida são mostrados alguns problemas derivados a partir do problema de visibilidade, que tem atraído a atenção da comunidade científica recentemente sendo que alguns deles pertencem a classe $\mathcal{NP}$-difícil.

1 Introdução

Um dos problemas centrais em geometria computacional consiste na determinação da área visível de um plano a partir de um ponto no seu interior. Este problema aparece em inúmeras áreas, tais como computação gráfica, projeto de VLSI, navegação robótica , redes de sensores sem fio. Em computação gráfica, por exemplo, a visibilidade é utilizada para realizar o *ray tracing* do modelo tridimensional de um objeto, gerando uma vista (bidimensional) da cena; em robótica, além dos grafos de visibilidade que permitem o planejamento de trajetória, outro problema interessante consiste em manter um grupo de robôs que devem cooperar entre si mantendo a visibilidade [12]. No projeto de VLSI, o problema de otimização de espaço dos componentes no circuito integrado pode ser reduzido a um problema de visibilidade. Nas redes de sensores sem fio, a visibilidade é utilizada para minimizar o número de agentes (estações rádio-base no caso de telefonia celular) necessários para cobrir uma determinada área. Todos esses problemas envolvem o cálculo da visibilidade de objetos no plano a partir de um determinado ponto. Na seção 2 o problema de visibilidade em polígonos é formalmente definido, na seção 3 um algoritmo sequencial ótimo é analisado, na seção 4 o estado da arte em algoritmos paralelos é revisto e, na seção 6 os problemas em aberto e trabalhos futuros é discutido.

2 Definição do Problema

Definição 1 *Dado um conjunto de objetos geométricos e um ponto p, o polígono de visibilidade de p, $Vis(p)$ é o conjunto de pontos q tais que o segmento de reta pq não é interceptado por nenhum objeto.*

Tal definição, embora seja intuitiva e simples, não é adequada para o desenvolvimento de algoritmos, pois infinitos pontos deveriam ser testados para determinar $Vis(p)$. Problemas de computação geométrica freqüêntemente apresentam característica de serem facilmente definidos porém apresentarem soluções complexas, causado pela desconsideração do conhecimento geométrico a respeito do problema. Restringindo o problema para objetos que possam ser descritos por segmentos de retas, a visibilidade pode ser redifinida como:

Definição 2 *Dado um conjunto ordenado de n segmentos de reta que não se interceptam, $S = \{s_1, s_2, ..., s_n\}$ exceto, possivelmente nas extremidades e um ponto p localizado no plano, determine a parcela de cada segmento s_i visível para um observador posicionado em p.*

Um polígono é dito simples se ele não possui buracos no seu interior.

3 Algoritmo sequencial

Existem diversas técnicas consolidadas atualmente para resolver problemas geométricos e, dentre as principais técnicas podemos citar a varredura no plano, onde uma linha imaginária percorre todo o plano construindo, a medida que desloca, a solução do problema para a região já visitada, cuja implementação é bastante simplificada notando-se que os pontos onde a solução até então encontrada geralmente não é alterada continuamente à medida que a linha varre o plano, mas apenas em pontos discretos denominados pontos de evento. Nos algoritmos incrementais os pontos ou arestas são acrescentadas uma a uma, ou em pequenos conjuntos, e uma nova solução é determinada a cada iteração. Uma forma de evitar entradas maliciosas é utilizar algoritmos aleatórios.

4 Algoritmo paralelo

Em [11] o problema é resolvido através da estratégia dividir-para-conquistar em umar arquitetura CREW PRAM com complexidade de tempo $O(\log n)$ com n processadores. Uma simplificação é realizada considerando o ponto p na posição infinito negativo da coordenada y, reduzindo o problema de visibilidade à determinação do envelope inferior, composto de parcelas de cada um dos n segmentos, e propõe que o caso genérico de um ponto em posição finita pode ser resolvido pelo mesmo algoritmo. Considere, por exemplo, um conjunto de quatro segmentos $S = \{s_1, s_2, s_3, s_4\}$ cada uma definida pelas coordenadas x_{il}, x_{ir}, representando as coordenadas x da extremidade mais a esquerda e mais a direita de s_i, respectivamente (Fig. 1). Para calcular o envelope inferior, atribui-se a cada processador uma aresta sendo este responsável por calcular a visibilidade isoladamente do segmento, em seguida determina-se a visibilidade dos pares de segmentos, e assim suscetivamente até que o envelope para todos os segmentos tenham sido determinados. Seja T a árvore binária balanceada do gerada pelo *merge sort* tradicional contendo n nós folhas, associa-se a cada nó v_i uma lista $L[v_i]$ dos vértices visíveis considerando apenas s_i, se o nó for folha, e uma lista ordenada de vértices visíveis dos segmentos pertencentes a subárvore associada, se for um nó interno. O cálculo da interseção pode ser realizado em $O(\log n)$ utilizando a estratégia denominada *pipelined merge sort* onde, ao contrário do *merge-sort* tradicional, onde a lista $L[v]$ é determinada em cada estágio s, calcula-se uma aproximação para $L[v]$ tal que $L_{s+1}[v]$ seja um melhoramento da aproximação em $L_s[v]$. Algumas definições são necessárias para melhor explicar o funcionamento do algoritmo. Dado uma lista ordenada $L = \{s_1, s2, ..., s_n\}$ define-se a $c - sample$ de L como sendo a sublista ordenada $Sample_c(L) = \{s_c, s_{2c}, ...s_{in}\}$. A altitude de um nó, $alt(v)$ é definido como sendo a distância de v até um nó folha. Um nó v é dito completo se $L_s[v] = L[v]$. O algoritmo procede tornando, em cada estágio, as listas $L_s[v]$ completas tais que $s \geq 3 \cdot alt(v)$ e, mantendo esta invariante após $3 \cdot h(T)$ estágios o nó raiz contêm a lista ordenada, onde $h(T)$ representa a altura de T. Em cada estágio, apenas os nós onde $alt(v) \leq s \leq 3 \cdot alt(v)$ são ativos, e tornando $Sample(L_s[v]) = Sample_4$ se $s \leq 3 \cdot alt(v)$, $Sample_2$, se $s = 3 \cdot alt(v) + 1$ e $Sample_1$ se $3 \cdot alt(v) + 2$, o algoritmo 1 é capaz de ordenar a lista em tempo $O(\log n)$. Desta forma é possível determinar com custo total $O(n \log n)$ a visibilidade dos segmentos.

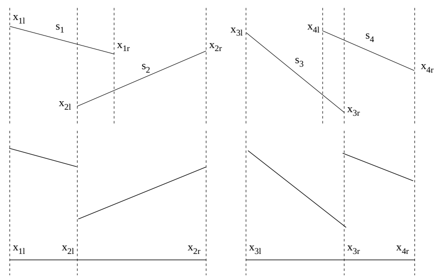


Figura 1: Visibilidade para $p = (0, -\infty)$

Algoritmo 1 Pipelined merge sort

entrada Para cada nó v da árvore binária, uma lista ordenada $L_s[v]$ tal que v é completo se $s \geq 3alt(v)$

saída Para cada nó v, uma lista ordenada $L_{s+1}[v]$ tal que v é completo se $s \geq 3alt(v) - 1$

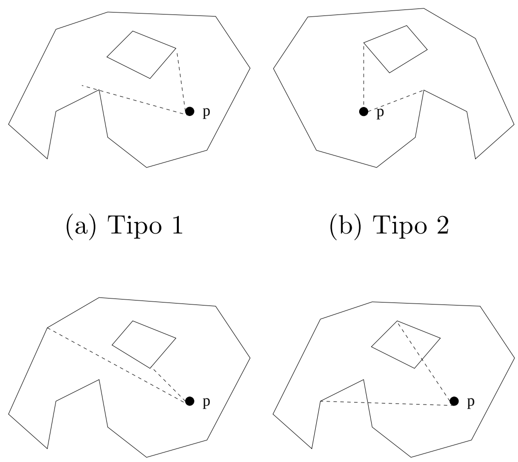
para todo os nós v tal que $alt(v) \leq s \leq 3 \cdot alt(v)$ **faça**

  seja u e w os filhos de v. Seja $L^l_{s+1}[u] = Sample(L_s[u])$ e $L^r_{s+1}[w] = Sample(L_s[w])$.

  $L_{s+1}[v] = \text{Merge}(L^l_{s+1}[u], L^r_{s+1}[w])$

fim para

Entretanto, não foram considerados os casos degenerados, quando dois segmentos possuem a mesma coordenada x ou retas verticais, entretanto, esses casos podem ser resolvidos utilizando a técnica conhecida como perturbação simbólica, que consiste em ordenar lexicograficamente na coordenada y.



(a) Tipo 1 (b) Tipo 2

(c) Tipo 3 (d) Tipo 4

Figura 2: Classificação dos vértices

Asano [3] utiliza uma arquitetura baseada em vetores sistólicos para resolver o problema de visibilidade com buracos com complexidade $O(n)$. Nesta arquitetura os processadores são dispostos linearmente formando um vetor onde cada processador possui dois vizinhos, exceto as extremidades. Os dados são alimentados a partir de um dos processadores das extremidades. O algoritmo possui três fases de complexidade de tempo linear, característico deste tipo de arquitetura, que consiste na distribuição de cada ponto para cada um dos processadores, no cálculo do resultado, e no retorno da resposta até o processador mestre.

Considere as arestas do contorno exterior orientadas no sentido anti-horário e as arestas dos componentes interiores no sentido horário, dado três vértices u, v, w respectivamente orientadas, v é classificado como sendo do tipo 1 se p estiver a esquerda de vu e a direita de uw, analogamente, ela é dita ser do 2 se vu estiver a direita e uw a esquerda, do tipo 3 se ambas as arestas estiverem a direita e do tipo 4 se ambas estiverem a esquerda.

Das definições anteriores, segue naturalmente que vértices do tipo 4 não são visíveis e, se vértices do tipo 1 e 2 são visíveis, então o polígono de visibilidade contêm um segmento entre v e a primeira interseção entre v e um raio emanando na direção pv (Fig. 2).

Algoritmo 2 Systolic Array

Spawn($v_1, ...v_n$), transmitindo para cada processador o um vértice e seu tipo correspondente.
para todo os nós v **faça**
  se o tipo de vértice v for 1 ou 2 **então**
    verifique se a aresta intercepta o segmento vp e o raio $r(v)$, partindo de v na direção vp, e se a interseção é mais próxima de v do que a interseção encontrada anteriormente.
  senão se o vértice for do tipo 3 **então**
    verifique se a aresta intercepta o segmento de reta vp.
  senão se o vértice for do tipo 4 **então**
    passe o resultado adiante
  fim se
  se a aresta transmitida intercepa o segmento vp **então**
    altere o tipo do vértice para 4.
  fim se
fim para
Retorne todos os vértices do tipo 1, 2 ou 3 e as interseções do raio $r(v)$.

[5] reduz o problema de *OR* de n bits para mostrar que o limite inferior de tempo para uma é de $O(\log n)$ e em seguida mostra um algoritmo ótimo no tempo para uma arquitetura *mesh-connected* contendo n^2 processadores, [1, 7] apresentam uma comparação da performance de vários algoritmos implementados em diversas arquiteturas diferentes. [4] propôs um algoritmo de custo ótimo em uma máquina EREW PRAM em tempo $O(\log n)$ utilizando $O(n/\log n)$ processadores e, em seguida [9] resolveu o problema para uma arquitetura *mesh-connected*.

5 Problemas Relacionados

Existem diversas variantes para o problema de visibilidade sendo, a mais famosa, o problema da galeria de arte, proposto por V. Klee, que consiste em determinar a quantidade mínima de guardas necessário para cobrir um museu, de forma que todas as paredes seja observáveis por algum guarda. [8] demonstrou que, para o caso de um polígono com n vértices e sem nenhum buraco, o número mínimo de guardas necessário é de $n/3$ e que ele é ótimo para certos polígonos, sendo que o melhor algoritmo existente possui complexidade $O(n)$. [6] e [10] demonstraram em 1991 que, para um polígono com h buracos, o mínimo de guardas necessários é igual a $(n+h)/3$. Mais tarde, em 1995, Bjorling propôs um algoritmo de complexidade $O(n^2)$ para determinar os pontos onde eles devem ser posicionados. Entretanto, determinar o número mínimo de guardas necessário para guardar um dado polígono é $\mathcal{NP}$-difícil, [13] demonstrou a complexidade do problema através de uma redução de 3-SAT.

Outro problema relacionado refere-se em determinar um caminho fechado de custo mínimo necessário para que todos os pontos do polígono sejam visíveis a pelo menos um ponto da trajetória, este problema também é conhecido como *minimum watchman tours* e pertence a classe polinomial caso não existam buracos no seu interior e a classe $\mathcal{NP}$-difícil caso existam, no caso onde o número de mudanças de trajetórias deve ser minimizado, [2] propôs um algoritmo não-determinístico com razão de aproximação $O(\log n)$. [14] demonstrou um algoritmo $O(n^5)$ para polígonos simples (sem buracos) onde o ponto de partida inicial não é pré-definido e representa o melhor tempo conhecido atualmente para esta classe de problemas.

6 Conclusão

O problema de visibilidade possui limite inferior $O(n)$ e, embora vários algoritmos tenham sido desenvolvidos nos últimos anos, apenas em 1991 um algoritmo ótimo foi proposto para resolver o problema.

Nota-se através da revisão bibliográfica que os trabalhos relacionados a visibilidade alcançaram o estado da arte em meados da década de 90 e pouco trabalho surgiu desde então, dando lugar para problema mais genéricos, como o *watchman tour* cujos algoritmos existentes, embora apresentam complexidade polinomial ainda possuem constantes elevadas como $O(n^5)$ e $O(n^9)$, dependendo da variante, e nenhum limite inferior ainda é conhecido para muitas classes desses problemas.

Referências

[1] S. G. Akl and K. A. Lyons. *Parallel Computational geometry*. Prentice Hall, 1993.

[2] E. M. Arkin, J. S. B. Mitchelly, and C. D. Piatko. Minimum-link watchman tours. Technical report, University at Stony Brock, 1994.

[3] T. Asano and H. Umeo. Systolic algorithms for computing the visibility polygon and triangulation of a polygonal region. *Parallel computing*, 6:209 – 216, 1988.

[4] M. J. Atallah, D. Z. Chen, and H. Wagener. An optimal parallem algorithm for the visibility of a simple polygon from a point. *Journal of the ACM*, 38(3):516 – 533, Julho 1991.

[5] D. Bhagavathi, V. V. Bokka, H. Gurla, S. Olariu, J. L. Schwing, I. Stojmenovic, and J. Zhang. Time-optimal visibility-related algorithms on meshes with multiple broadcasting. *IEEE Transactions on parallel and distributed systems*, 6(7):687 – 703, Julho 1995.

[6] I. Bjorling-Sachs and D. L. Souvaine. A tight bound for guarding polygons with holes. Technical Report LCSR-TR-165, Lab. Comput. Science/Rutgers University, 1991.

[7] D. Z. Chen. Efficient geometric algorithms on the erew pram. *IEEE Transactions on parallel and distributed systems*, 6(1):41 – 47, Janeiro 1995.

[8] Chvatal. A combinatorial theorem in plane geometry. *J. Combinatorial Theory*, 18(39):39–41, 1975.

[9] S. Guha. Optimal mesh algorithms for proximity and visibility problems in simple polygons. *Parallel algorithms and aplications*, 13:167 – 185, Janeiro 1998.

[10] F. Hoffmann, M. Kaufmann, and K. Kriegel. The art gallery theorem for polygons with holes. In *Proc. 32nd annual IEEE symposium found. computer science*, pages 39–48, 1991.

[11] J. JáJá. *An introduction to parallel algorithms*. Addison-Wesley, 1992.

[12] S. M. LaValle, H. H. González-Baños, C. Becker, and J.-C. Latombe. Motion strategies for maintaining visibility of a moving target. *Proc. of the IEEE International Conference on Robotics and Automation*, 1997.

[13] D. T. Lee and A. K. Lin. Computacional complexity of art gallery problems. *IEEE Trans. Info Th.*, 32:415 – 421, 1979.

[14] X. Tan. Fast computation of shortest watchman routes in simple polygons. *Inf. Proc. Lett.*, 77:27–33, 2001.