Una propuesta para mejorar el cálculo de Sumas de Minkowski entre polígonos.

María Teresa Taranilla, Marcela Printista, Edilma Olinda Gagliardi

Departamento de Informática
Facultad de Ciencias Físico, Matemáticas y Naturales
Universidad Nacional de San Luis, Argentina
{ tarani,mprinti,oli }@unsl.edu.ar
Fax: 54-2652-430224

Resumen:

La Geometría Computacional es una disciplina que brinda un marco teórico y formal para dar soluciones a problemas de tipo geométrico. En este sentido, las operaciones entre polígonos modelan y brindan soluciones a una gama de problemas del mundo real. Una de estas operaciones es la denominada Suma de Minkowski. Esta operación es utilizada en un amplio rango de aplicaciones, tales como planificación de movimientos de robots, procesamiento de imágenes, sistemas de información geográfica, marcado y corte de moldes, entre otras.

En este trabajo se presentan las Sumas de Minkowski, los algoritmos y la complejidad de su cálculo entre polígonos y una propuesta de realizar estas operaciones haciendo énfasis en la performance de los algoritmos.

Palabras claves: Sumas de Minkowski. Geometría Computacional. Paralelismo de Datos. Pipeline.

1 Introducción

La Geometría Computacional es una disciplina que brinda un marco teórico y formal para dar soluciones a problemas de tipo geométrico[Tou85] [Tou92]. En este sentido, las operaciones entre polígonos modelan y brindan soluciones a una gama de problemas del mundo real. Una de estas operaciones es la denominada Suma de Minkowski. La suma de Minkowski tiene propiedades interesantes que la hace una herramienta útil que puede ser aplicada en la resolución de diversos problemas. Se utiliza en un amplio rango de aplicaciones, tales como planificación de movimientos de robots [Lat91], procesamiento de imágenes [Ser82] [Ser88], sistemas de información geográfica [HCC98], marcado y corte de moldes[Li94], entre otras.

Dados dos conjuntos Q y R $\subset R^2$, la suma de Minkowski de Q y R, se define como:

$Q \oplus R = \{ q + r : q \in Q , r \in R \}$ donde q + r es un vector que representa la suma de los vectores q y r. Es decir que dados los puntos $q = (q_x, q_y)$ y $r = (r_x, r_y)$, tenemos que $q + r = (q_x + r_x, q_y + r_y)$.

Es posible aplicar la definición de sumas de Minkowski a los polígonos, ya que un polígono es un conjunto de puntos en el plano. En la figura 1 podemos observar la suma de Minkowski entre dos polígonos convexos.

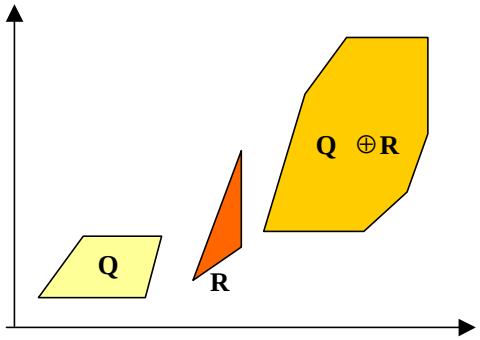


Figura 1

Desde el punto de vista estructural, el problema de calcular la suma de Minkowski de dos polígonos se caracteriza porque puede ser dividido en términos de una serie de computaciones independientes, cada una fuertemente asociada a una sub-estructura de datos. Tales computaciones son trivialmente paralelizables siguiendo un modelo de paralelismo de datos.

Desde el punto de vista geométrico, el problema exhibe algunas propiedades en el plano y espacio que serán explotadas para reducir el número de operaciones requeridas para sumar dos polígonos.

En el presente artículo, introducimos el concepto de sumas de Minkowski, presentamos los algoritmos que permiten realizar su cálculo entre polígonos convexos y no convexos. Para el caso de la suma de polígonos convexos presentamos algoritmos alternativos basándonos en algunas propiedades fundamentales del problema, como son sus características estructurales y geométricas. Finalmente, para el caso de la suma de polígonos no convexos, la propuesta original que resuelve el problema se basa en reducir el problema a la suma de polígonos convexos. Por lo que los algoritmos alternativos presentados en este trabajo, explotan explícitamente la propiedad estructural, ya que la geométrica será heredada por transitividad en la aplicación de algoritmos.

2 Construcción de sumas de Minkowski

2.1 Suma de Minkowski de polígonos convexos

Vamos a presentar algoritmos que permiten calcular la suma de Minkowski de dos polígonos convexos.

Un primer algoritmo que denominamos *fuerza bruta*, se basa en la siguiente idea: teniendo dos polígonos convexos *Q y R*, para todos los vértices de *Q y R* armamos pares de vértices, uno de *Q* y otro de *R*, y calculamos la suma de sus coordenadas, obteniendo *n* x *m* puntos candidatos para los vértices de $Q \oplus R$. Luego, calculamos el cierre convexo de esos puntos candidatos y obtenemos un nuevo polígono que es la suma de Minkowski de *Q* y R.

Algoritmo 1 (fuerza bruta):

Entrada: *Un polígono convexo Q con n vértices $v_1,...,v_n$ y un polígono convexo R con m vértices $w_1,...,w_m$. Se asume que la lista de vértices estará ordenada siguiendo el sentido contrario al de las agujas del reloj, siendo v_1 y w_1 los vértices con menor coordenada y.*

Salida: *La suma de Minkowski de $Q \oplus R$.*

1. $i \leftarrow 1;\ j \leftarrow 1;\ k \leftarrow 1;$
2. ***Para*** *i* ***desde*** *1* ***hasta*** *n* ***hacer***
3. ***Para*** *j* ***desde*** *1* ***hasta*** *m* ***hacer***
4. $s_k \leftarrow (v_i + w_j)$
5. $j \leftarrow (j + 1)$
6. $k \leftarrow (k + 1)$
7. ***Fin para***
8. $i \leftarrow (i + 1)$
9. ***Fin para***
10. $Q \oplus R \leftarrow Cierre\_Convexo\ (s_1,\ldots,s_k)$

Análisis: El enfoque de este algoritmo es muy sencillo, debiendo sólo computar una interacción entre cada par de vértices. La complejidad de este algoritmo es de orden cuadrático, lo cual, en tiempo de ejecución es ineficiente a medida que se incremente la cantidad de vértices de los polígonos.

Distintas alternativas han surgido para mejorar la performance del algoritmo. Estas alternativas se basan en analizar las propiedades estructurales del problema o sus propiedades geométricas.

2.1.1 Paralelismo de Datos en la Suma de Minkowski de polígonos convexos.

Los algoritmos que implementan el concepto de interacción total son bien conocidos. Los más directos se basan en un principio conocido como all-pairs [Han95]. Un problema all-pairs requiere una computación sobre cada posible sub-conjunto de dos elementos. Cada computación puede operar y/o transformar un par de elementos sin involucrar ningún otro elemento del conjunto. Por lo que, el problema queda caracterizado en términos de un conjunto de sub-computaciones independientes altamente paralelizables [WA99].

Una estructura algorítmica que ha sido ampliamente utilizada para resolver este tipo de problema, es el pipeline paralelo.

La técnica de pipeline se caracteriza por dividir el problema en sub-problemas que pueden ser ejecutados en sucesión. Los datos de entrada (inputs) son divididos en partes más pequeñas y procesadas una después de otras hasta completar la solución. En un pipeline cada sub-problema será procesado por una etapa del pipeline y de esta manera, un algoritmo puede ser eficientemente

ejecutado si cada una de estas etapas es alojada en un procesador diferente[Han95], [Qui94], [WA99].

La solución paralela considera la existencia de un pipeline de $NP = Max(n,m)$ procesadores (etapas).

Cada par de vértices ($v_i \in Q$ y $w_i \in R$) será distribuido en cada etapa del pipeline. En el caso que n sea distinto a m, la situación será indicada con la notación $(v_i, \_)$ o $(\_, w_i)$.

Sin pérdida de generalidad suponemos que los vértices del polígono comienzan numerados desde 0.

Algoritmo 2: (all- pairs pipeline)

Entrada: *Un polígono convexo Q con n vértices $v_1,...,v_n$, y un polígono convexo R con m vértices $w_1,...,w_m$*
La distribución de la secuencia de inputs tendrá la siguiente configuración inicial:
$0 \le i < Min(n,m)$, entonces P_i contiene el elemento (v_i, w_i)
$NP > i \ge n = Min(n,m)$, entonces P_i contiene $(\_, w_i)$
$NP > i \ge m = Min(n,m)$, entonces P_i contiene $(v_i, \_)$

Salida: *La distribución del output tendrá la siguiente configuración:*

$0 \le i < Min(n,m)$, entonces P_i contiene los elementos resultantes:
$v_i + w_{fase}$ (desde fase =0 hasta NP-1)

Hacer en paralelo para todo P_i *(desde i=0 hasta fase=NP-1)*

```
1.  min=Min(n, m)
2.  NP=Max(n,m)
3.  Para fase desde 0 hasta NP-1 hacer
4.      Si Pi ≤ min  Entonces  Computar (vi + wi)
5.      Si  (n  = min ) Entonces
6.                      Enviar_Siguiente(w(fase +NP-i) mod NP)
7.                      Recibir_Anterior(w(fase +1+NP-i) mod NP)
8.                  Sino
9.                      Enviar_Siguiente(v(fase +NP-i) mod NP)
10.                     Recibir_Anterior(v(fase +1+NP-i) mod NP)
11.     fase = fase +1
12. Fin Para
```

Análisis: Previo al trabajo del pipeline de procesadores, la entrada de datos debe ser particionada y distribuida entre los $NP = Max(n, m)$ procesadores en grupos formados por un vértice v_i y un vértice w_j. El esfuerzo computacional por fase, de cada estado (procesador) del pipeline, es el mismo y consiste de una operación de suma y dos pasajes de mensajes entre procesadores vecinos en el pipeline. El algoritmo pipeline resuelve la suma de polígonos convexos en NP fases. Posteriormente un procesador deberá recolectar desde los procesadores los vértices del nuevo polígono y ejecutar el cierre convexo.

2.1.2 Propiedades Geométricas de la Suma de Minkowski

Por otro lado, desde el punto de vista geométrico, es posible mejorar el costo del algoritmo de fuerza bruta basándonos en la siguiente observación acerca de los puntos extremos de la suma de Minkowski de dos polígonos.

Observación: *Para una dirección dada, un punto extremo en $Q \oplus R$ es la suma de los puntos extremos de Q y de R en la misma dirección.*

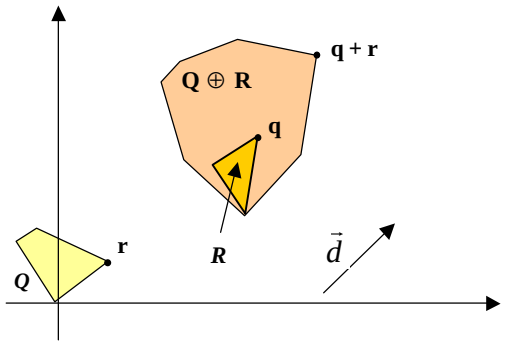


Figura 2

Haciendo uso de esta observación podemos encontrar un algoritmo que calcula la suma de Minkowski de dos polígonos convexos con complejidad lineal, mirando sólo los pares de vértices que son extremos en una misma dirección.

En un polígono convexo, si cambiamos la dirección $\vec{d}$ siguiendo el sentido contrario a las agujas del reloj, obtenemos una secuencia de puntos extremos que contienen los vértices del polígono ordenados exactamente como están ubicados si recorremos el borde del polígono en el sentido contrario a las agujas del reloj.

El algoritmo 3 recorre las direcciones siguiendo el sentido contrario a las agujas del reloj y usa la observación anterior para recorrer ambos polígonos y encontrar sus puntos extremos.

Algoritmo 3 (Suma de Minkowski de dos polígonos convexos)

Entrada: *Un polígono convexo Q con vértices $v_1,\ldots, v_n$, y un polígono convexo R con vértices $w_1,\ldots,w_m$. Se asume que la lista de vértices estará ordenada siguiendo el sentido contrario al de las agujas del reloj, siendo v_1 y w_1 los vértices con menor coordenada y. La notación ángulo(qr) para denotar el ángulo que forma el vector $\overrightarrow{qr}$ con el eje positivo de abscisas.*

Salida: *La suma de Minkowski de Q ⊕ R*

1. $i \leftarrow 1$; $j \leftarrow 1$
2. $v_{n+1} \leftarrow v_1$; $w_{m+1} \leftarrow w_1$
3. ***Repetir***
4. *Agregar* $v_i + w_j$ *como un vértice en Q ⊕ R*
5. ***Si ángulo***$(v_iv_{i+1}) <$ *ángulo*(w_jw_{j+1})
6. ***entonces*** $i \leftarrow (i + 1)$
7. ***sino Si*** *ángulo*$(v_iv_{i+1}) >$ *ángulo*(w_jw_{j+1})
8. ***entonces*** $j \leftarrow (j + 1)$
9. ***sino*** $i \leftarrow (i + 1)$
10. $j \leftarrow (j + 1)$
11. ***Si*** $i > n+1$ ***entonces*** $i \leftarrow n+1$
12. ***Si*** $j > m+1$ ***entonces*** $j \leftarrow m+1$
13. ***hasta que*** $i = n +1$ ***y*** $j = m +1$

Análisis: Este algoritmo se ejecuta en un tiempo lineal $O(m+n)$ ya que en cada ejecución de la sentencia de repetición ocurren hasta alcanzar los valores $n+1$ y $m+1$. Cualquier vértice de la suma de Minkowski $Q \oplus R$ es la suma de dos vértices originales que son extremos en una dirección común en Q y R. Ya que los polígonos son convexos, la verificación del ángulo asegura que esos pares extremos son encontrados.

2.2 Suma de Minkowski de un polígono convexo y uno no convexo

Consideremos la suma de Minkowski de un polígono no convexo *S* y un polígono convexo *R* con *n* y *m* vértices respectivamente.

En este caso, para calcular la suma de Minkowski, primero debemos reducir el caso a suma de polígonos convexos. Para ello vamos a descomponer el polígono no convexo en partes convexas. Una manera de hacerlo es triangulando el polígono. Después de triangularlo obtenemos *n*-2 triángulos. Luego, calculamos la suma de Minkowski de cada triángulo t_i obtenido con el polígono convexo *R*, usando el algoritmo 3 que calcula la suma de dos polígonos convexos con un orden lineal. La suma de Minkowski de *S* y *R* será igual a la unión de esas sumas, $S \oplus R = \bigcup_{i=1}^{n-2} t_i \oplus R$

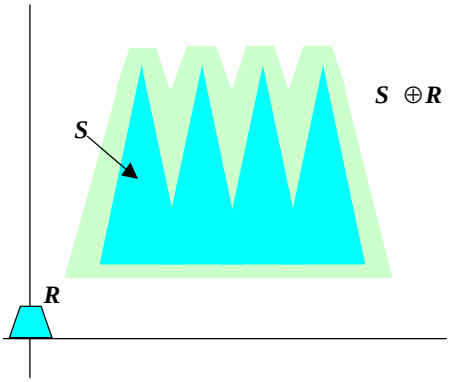
Algoritmo 4 (Suma de Minkowski de un polígono convexo con uno no convexo)

Entrada: *Un polígono no convexo S con n vértices $v_1,\ldots,v_n$, y un polígono convexo R con m vértices $w_1,\ldots,w_m$.*
Salida: *La suma de Minkowski de* $S \oplus R$

1. *Triangular el polígono no convexo S, obteniendo n-2 triángulos.*
2. ***Para*** *i* ***desde*** *1* ***hasta*** *n-2*
3. $S_i \leftarrow$ *calcular* $t_i \oplus R$
4. ***Fin para***
5. $S \oplus R \leftarrow$ *calcular* $\bigcup_{i=1}^{n-2} S_i$

Análisis: Para analizar la complejidad de este algoritmo, debemos tener en cuenta varios pasos. Primero, triangular el polígono *S* en el paso 1 puede hacerse en orden lineal [Cha91], luego en el paso 3 el cálculo de la suma de Minkowski de cada triángulo t_i con *R* lleva un tiempo de $O(m)$, y como este cálculo se repite *n* veces, la complejidad de este paso es de $O(n \times m)$. Por último debemos analizar cuál es el costo de calcular la $\bigcup_{i=1}^{n-2} S_i$. La unión de dos polígonos convexos de *r* y *s* lados toma un tiempo de $O(r+s)$. Como cada polígono S_i tiene $m + 3$ lados , el cálculo de $\bigcup_{i=1}^{n-2} S_i$ que es igual a $S \oplus Q$, se puede llevar a cabo en un tiempo de $O(n \times m)$.

Esto implica que la suma de Minkowski de un polígono no convexo *S* y un polígono convexo *R* con *n* y *m* vértices, $S \oplus R$, tiene una la complejidad de $O(n \times m)$.



Considerando un polígono S no convexo de tipo peineta con $\lfloor n/2 \rfloor$ puntas apuntando hacia arriba y un polígono convexo *R* mucho más pequeño que se corresponde con la mitad superior de un polígono regular $(2m-2)$-gono.
La suma de Minkowski de estos dos polígonos tiene también $\lfloor n/2 \rfloor$ puntas cada una de las cuales tiene *m* vértices en su tope.

Figura 3: Suma de Minkowski de un polígono convexo con uno no convexo

2.2.1 Paralelismo de datos en la suma de Minkowski de un polígono convexo y uno no convexo

En esta sección nos basamos en un esquema de particionado de datos. Con la disponibilidad de NP = n -2 procesadores, se puede implementar un algoritmo paralelo que particione el conjunto de datos a tratar y que haga responsable a cada procesador de una parte de la solución final.

Dado que la suma de Minkowski opera sobre el polígono convexo *R* y sobre cada uno de los triángulos resultantes de la triangulación, t_i, sin requerir de ningún otro triángulo (t_j , *con j distinto de i)*, se puede obtener una computación paralela trivial. Una vez efectuada la triangulación de polígonos, la entrada de datos debe ser particionada y distribuida entre los procesadores en grupos formados por un polígono t_i y el polígono *R*. Esto puede ser realizado sin ninguna técnica de programación especial, mas que *n-2* pasajes de mensajes desde el procesador que realizó la triangulación hacia los *NP* (o los restantes *NP-1*) procesadores. Cada procesador P_i recibirá la información que representa la estructura del polígono convexo, t_i. El tiempo de computación se reduce al tiempo de computar la suma de Minkowski de dos polígonos convexos.

2.3 Suma de Minkowski de dos polígonos no convexos

En el caso de tener de dos polígonos no convexos *S* y *Z*, para calcular la suma de Minkowski de ambos se deben descomponer ambos polígonos en partes convexas. Entonces se triangulan ambos polígonos, y a continuación se debe calcular la suma de Minkowski de cada par de triángulos usando el algoritmo 3, para luego calcular la unión de los polígonos obtenidos en cada una de las sumas.

Algoritmo 5 (Suma de Minkowski entre dos polígonos no convexos)

Entrada: *Un polígono no convexo S con n vértices* $v_1,\ldots, v_n$ *, y un polígono no convexo Z con m vértices* $w_1,\ldots,w_m$.
Salida: *La suma de Minkowski de* $S \oplus Z$

1. *Triangular el polígono no convexo S, obteniendo n-2 triángulos.*
2. *Triangular el polígono no convexo Z, obteniendo m-2 triángulos.*
3. ***Para*** *i* ***desde*** *1* ***hasta*** *n-2*
4. ***Para*** *j* ***desde*** *1* ***hasta*** *m-2*
5. $R_k \leftarrow$ *calcular* $t_i \oplus u_j$
6. ***Fin para***
7. ***Fin para***
8. $S \oplus Z \leftarrow$ *calcular* $\bigcup_{k=1}^{(n-2)(m-2)} R_k$

Análisis: cuando se triangulan ambos polígonos, se obtiene un conjunto de *n-2* triángulos t_i y otro conjunto de *m*-2 triángulos u_j, estos pasos tienen una complejidad de $O(n)$ y $O(m)$ respectivamente. La suma de Minkowski es ahora la de los pares t_i, u_j. Cada suma $t_i \oplus u_j$ tiene una complejidad constante, pero hay *n m* parejas de triángulos. Por lo tanto, $S \oplus Z$ es la unión de (*n*-2)(*m*-2) polígonos de complejidad constante. Esto implica que la complejidad total de $S \oplus Z$ será $O(n^2 \times m^2)$ en el peor de los casos.

2.3.1. Paralelismo de datos en la Suma de Minkowski de dos polígonos no convexos.

El enfoque de particionado de datos es similar al de la sección 2.2.1. Una vez efectuada la triangulación de los polígonos, la entrada de datos debe ser particionada y distribuida entre los

procesadores en grupos formados por un polígono t_i y un polígono u_j. Con la disponibilidad de $NP = n + m - 4$ procesadores, se puede implementar un algoritmo paralelo que particione el conjunto de datos a tratar y que haga responsable a cada procesador de una parte de la solución final.

Para que cada procesador pueda resolver su tarea es necesaria una distribución de datos iniciales. Esto implica $n + m - 4$ pasaje de mensajes punto a punto desde el procesador que realizó la triangulación hacia los *NP* (o los restantes *NP-1*) procesadores. Cada mensaje contiene la información de representación de dos polígonos convexos. El tiempo de computación se reduce al tiempo de computar la suma de Minkowski de dos polígonos convexos.

3 Conclusiones

La Geometría Computacional es una disciplina propia de la Matemáticas, la cual brinda un marco teórico y formal para el diseño y análisis de algoritmos requeridos para dar soluciones a problemas que surgen en áreas de la Computación. En nuestro trabajo nos proponemos observar y mejorar las técnicas algorítmicas y las estructuras de datos que surgen de la Geometría Computacional, además de la interacción con otras ramas de la Informática.

Hemos mostrado cómo construir la solución a un problema, cuyo dominio de representación puede ser de alguna manera particionado, basándonos en soluciones parciales elaboradas concurrente o simultáneamente. Con la disponibilidad de múltiples procesadores, en estos problemas, el aumento en la dificultad del problema a resolver no necesariamente implica un aumento en la complejidad del tiempo de ejecución del algoritmo que la resuelve.

Referencias Bibliográficas

[AHU74] Aho, A.V.; Hopcroft, J. E.; Ullman, J. *The desing and analysis of computer algorithms*, Addison-Wesley Series in Computer Science and Information Processing, 1974.

[BKOS97] de Berg, M; Kreveld, Overmars, M; Schwarzkopf. *Computational Geometry: algorithms and applications*, Springer Verlag, BH 1997

[Cha91] Chazelle, B., *Triangulating a simple polygon in linear time*, Discrete Computational. Geometry, 6: 485-524,1991

[Han95] Hansen B., *Studies in Computational Science: Parallel Programming Paradigms.* Prentice Hall, Englewood, New Jersey. 1995.

[HCC98] Heywood I., Cornelius S., Carver S., *Geographical Information Systems*, Addison-Wesley Longman, New York, 1998.

[Lat91] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publisher, Boston, 1991.

[Li94] Li, Zhenyu *Compation Algorithms for Non-Convex Polygons and their Applications*, tesis doctoral, Universidad de Harvard, 1994

[PS85] Preparata,F.; Shamos,M. *Computational Geometry: an Introduction*, Springer Verlag, NY 1985.

[Qui94] Quinn M., *Parallel Computing.Theory and Practice.* 2th.Edtition. McGraw-Hill, 1994.

[Ser82] Serra J., *Image Analysis and Mathematical Morphology*, Academic Press, New York, 1982.

[Ser88] Serra J., *Image Analysis and Mathematical Morphology, Vol II: Theorical Advances*, Academic Press, New York, 1988.

[TK02] Taranilla, M.T, Kavka, G.; *Implementación de una herramienta para el cálculo y visualización de sumas de Minkowski*. UNSL, 2002.

[Tou85] Toussaint, G.T. *Computational Geometry*, Edited by North-Holland, Amsterdam, 1985 .

[Tou92] Toussaint, G.T. *What is computational geometry?* Proc. IEEE, vol. 80, No. 9,. (1347-1363),1992.

[WA99] Wilkinson B, Allen M. *Parallel Programming: Techniques and Application using networked Workstation and Parallel Computers*. Prentice-Hall. 1999