الگوریتم هایی ابتکاری برای تولید چند ضلعی تصادفی با استفاده از مجموعه ای از رئوس داده شده

علی نوراله [۱]، لعیا محمدی [۲]

[۱] دانشکده مهندسی برق و کامپیوتر دانشگاه آزاد اسلامی قزوین و دانشکده مهندسی برق و کامپیوتر دانشگاه تربیت دبیر شهید رجایی، تهران، nourollah@aut.ac.ir

[۲] دانشکده مهندسی برق و کامپیوتر دانشگاه آزاد اسلامی قزوین، l.mohammadi@qiau.ac.ir

چکیده – یکی از مسائل مطرح شده در گرافیک کامپیوتری و هندسه محاسباتی تولید چندضلعی‌های تصادفی می‌باشد. در این مقاله، مسئله تولید چندضلعی‌های تصادفی ساده بر روی یک مجموعه از رئوس در نظر گرفته می‌شود. اهمیت این مسئله در ارزیابی عملی الگوریتم‌هایی است که روی چند ضلعی‌ها عمل می‌کنند و لازم است تا دقت الگوریتم بررسی شده و میزان مصرف واقعی پردازنده یک الگوریتم به طور تجربی تضمین شود. تا کنون هیچ راه حلی با زمان چند جمله‌ای برای تولید تصادفی یکنواخت چندضلعی‌ها شناخته نشده است. در این مقاله، سه الگوریتم ابتکاری جدید برای تولید چندضلعی‌های تصادفی با استفاده از مجموعه‌ای از رئوس تصادفی با پیچیدگی زمانی $O(n\ logn)$ ارائه می‌شود.

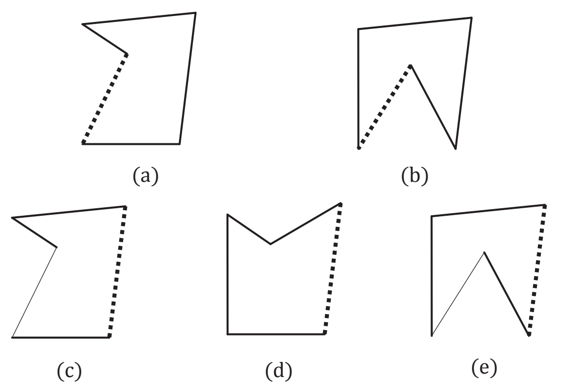
کلیدواژه- پیمایش زاویه‌ای، پوسته محدب، چندضلعی ساده تصادفی، چندضلعی ستاره‌ای شکل، گرافیک کامپیوتری.

۱ – مقدمه

گرافیک کامپیوتری علم محاسبه تصاویر است که منجر به ساخت آنها می‌شود. یکی از شاخه های مهم گرافیک کامپیوتری، هندسه محاسباتی است. هندسه محاسباتی، محاسباتی را روی اشکال هندسی مانند چندضلعی‌ها انجام می‌دهد. چندضلعی‌ها اشکال مناسبی برای نمایش اشیای دنیای واقعی هستند و هر شیئی در طبیعت به صورت مجموعه‌ای از چند ضلعی‌ها قابل نمایش است. به دلیل اهمیت این مسئله، یکی از مسائل مهم مطرح شده در گرافیک کامپیوتری و هندسه محاسباتی، مسئله تولید چندضلعی‌های تصادفی می‌باشد.

مسئله تولید چندضلعی‌های تصادفی به شکل زیر تعریف می‌شود: با داشتن مجموعه‌ای از رئوس S می‌توان یک چندضلعی ساده روی S، به طور تصادفی با یک توزیع یکنواخت تولید کرد (یعنی اگر k چندضلعی ساده روی S وجود داشته باشد تولید هر یک از این چندضلعی‌ها با احتمال $۱/k$ مد نظر است).

با در نظر گرفتن دو جفت از رئوس متفاوت و دو یال تعریف شده به وسیله این جفتها به آسانی مشاهده می‌شود که تعداد چند ضلعی‌های ساده شامل یکی از این یال‌ها می تواند مختلف باشد. این موضوع در شکل ۱ نشان داده شده که در آن یال مورد نظر با یک خط چین مشخص شده است: برای اولین یال دقیقا دو چندضلعی ساده وجود دارد که شامل آن است (شکل ۱a و۱b) درحالیکه برای یال دوم، سه چندضلعی‌ساده شامل آن وجود دارد(شکل۱c و ۱dو ۱e). متاسفانه هیچ الگوریتمی در زمان چند جمله‌ای شناخته نشده است که چندضلعی‌های ساده تصادفی با یک توزیع یکسان تولید کند. این مسئله محققان را بر آن می‌دارد تا الگوریتم‌های ابتکاری را برای تولید چندضلعی‌های ساده تصادفی به کار گیرند.



شکل ۱: دو یال متعلق به تعداد مختلفی از چند ضلعی های ساده.

تولید چندضلعی‌های ساده تصادفی دو زمینه کاربردی مهم دارد:
۱- اعتبارسنجی تجربی یک الگوریتم عمل‌کننده روی چندضلعیها
۲- تست عملی مصرف پردازنده آن.

الگوریتم‌های بسیار زیادی برای حل مسئله تولید چندضلعی های تصادفی ارائه شده است که برخی از آنها در این مقاله به اختصار بیان می‌شوند[۱].

ادامه این مقاله به این شکل سازماندهی شده است: بخش ۲ به کارهای انجام‌شده برای تولید چندضلعی‌های تصادفی می‌پردازد. در بخش ۳ سه الگوریتم پیشنهادی مطرح می‌شود. بخش۴ کارایی الگوریتم‌های مطرح شده را مورد بررسی قرار می‌دهد و نهایتا در بخش ۵ نتیجه‌گیری مطرح خواهد شد.

۲ -کارهای انجام شده

بنابر دلایل ذکر شده در بخش قبل، تولید تصادفی اشیای هندسی مورد توجه خاص محققان قرار گرفته است. Epstein و Sack یک الگوریتم $O(n^۴)$ برای تولید مثلث‌بندی چندضلعی‌های ساده به طور تصادفی ارائه کردند[۲].

Devroye، Epstein و Sack، تولید تصادفی فاصله‌ها و اَبرمستطیل‌ها را مطالعه کردند. آنها مسئله تولید یک اَبرمستطیل تصادفی را در یک اَبرمکعب واحد در نظر گرفتند[۳].

Atkinson وSack، تولید یکسان جنگل‌های با محدودیت ارتفاع را مطالعه کردند. یک درخت k-way یا تهی است یا شامل یک گره ریشه است و یک ترتیب منظم از k زیردرخت دارد. الگوریتم آنها در $O(n^۳.h)$ اجرا می‌شود که h عمق درخت را مشخص می‌کند[۴].

یک الگوریتم در حداکثر زمان $O(n^۲)$ برای تولید تصادفی چندضلعی‌های x-monotone توسط Zhu و دیگران توصیف شده بود[۵]. یک راهکار جالب برای تولید چندضلعی‌های تصادفی در سطح (ولی نه روی یک مجموعه رئوس) به وسیله O'Rourke و Virmani، ارائه شد[۶]. در بخش بعدی الگوریتم های ابتکاری پیشنهادی برای این مسئله مطرح می‌شود.

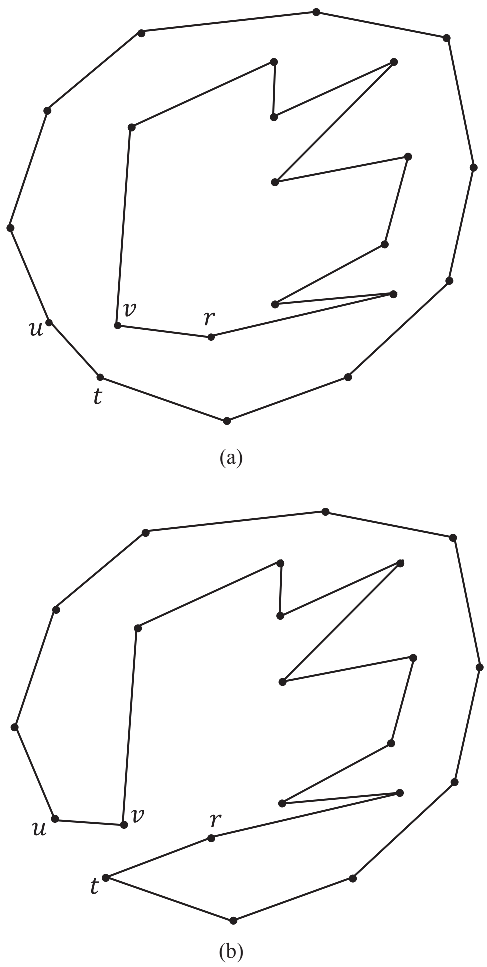
۳ -الگوریتم‌های پیشنهادی

در این بخش سه الگوریتم پیشنهادی برای تولید چندضلعی‌های تصادفی مطرح می‌شود. هر سه الگوریتم پیچیدگی زمانی $O(n\ logn)$ دارند.

۳ -۱ - الگوریتم ۱

فرض کنید S مجموعه رئوس تصادفی است و هیچ سه نقطه ای همراستا (هم‌خط) نیستند. پوسته محدب یک مجموعه رئوس S در صفحه ($CH(S)$)، یک چند ضلعی محدب با کوچکترین مساحتی است که شامل آن رئوس باشد. با این توضیح، این الگوریتم به این ترتیب عمل می‌کند که ابتدا $CH(S)$ به دست می‌آید. فرض کنید $\acute{S}$ مجموعه رئوس داخلی این پوسته محدب باشد. سپس برای مجموعه رئوس داخل پوسته محدب به این ترتیب عمل می‌شود که ابتدا راسی با کمترین مختصات y پیدا می‌شود. فرض می شود این راس v باشد. رئوس $\acute{S}$ بر اساس زاویه قطبی در جهت پادساعتگرد حول v مرتب می‌شوند و با پیمایش زاویه‌ای این رئوس، حول راس v، رئوس ملاقات شده به هم متصل می‌شوند. آخرین نقطه ملاقات شده به نقطه v متصل می‌شود و به این ترتیب یک چندضلعی ستاره‌ای (که از نقطه‌ای داخل آن تمام رئوس قابل دید هستند) حاصل می‌شود. حال از مجموعه رئوس روی $CH(S)$، نزدیکترین راس به v انتخاب شده و فرض می شود این راس u باشد. اگر راس مجاور به v روی چندضلعی ستاره‌ای شکل داخلی در جهت پادساعتگرد r، و راس مجاور به راس u روی $CH(S)$، در جهت پادساعتگرد، t باشد(شکل ۲a)، یالهای (v,r), (u,t) با (v,u) , (t,r) جایگزین می‌شوند. به این ترتیب یک چندضلعی تصادفی ایجاد می‌شود(شکل ۲b). روند این الگوریتم به صورت زیر می‌باشد:

- ابتدا $CH(S)$ روی مجموعه رئوس تصادفی S محاسبه می‌شود.
- مجموعه رئوس داخل $CH(S)$ را $\acute{S}$ در نظر گرفته و راس با کمترین مختصات y از این مجموعه رئوس انتخاب می‌شود(راس v).
- رئوس $\acute{S}$ بر اساس زاویه قطبی در جهت پادساعتگرد ملاقات می‌شوند و با پیمایش زاویه‌ای حول راس v و اتصال رئوس مرتب شده یک چندضلعی ستاره‌ای حاصل می‌شود.
- از مجموعه رئوس روی $CH(S)$ نزدیکترین راس به راس v انتخاب می‌شود(راس u).
- راس مجاورv در جهت پادساعتگرد r و راس مجاور u در جهت پادساعتگرد t نامگذاری می‌شوند.
- یالهای (v,r) , (u,t) با (v,u) , (t,r) جایگزین می‌شوند.



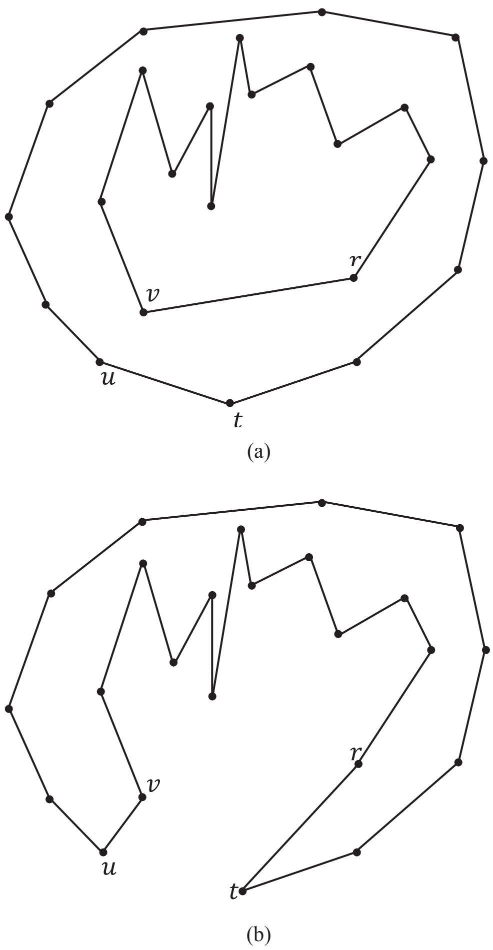
شکل ۲: (a) ساخت یک چند ضلعی ستاره ای با نقاط داخل پوسته محدب، (b) ساخت چند ضلعی تصادفی با جا به جایی یالهای (u,t) , (v,r) با (t,r) , (v,u)

۳ ۴ -الگوریتم ۲

در این الگوریتم، نیز مانند الگوریتم قبل ابتدا $CH(S)$ محاسبه می‌شود. فرض می‌شود مجموعه رئوس داخل پوسته محدب $\acute{S}$ باشد. برای این مجموعه رئوس الگوریتم موجود Convex Buttom به کار می‌رود که دارای مرتبه زمانی $O(n\ logn)$ می‌باشد و به این ترتیب عمل می‌کند که ابتدا دو راسی که به ترتیب کمترین و بیشترین مختصات x را دارند در نظر گرفته می‌شوند. این رئوس به وسیله یک خط فرضی به هم متصل می‌شوند به طوریکه مجموعه رئوس به دو نیمه بالایی و پایینی تقسیم می‌شوند (اگر مجموعه رئوس نیمه پایین تهی باشد خط فرضی به عنوان پوسته محدب دو نقطه به شمار می‌آید و باز هم الگوریتم عمل می‌نماید). حال پوسته محدب نیمه پایین خط فرضی محاسبه می‌شود و خط فرضی حذف می‌شود. همه رئوس باقی‌مانده از چپ به راست به ترتیب x مرتب شده و چپ‌ترین و راست‌ترین نقطه به ترتیب به رئوس انتهایی چپ و راست پوسته محدب متصل می‌شوند[۷].

در ادامه، همانند الگوریتم بالا رئوس v و u و r و t به همان ترتیب توضیح داده شده در الگوریتم قبل، در نظرگرفته می‌شوند(شکلa۳) و یالهای $(v,r), (u,t)$ با $(v,u), (t,r)$ جایگزین می‌شوند(شکل b۳) چندضلعی حاصل یک چندضلعی تصادفی ساده می‌باشد. روند این الگوریتم با فرض داشتن الگوریتم Convex Buttom به صورت زیر است:

- ابتدا $CH(S)$ روی مجموعه رئوس تصادفی S محاسبه می‌شود.
- مجموعه رئوس داخل $CH(S)$ را $\acute{S}$ در نظر گرفته و راس با کمترین مختصات y از این مجموعه رئوس انتخاب می‌شود(راس v).
- برای رئوس داخل پوسته محدب (مجموعه$\acute{S}$) الگوریتم Convex Buttom فراخوانی می‌شود.



شکل ۳: (a) ساخت یک چند ضلعی با الگوریتم Convex Buttom با نقاط داخل پوسته محدب، (b) ساخت چند ضلعی تصادفی با جا به جایی یالهای (v,r) , (u,t) با (v,u) , (t,r) .

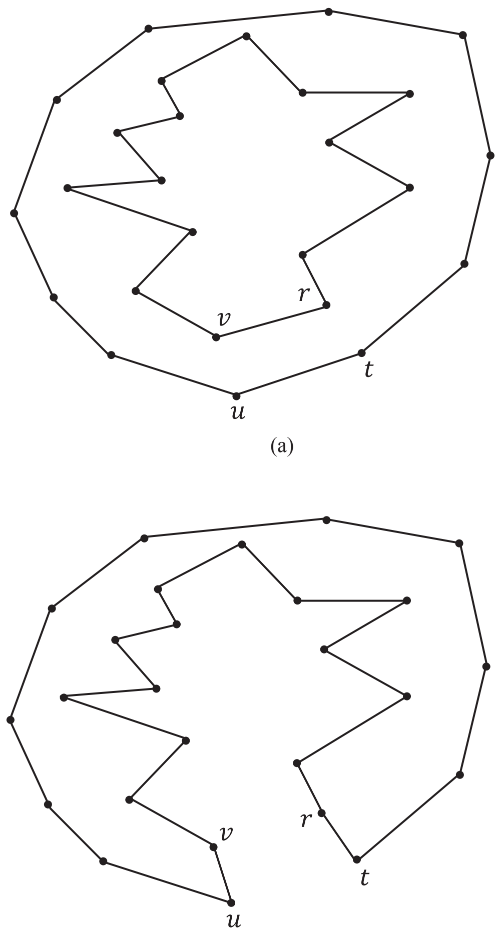
- از مجموعه رئوس روی $CH(S)$ نزدیکترین راس به راس v انتخاب می‌شود(راس u).
- راس مجاور v در جهت پادساعتگرد r و راس مجاور u در جهت پادساعتگرد t نامگذاری می‌شوند.
- یالهای (v,r) , (u,t) با (v,u) , (t,r) جایگزین می‌شوند.

۳-۳ -الگوریتم ۳

در این الگوریتم، همانند دو الگوریتم قبل ابتدا $CH(S)$ محاسبه می‌شود. فرض می‌شود مجموعه رئوس داخل پوسته محدب $\acute{S}$ باشد. برای این مجموعه رئوس الگوریتم موجود TwoPeasants (برای خط فرضی بر اساس مختصات y) به کار می رود که دارای مرتبه زمانی $O(n\ logn)$ می‌باشد و به این ترتیب عمل می‌کند که ابتدا دو نقطه ای که به ترتیب کمترین و بیشترین مقدار y را دارند در نظر گرفته می‌شوند. این رئوس به وسیله یک خط فرضی به هم متصل شده به طوریکه مجموعه رئوس به دو نیمه بالایی و پایینی تقسیم می‌شوند (اگر یک مجموعه نقطه تهی باشد باز هم الگوریتم عمل می‌نماید) [۷].گام‌های بعدی به این ترتیبند: رئوس نیمه بالا با شروع از نقطه انتهایی چپ مرتب شده، رئوس پایین نیز به همین ترتیب مرتب می‌شوند و رئوس انتهایی از دو طرف به هم متصل می شوند. در ادامه، همانند دو الگوریتم قبلی رئوس v و u و r و t به همان ترتیب توضیح داده شده در بخش ۲-۳ برای این الگوریتم نیز درنظر گرفته شده(شکل a۴) و یالهای (v,r) , (u,t) را با (v,u) , (t,r) جایگزین می‌شوند شکل حاصل یک چندضلعی ساده تصادفی می‌باشد(شکل b۴). روند این الگوریتم با فرض داشتن الگوریتم TwoPeasants به صورت زیر می‌باشد:

- ابتدا $CH(S)$ روی مجموعه رئوس تصادفی S محاسبه می‌شود.
- مجموعه رئوس داخل $CH(S)$ را $\acute{S}$ در نظر گرفته و راس با کمترین مختصات y از این مجموعه رئوس انتخاب می‌شود(راس v).
- برای رئوس داخل پوسته محدب (مجموعه $\acute{S}$) الگوریتم TwoPeasants فراخوانی می‌شود.
- از مجموعه رئوس روی $CH(S)$ نزدیکترین راس به راس v انتخاب می‌شود(راس u).
- راس مجاور v در جهت پادساعتگرد r و راس مجاور u در جهت پادساعتگرد t نامگذاری می‌شوند.
- یالهای (v,r) , (u,t) با (v,u) , (t,r) جایگزین می‌شوند.

در بخش بعدی کارایی الگوریتم‌های پیشنهادی مطرح شده مورد بررسی قرار می‌گیرد.



شکل ۴: (a) ساخت یک چند ضلعی با الگوریتم TwoPeasants با نقاط داخل پوسته محدب، (b) ساخت چند ضلعی تصادفی با جا به جایی یالهای (v,r) , (u,t) با (v,u) , (t,r) .

۴ -بررسی کارایی

در این بخش کارایی و پیچیدگی زمانی الگوریتم‌های پیشنهادی بررسی می‌شود. در الگوریتم پیشنهادی ۱ از آنجا که مرتبه زمانی محاسبه $CH(S)$، $O(n\ logn)$ و پیدا کردن راسی با کمترین مختصات y، (راس v)، $O(n)$ است، همچنین چون پیچیدگی زمانی مرتب سازی مجموعه رئوس $\acute{S}$ بر حسب زاویه قطبی حول راس v، $O(n\ logn)$ و پیداکردن نزدیکترین راس به راس v، از مجموعه رئوس $CH(S)$، $O(n)$ بوده و جابه‌جایی یالهای (v,r), (u,t) با (v,u) , (t,r) در زمان خطی انجام می پذیرد بنابراین مرتبه زمانی این الگوریتم $O(n\ logn)$ می‌باشد.

برای الگوریتم پیشنهادی۲، چون مرتبه زمانی محاسبه $CH(S)$، $O(n\ logn)$ و پیدا کردن راسی با کمترین مختصات y، (راسv)، $O(n)$ بوده و همچنین مرتبه زمانی الگوریتم Convex Buttom، $O(n\ logn)$ می‌باشد و نیز چون پیدا کردن نزدیکترین راس به راس v، از مجموعه رئوس $CH(S)$، $O(n)$ بوده و جابه جایی یالها در زمان خطی انجام می پذیرد، لذا پیچیدگی زمانی این الگوریتم نیز $O(n\ logn)$ است.

در الگوریتم پیشنهادی ۳ مرتبه زمانی محاسبه $CH(S)$، $O(n\ logn)$ و پیدا کردن نقطه ای با کمترین مختصات y، (راسv)، $O(n)$ بوده و مرتبه زمانی الگوریتم TwoPeasants ، $O(n\ logn)$ می‌باشد، همچنین پیدا کردن نزدیکترین راس به راس v، از مجموعه رئوس $CH(S)$، $O(n)$ بوده و جابه جایی یالها در زمان خطی انجام می پذیرد، بنابراین مانند الگوریتم‌های قبل، پیچیدگی زمانی این الگوریتم نیز $O(n\ logn)$ خواهد بود. در بخش بعدی نتیجه کار مورد بررسی قرار می‌گیرد.

۵ -نتیجه گیری

در این مقاله، مروری بر الگوریتم‌های موجود برای تولید چندضلعی‌های تصادفی شد. الگوریتم‌های بسیار زیادی برای تولید چندضلعی‌های تصادفی ارائه شده‌اند اما تا کنون هیچ الگوریتمی برای مسئله تولید چندضلعی‌های ساده تصادفی با پیچیدگی زمانی خطی ارائه نشده و این مسئله باعث به کاربردن راهکارهای ابتکاری شده است. دراین مقاله سه الگوریتم ابتکاری برای تولید چندضلعی‌های تصادفی ساده که دارای پیچیدگی زمانی $O(n\ logn)$ می‌باشند مطرح شد که از برخی الگوریتم‌های موجود برای ساخت چند ضلعی های تصادفی با مرتبه زمانی $O(n\ logn)$ در آنها استفاده می شود.

مراجع

[۱] T.Auer, "Heuristic for generation of polygons". *In Proc.Canada. Conf. Compute. Geom* , pp. ٣٤-٤٤،١٩٩٤.

[۲] P.Epstein and J.-R.Sack, “Generating objects at Random” *Master's thesis CS Dept., Carleton university, Ottawa K١ S٥ B٦, Canada*, ١٩٩٢.

[۳] L.Derroy, P.Epstein and J.-R.Sack, “on generating random intervals and hyperrectangles” *J.Compute.Graph.Stat* , pp. ٢٩١-٣٠٧, ١٩٩٣.

[٤] M.D. Atkinson and J.-R.Sack, “Uniform generation of fore of restricted height” *Inform.Process.Lett*, pp. ٣٢٣-٣٢٧, ١٩٩٤.

[٥] C. Zhu, G.Sundaram, J.Snoeyink and J.S.B. Mitchell, “Generating random polygon with given vertices,” *Compute.Geom.:Theory Applic.,to appear* ١٩٩٦.

[٦] J.O'Rourke and M.Virmani, “generating random polygons” *Technical report* ١١ *.CS Dept.Smith College, MA, USA*, , ١٩٩١.

[٧] http://www.geometrylab.de/polygon/Random Polygon.html.en