2. Intersecção de polígonos convexos

Estudaremos aqui o problema de construir a intersecção $P \cap Q$ de dois polígonos convexos dados P e Q. Para facilitar a exposição assumiremos que a intersecção entre os polígonos é 'não-degenerada': quando duas arestas se intersectam elas se intersectam em um único ponto. Dada está hipótese de não-degenerecência a intersecão de P e Q e formada por cadeia de P e Q alternadamente. (Veja Figura 1.)

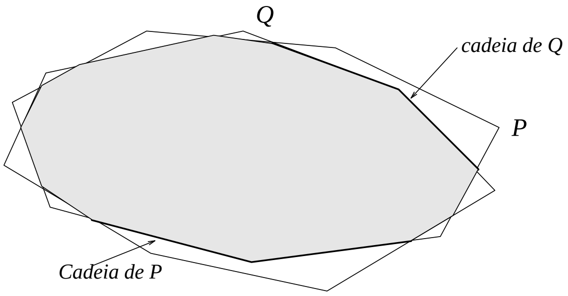


Figura 1. A intersecção entre P e Q.

A intersecção de dois polígonos arbitrários de n e m vértices pode ter complexidade quadrática (i.e. $\Omega(nm)$, veja Figura 2). Entretanto a interseção de dois polígonos convexos tem complexidade linear (i.e. $O(m+n)$). Intersecção de polígonos convexos é uma componente chave de muitos algoritmos (veja [4] Capítulo 7).

Existem vários algoritmos de complexidade de tempo linear para construir a intersecção entre dois polígonos convexos. O primeiro destes algoritmos lineares foi obtido por Shamos [5]. O algoritmo que veremos aqui é devido a O'Rourke, Chien, Olson & Naddor [3].

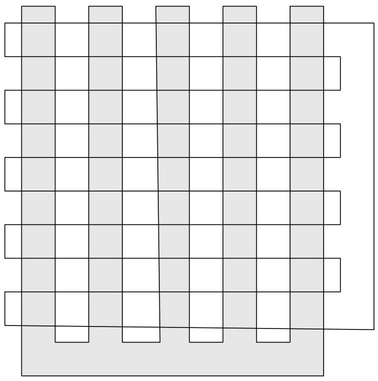


Figura 2. A intersecção de dois polígonos pode ter complexidade quadrática.

Observação. O material desta seção foi extraído do Capítulos 6 de [1], Capítulo 7 de [2], e Capítulo 7 de [4].

2.1. **O Algoritmo.** Sejam P e Q dois polígonos convexos. Assuma (como sempre temos feito) que as fronteiras de P e Q estão orientadas no sentido anti-horário. O algoritmo mantém um apontador p para uma aresta de P e um apontador q para uma aresta de Q (não faremos distinção alguma entre o apontador p ou q e aresta que eles apontam; assim, por exemplo, escreveremos a aresta p ao invés de a aresta apontada por p) . A idéia do algoritmo é avançar estes apontadores ao redor das fronteiras dos polígonos ao mesmo tempo que o polígono $P \cap Q$ vai sendo construído. Estas aresta 'caçam' uma a outra ao redor do polígono em busca dos pontos de intersecção. Como pontos de intersecção são encontrados na ordem que eles ocorrem em volta de $P \cap Q$, a construção do polígono que representa a intersecção estará completa quando um ponto de intersecção for obtido pela segunda vez. Se nenhuma intersecção é encontra então um teste simples pode decidir se os polígonos são disjuntos ou um está contido dentro do outro.

Chamaremos de *foice* cada uma das regiões limitadas por uma cadeia de P e uma cadeia de Q cujos extremos coincidem com um ponto da intersecção da fronteira de P com a fronteira de Q. A *cadeia interna* de uma foice é a cadeia da foice que pertence a fronteira de $P \cap Q$. (Veja Figura 3.)

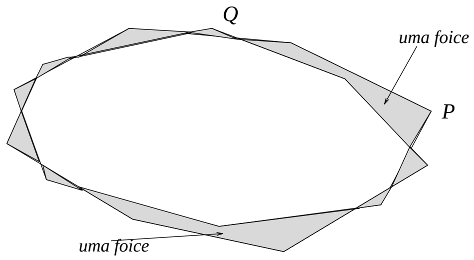


FIGURA 3. As foices que envolvem a intersecção dos polígonos.

Numa primeira fase o algoritmo avança p e q no sentido anti-horário até que ambos pertençam a uma mesma foice. Numa segunda fase p e q continuam a avançar no sentido anti-horário, mas ambas arestas se movem sincronizadas de uma foice para a próxima foice. Antes que as arestas passem para a próxima foice uma ponto de intersecção entre p e q é encontrado (um vértice de $P \cap Q$). O polígono $P \cap Q$ é construído durante a segunda fase. Se a aresta p pertence a cadeia interna da foice atual então ao avançarmos p a sua extremidade 'destino' é inserida na estrutura que representa a fronteira de $P \cap Q$. Analogamente, para a aresta q. Sempre que p e q se intersectam, o ponto de intersecção é calculado e inserido na estrutura que representa a fronteira de $P \cap Q$.

O algoritmo usa *regras de avanço* para decidir qual das arestas será avançada em cada iteração. As regras de avanço dependem da seguinte noção. Dizemos que uma aresta a *aponta para* uma aresta b se a semi-reta determinada por a (começando na extremidade destino de a) intersecta a reta determinada por b (veja Figura 4). Se a é uma aresta (orientada) de um polígono ligando o ponto a_1 ao ponto a_2 denotaremos por $\vec{a}$ o vetor $a_2 - a_1$. Uma aresta a aponta para b se uma das seguintes condições é satisfeita:

1. $\vec{a} \times \vec{b} \geq 0$ e o vértice destino de a está à esquerda de b, ou

2. $\vec{a} \times \vec{b} < 0$ e o vértice destino de a está à direita de b.

Note que $\vec{a} \times \vec{b} > 0$ corresponde ao caso em que o ângulo de $\vec{a}$ até $\vec{b}$ é menor do que no máximo π.

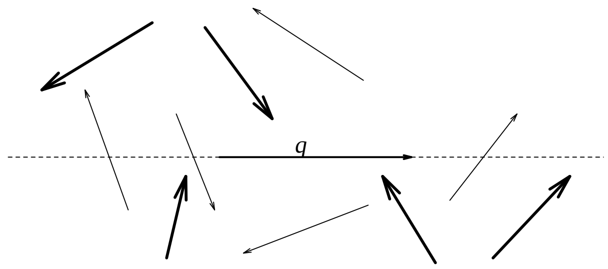


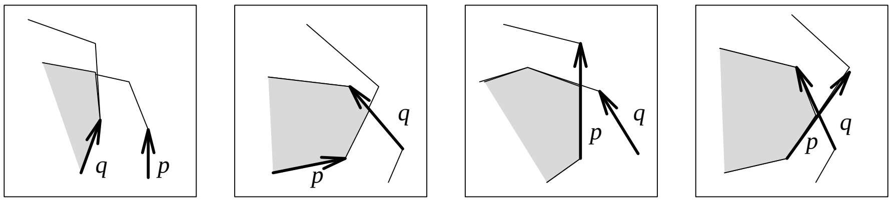
Figura 4. As aresta espessas correspondem a arestas que apontam para q.

As regras de avanço são feitas de tal maneira que o próximo ponto de intersecção de $P \cap Q$ que deve ser encontrado não seja pulado. As regras fazem distinção entre a aresta corrente que pode conter o próximo ponto de intersecção e a aresta corrente que não tem a menor chance de conter o próximo ponto de intersecão; esta última aresta pode ser avançada sem preocupação. Abaixo dizemos uma aresta a está *por fora* de uma aresta b se o ponto destino de a está à direita de b. As regras de avanço estão divididas em quatro:

1. p e q apontam um para o outro: avance a aresta (p ou q) que está por fora de outra. Na Figura 5(a), a aresta p será avançada. O próximo ponto de intersecção não pode estar sobre a aresta p já que p está fora da intersecção dos polígonos.
2. p aponta para q mas q não aponta para p: insira o ponto destino de p na estrutura que representa o polígono $P \cap Q$ se p não está por fora de q, e avance a aresta p. Na Figura 5(b) a aresta p não pode conter a próxima intersecção. A figura mostra a situação em que a aresta p que será avançada não esta por fora da aresta q.
3. q aponta para p mas p não aponta para q: insira o ponto destino de q na estrutura que representa o polígono $P \cap Q$ se q não está por fora de p, e avance a aresta q. Veja Figura 5(c). Este caso é simétrico ao anterior. A figura mostra a situação em que a aresta q está por fora da aresta p.
4. p não aponta para q e q não aponta para p: avance a aresta (p ou q) que está por fora da outra. Na Figura 5(d) a aresta p será avançada, pois p está por fora de q.

Abaixo encontra-se uma descrição em alto nível do algoritmo.

Algoritmo `Intersecção-de-Polígonos-Convexos`(P, Q).
`Entrada:` Dois polígonos convexos P e Q com n e m vértices, respectivamente.
`Saída:` O polígono $P \cap Q$.
1. Escolha uma aresta arbitrária p e q de P e Q, respectivamente.



(a) (b) (c) (d)

FIGURA 5. As quatro regras de avanço: (a) avança p; (b) avança p; (c) avança q; e (d) avança p. apontam para q.

2. **repeat**
3.   **if** p intersecta q **then**
4.   Verifique se a construção de $P \cap Q$ está completa.
5.   Atualize o indicador de *interior*.
6.   Avance p ou q dependendo da situação geométrica.
7. **until** o número de iterações é maior que $2(n+m)$.
8. Trate os casos $P \cap Q = \emptyset$, $P \subset Q$ e $P \subset Q$.

Depois de no máximo $2(m+n)$ iterações do laço entre as Linhas 3–7 sem que uma intersecção seja encontrada o algoritmo pára e subsequentemente decide se $P \subset Q$, $Q \subset P$ ou $P \cap Q = \emptyset$ (Linha 8). Alternativamente, se um ponto de intersecção é encontrado então o algoritmo continua até que este ponto seja encontrado uma segunda vez (Linha 4).

A váriavel *interior* (Linha 5 do algoritmo) indica qual dos polígonos está correntemente no interior do outro—i.e. o polígono cuja aresta corrente está na cadeia interior da foice corrente.

2.2. **Corretude.** A corretude do algoritmo também mostra que o mesmo conjunto de regras para avançar funciona para ambas as fases do algoritmo. As regras de avanço colocam p e q na mesma foice para que então p e q começem a avançar sincronizadamente de uma foice para a próxima foice. Denotaremos por n e m o número de vértices de P e Q, respectivamente. A corretude do algoritmo é baseado nos dois teoremas a seguir:

Teorema 1. *Se as arestas correntes p e q pertencem a uma mesma foice então o próximo ponto de intersecção (no qual a foice termina) será o próximo ponto de intersecção encontrado pelo algoritmo.*

Teorema 2. *Se as fronteiras de P e Q se intersectam então as arestas p e q se intersectarão depois de não mais do que $2(n+m)$ avanços.*

O Teorema 2 garante que o algoritmo irá encontrar um ponto de intersecção, se um tal ponto existir. Como as arestas p e q pertencem a um mesma foice se elas se cruzam então o Teorema 1 garante que as demais intersecções serão encontradas.

Prova do Teorema 1. Suponha que (digamos) p e q pertence a uma mesma foice e q alcança o próximo ponto de intersecção antes de p. Mostraremos que q ficará estacionado enquanto p fará uma série de avanços até alcançar q. Temos dois casos a considerar. Suponha primeiro que p está por fora de q

(veja Figura $6(a)$). Neste caso q ficará fixo enquanto p avançará fazendo zero ou mais aplicações da Regra 4, depois através de zero ou mais aplicações da Regra 1, e depois através de zero ou mais aplicações da Regra 2. No segundo caso, quando p não está por fora de q (veja Figura $6(b)$), q ficará fixo enquato p avança através de zero ou mais aplicações da Regra 2. ∎

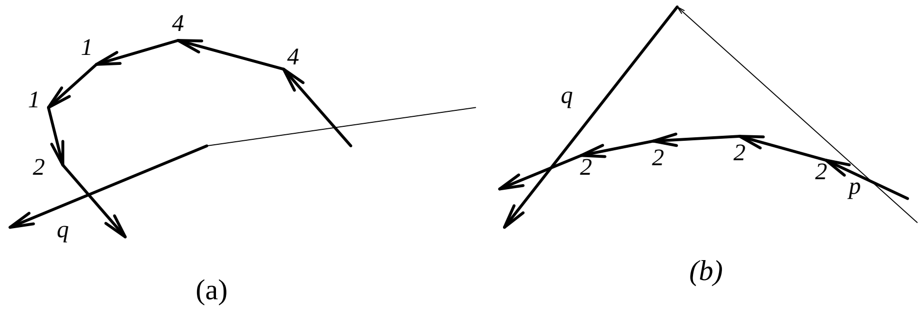


FIGURA 6. As quatro regras de avanço: (a) avança p; (b) avança p; (c) avança q; e (d) avança p. apontam para q.

Prova do Teorema 2. Assuma que as fronteiras de P e Q se intersectam. Depois de $m+n$ iteraçoes, ou p ou q deve ter percorrido toda a fronteira ao redor de P ou Q, respectivamente. Suponha que p tenham percorrido toda a fronteira de P. Em algum momento p deve ter se posicionado em uma aresta que contém um ponto de intersecção no qual o polígono Q passa do exterior para o interior de P. (Isto ocorre pois existem pelo menos dois pontos de intersecção e a direção das intersecções se alternam.) Seja q a aresta de Q quando p está na posição desta intersecção.

Na Figura 7, a fronteira de Q esta particionada em duas cadeias C_r e C_s. A primeira cadeia, C_r termina na aresta q_r, a aresta de Q que entra em P através da aresta p. A outra cadeia, C_s, termina na aresta q_s, cujo vertices estão à direita de p e cujo destino está o mais distante possível da reta determinada por p. Existem dois casos a considerar, dependendo de qual a cadeia que a aresta q pertence.

Caso 1: ($q \in C_r$) Neste caso, p fica fixo enquanto q avança através zero ou mais aplicações da Regra 3, então Regra 4, então Regra 1 e finalmente Regra 3, na qual o ponto de intersecção é encontrado.

Caso 2: ($q \in C_s$) Neste caso, q permanece fixo enquanto p avança através de zero ou mais aplicações da Regra 2, então Regra 4, então Regra 1, e finalmente Regra 2, neste momento p estará dentro de Q. Neste instante, p e q podem avançar um certo número de vezes—entretanto, q não pode avançar além do próximo ponto de intersecção antes que p atinga o ponto de intersecção anterior a q (se p já não tiver feito isto). Como p e q terminam na mesma foice, Teorema 1 garante que depois de alguns avanços adicionais, eles se cruzarão no ponto de intersecção na qual esta foice termina.

Para ver porque $2(m+n)$ iterações são suficientes para encontrar algum ponto de intersecção, observe que a posição inicial de p e q usada nos casos acima (a fronteira de Q entrando em P através de p e q situada em qualuer lugar) é atingida depois de no máximo $(m+n)$ iterações. Como depois disso nem p nem q avançam completamente ao redor dos seus respectivos polígonos até encontrar a primeira intersecção, então não mais do que $(m+n)$ avanços adicionais serão necessários. ∎

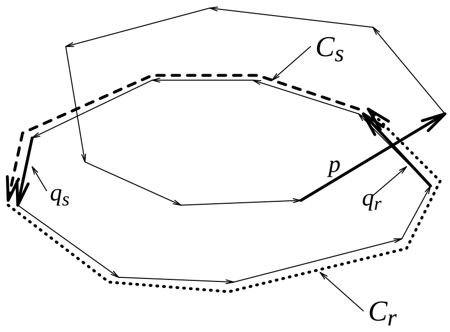


FIGURA 7. Figura ilustrando a prova do Teorema 2.

2.3. **Robustez.** O algoritmo para encontrar a intersecções de dois polígonos é suscetível a erros de arrendondamento.