الگوریتم‌هایی بهینه برای تولید چندضلعی‌های تصادفی مارپیچی و چندبخشی

لعیا محمدی[۱]، علی نوراله[۲] و سمیرا حسینی[۳]
[۱] دانشکده مهندسی برق و کامپیوتر، دانشگاه آزاد اسلامی قزوین، L.mohammadi@qiau.ac.ir
[۲] دانشکده مهندسی برق و کامپیوتر، دانشگاه آزاد اسلامی قزوین و دانشکده مهندسی برق و کامپیوتر، دانشگاه تربیت دبیر شهید رجایی تهران
[۳] دانشکده مهندسی برق و کامپیوتر، دانشگاه آزاد اسلامی قزوین، S.hosseini@qiau.ac.ir

چکیده

یکی از مسائل مهم در گرافیک کامپیوتری و هندسه محاسباتی تولید چندضلعی‌های تصادفی است. از آنجا که برای ارزیابی الگوریتم‌های گرافیکی و هندسی غالبا ممکن نیست تا مجموعه داده‌ای واقعی داشت، یک مجموعه داده تصادفی می‌تواند جایگزین مناسبی باشد. در این مقاله مسأله تولید چندضلعی‌های تصادفی ساده بر روی یک مجموعه از رئوس در نظر گرفته می‌شود که از پوسته‌های محدب فرضی و افراز فضایی برای تولید آنها استفاده می‌شود. از این چندضلعی‌ها می‌توان برای ارزیابی بسیاری از مسائل گرافیک کامپیوتری و هندسه محاسباتی مانند روشن‌سازی و موزه هنری استفاده کرد. از آنجا که تا کنون هیچ راه‌حلی با زمان چندجمله‌ای برای تولید تصادفی یکنواخت چندضلعی‌ها شناخته نشده است، می‌توان ثابت کرد تولید چندضلعی در زمان کمتر از $O(nlogn)$ امکان‌پذیر نیست، لذا الگوریتم‌های ارائه‌شده بهینه می‌باشند. در این مقاله دو الگوریتم ابتکاری جدید برای تولید چندضلعی‌های مارپیچی و چندبخشی با پیچیدگی زمانی $O(nlogn)$ ارائه شده است که بهینه هستند.

کلمات کلیدی

پوسته محدب، چندضلعی ساده تصادفی، قابلیت دید، گرافیک کامپیوتری.

۱- مقدمه

هندسه محاسباتی، یکی از شاخه‌های مهم گرافیک کامپیوتری است. اغلب محاسبات در هندسه محاسباتی و گرافیک کامپیوتری روی اشیاء هندسی شناخته شده مانند چندضلعی‌ها انجام می‌شوند. هر شیئی در طبیعت بصورت مجموعه‌ای از چندضلعی‌ها قابل نمایش است و چندضلعی ها یک مدل مناسب برای شبیه سازی اشیاء در دنیای واقعی هستند[1,2].

یکی از مسائل مطرح بهینه‌سازی در هندسه محاسباتی مسأله تولید چندضلعی‌های تصادفی می‌باشد. به دلیل اهمیت کاربرد چندضلعی‌ها این مسأله بسیار مورد توجه محققان قرار گرفته است. هیچ الگوریتمی با زمان چند جمله‌ای برای حل این مسأله وجود ندارد، لذا محققان سعی می‌کنند از الگوریتم‌های ابتکاری و بهینه استفاده کنند[3-5]. در این مسأله بهینه‌سازی تابع هدف زمان الگوریتم بوده و هدف مسأله کم کردن زمان تولید یک چندضلعی است. با توجه به اینکه نمی‌توان یک چندضلعی تصادفی را در زمان کمتر از $O(nlogn)$ تولید کرد لذا این زمان برای این مسأله بهینه می‌باشد. از آنجا که مجموعه داده‌ای واقعی برای ارزیابی و آزمایش الگوریتم‌های هندسی وجود ندارد یک جایگزین مناسب برای آنها، مجموعه داده تصادفی است. نمونه‌هایی از کاربرد آنها شامل حروف منفرد برای الگوی خودکار، یک مانع برای یک محیط ربات و یا جسمی که باید روی صفحه گرافیکی نمایش داده شود است. همچنین چندضلعی‌ها برای ارائه انواع وسیعی از شکل‌های گرافیک کامپیوتری، بینایی ماشین، تشخیص الگو و دیگر زمینه های هندسی استفاده می‌شوند[3-6].

مسأله 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 T چندضلعی ساده روی S وجود داشته باشد تولید هر یک از این چندضلعی‌ها با احتمال $\frac{1}{T}$ مدنظر است)[7].

[۲] استادیار

از مسائل مهمی که چند ضلعی‌ها در آن نقش عمده‌ای دارند، روشن سازی و موزه هنری است که هدف آن حفاظت یک موزه هنری چندضلعی شکل و پوشش آن با کمترین تعداد نگهبان یا دوربین است. هدف این مقاله ارائه الگوریتم‌هایی برای تولید چندضلعی‌های تصادفی جهت ارزیابی این مسائل در بدترین حالات است.

ادامه مقاله به این شکل سازماندهی شده است: بخش ۲ به کارهای انجام شده برای تولید چندضلعی‌های تصادفی می پردازد. در بخش ۳ مفاهیم اولیه مورد نظر بیان می‌شود. در بخش ۴، دو الگوریتم ابتکاری بهینه برای تولید چندضلعی‌های تصادفی مارپیچی و چندبخشی مطرح شده و کارایی و صحت آنها مورد بررسی قرار می‌گیرند و نهایتا در بخش ۵ نتیجه گیری مطرح می‌شود.

۲- کارهای انجام شده

مسأله تولید تصادفی چندضلعی‌های ساده بسیار مورد توجه محققان بوده است. Epstein تولید یکنواخت تصادفی مثلث بندی‌ها را مورد مطالعه قرار دارد[1].

الگوریتمی برای تولید تصادفی چندضلعیهای monotone روی یک مجموعه نقاط داده شده توسط Zhu ارائه شد[5]. O'Rouke، Virmani یک الگوریتم ابتکاری برای تولید چندضلعی‌های ساده ارائه دادند[8]. Auer و Held الگوریتم های زیر را برای تولید چندضلعی‌های تصادفی ارائه دادند:

Steady Growth: این الگوریتم بطور افزایشی یک نقطه را بعد از دیگری اضافه می‌کند که در بدترین حالت پیچیدگی زمانی$O(n^2)$ و در بهترین حالت پیچیدگی زمانی برابر با $O(nlogn)$ دارد.

Space Partitioning: این الگوریتم مبتنی بر الگوریتم تقسیم و غلبه است که دارای پیچیدگی زمانی$O(n^2)$ می‌باشد.

Permute & Reject: این الگوریتم جایگشت‌هایی (چندضلعی هایی) تصادفی ایجاد می‌کند و بررسی می‌کند که آیا چندضلعی حاصل با یک چندضلعی ساده متناظر است یا خیر. این روند تا وقتی ادامه پیدا می‌کند که الگوریتم با یک چندضلعی ساده مواجه شود. پیچیدگی زمانی این الگوریتم$O(nlogn)$ است.

2-opt move: این الگوریتم از یک چندضلعی کاملا تصادفی شروع کرده و آنقدر یال‌های متقاطع را جستجو و جابه جا می‌کند تا به یک چندضلعی ساده برسد و پیچیدگی آن $O(n^4)$ است[7] .

۳- مفاهیم اولیه

اگرS یک مجموعه رئوس تصادفی باشد،T چندضلعی ساده روی S می‌تواند وجود داشته باشد که هر چندضلعی با احتمال $\frac{1}{T}$ تولید شود. فرض می‌شود که هیچ سه نقطه‌ای همراستا (همخط) نیستند. یک چندضلعی ساده، مجموعه محدودی از پاره‌خط‌هایی است که یک محیط بسته را تشکیل می‌دهند بطوریکه هیچ دو پاره‌خطی جز در نقاط انتهایی همدیگر را قطع نکنند، به عبارت دیگر، یک چندضلعی سادهP روی S ، یک چندضلعی است که یال‌های آن جز در رئوسS همدیگر را قطع نمی‌کنند. یک چندضلعی محدب، یک چندضلعی ساده است که هیچ یک از زوایای داخلی آن بزرگتر از ۱۸۰ درجه نیست. پوسته محدب مجموعه نقاط S در صفحه$(CH(S))$، کوچکترین چندضلعی محدبP است که S را محصور می‌کند.

فرض می‌شود k تعداد لایه‌های $CH(S)$ باشد، هر لایه باl_c معرفی می‌شود($1 \le l_c \le k, k > 1$). با فرض شماره گذاری لایه‌ها از بیرونی‌ترین لایه تا درونی‌ترین لایه پوسته محدب فرض می‌شود n_i تعداد رئوس l_1 باشد. تعریف می‌شود راس y از راسx قابل دید است اگر$P \subseteq \overline{xy}$ و $\overline{xy}$ خارج ازP واقع نباشد، به عبارتی x نسبت به y قابلیت دید دارد. یك چندضلعی مارپیچی، یک چندضلعی است که با داشتن لایه‌های محدب تودرتوی مجموعه نقاط تصادفی، از طریق برقراری اتصال بین هر دولایه متوالی به دست می‌آید. یك چندضلعی چندبخشی، یك چندضلعی حاصل از افراز فضایی مجموعه نقاط است که هر بخش حاصل، یك چندضلعی مارپیچی دارد. در بخش بعدی دو الگوریتم پیشنهادی بهینه برای تولید چندضلعی‌های تصادفی مارپیچی و چندبخشی مطرح می‌شود.

۴- الگوریتم‌های پیشنهادی

در این بخش دو الگوریتم ابتکاری بهینه برای تولید چندضلعی های تصادفی مارپیچی و چندبخشی با پیچیدگی زمانی $O(nlogn)$ که در آن n تعداد رئوس چند ضلعی تصادفی ارائه شده است.

۴-۱- الگوریتم تولید چندضلعی مارپیچی

در این الگوریتم پیشنهادی که یک چندضلعی تصادفی مارپیچی روی مجموعه رئوسS ایجاد می‌کند، ابتدا طبق الگوریتم حریصانه گراهام[1]، $CH(S)$ به دست می‌آید. پیچیدگی زمانی الگوریتم گراهام، $O(nlogn)$ است. پس از تعیین $CH(S)$،برای هر مجموعه نقاط باقیمانده S تا وقتی که n مخالف صفر باشد مجدداً الگوریتم گراهام فراخوانی می‌شود تا همه لایه‌های پوسته

محدب به دست آیند. روند کلی الگوریتم پیشنهادی به شرح زیر است:

گام ۱- تا زمانی که n مخالف صفر باشد، هر لایه پوسته محدب طبق الگوریتم گراهام روی مجموعه نقاط S محاسبه و رسم می‌شود. برای $k \geq 2$ گام‌های بعد اجرا می‌شوند.

گام ۲- برای لایه‌های پوسته محدب l_k تا l_2 (از بیرونی‌ترین لایه تا درونی‌ترین لایه) روند زیر تکرار می‌شود:

گام ۲-۱- برای دولایه پوسته محدبl_k و l_{k-1}، یک یال دلخواه (e,d) روی لایهl_k انتخاب می‌شود.

گام ۲-۲- از لایهl_{k-1}، راسی مانند a که نزدیکترین راس به (e,d) می‌باشد و دو راس مجاورش به ترتیب با نام‌هایb و c در نظر گرفته می‌شوند.

گام ۲-۳- اگر در جهت خلاف عقربه‌های ساعت، راس b از راس e قابل دید بود یال‌های(b,a) و(e,d) حذف می‌شوند و یال‌های(b,e) و (a,d) جایگزین می‌شوند، در غیر این صورت یال‌های (a,c) و (e,d) حذف می‌شوند و یال‌های (a,e) و (c,d) جایگزین می‌شوند.

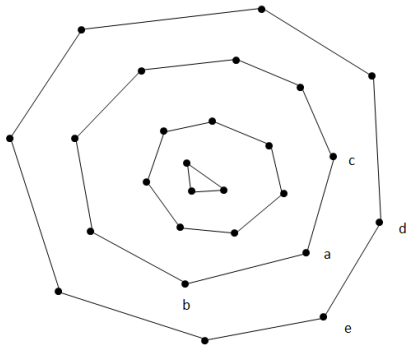
گام ۳- برای دو لایهl_2 و l_1 :

گام ۳-۱- اگر $l_1 \geq 3$ ، مرحله ۲ اجرا می‌شود.

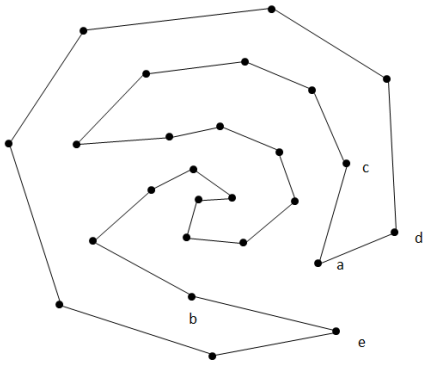
گام ۳-۲- اگر $l_1 = 2$، یک یال دلخواه (e,d) از لایه l_2 انتخاب می‌شود و راسی مانند a که نزدیک‌ترین راس به یال (e,d) می‌باشد از لایه l_1 در نظر گرفته می‌شود، حال اگر راس b از راس e قابل دید بود، یال‌های(b,a) و (e,d) حذف می‌شوند و یال‌های (b,e) و(a,d) جایگزین می‌شوند، در غیر این صورت یال‌های(b,a) و (e,d) حذف می‌شوند و یال‌های(a,e) و (b,d) جایگزین می‌شوند.

گام ۳-۳- اگر$l_1 = 1$، با فرض اینکه راس منفرد l_1، a باشد یال(e,d) حذف شده و یال‌های(a,e) و (a,d) اضافه می‌شوند (شکل۱ و شکل۲).

در بخش بعد کارایی الگوریتم مارپیچی پیشنهادی مورد بررسی قرار می‌گیرد.



شکل (۱): پیدا کردن یک یال دلخواه (e,d) و نزدیک‌ترین نقطه به آن از لایه داخلی



شکل (۲): ساخت یک چندضلعی تصادفی مارپیچی

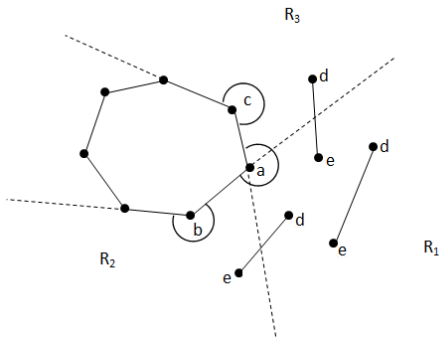
۴-۲- بررسی کارایی الگوریتم چندضلعی مارپیچی

در این بخش کارایی الگوریتم پیشنهادی تولید چندضلعی تصادفی مارپیچی با محاسبه پیچیدگی زمانی آن بررسی می‌شود. در این الگوریتم زمان به دست آوردن بیرونی‌ترین لایه پوسته محدب طبق الگوریتم گراهام، $O(nlogn)$ می‌باشد که چون n نقطه محاسبه می‌شود، بیشترین زمان موردنیاز برای تشکیل پوسته محدب خواهد بود. در هر دو لایه محدب تو در تو پیدا کردن راسی مانند a، از لایه داخلی که نزدیک‌ترین راس به یال دلخواه (e,d) از لایه بیرونی باشد، حداکثر $O(n)$ خواهد بود بنابراین در مجموع پیچیدگی زمانی این الگوریتم$O(nlogn)$ خواهد بود.

۴-۳- بررسی صحت الگوریتم چندضلعی مارپیچی

در این بخش صحت الگوریتم ابتکاری مطرح شده بررسی می‌شود. برای اثبات صحت درستی این الگوریتم روی قابلیت دید بیرونی رئوس مجاور a یعنی b و c بحث می‌شود و با فرض اینکه راس a نزدیکترین راس از لایه داخلی به یال (e,d) است، نشان داده می‌شود که یال (e,d) حتما در این ناحیه قابل دید قرار دارد. ابتدا با امتداد دادن یال‌های مجاور راس c از دو راس مجاور c و نیز با امتداد دادن یال‌های مجاور راس b از دو راس مجاور آن ناحیه قابل دید بیرونی رئوس b و c به دست می‌آید و بدیهی است که ناحیه قابل دید راس a نیز در این ناحیه قرار دارد.

حال با فرض ۳ ناحیه مشخص شده R_1 ، R_2 و R_3 در شکل ۳، اگر (e,d) در ناحیه R_1 باشد واضح است که از دو راس یا هر سه راس مفروض قابل دید خواهد بود و اگر یک راس آن در یکی از دو ناحیه مجاور باشد، هر دور راس آن از راس a و یکی از رئوس مجاور قابل دید خواهد بود (هر راس مفروض (e,d) از یکی از رئوس مفروض b و a یا a و c قابل دید است). بنابراین بدیهی است که یال (e,d) حتما نسبت به یال‌های (a,b) یا (a,c) قابلیت دید دارد.



شکل (۳): قابلیت دید یال (e,d) از راس a و یک راس مجاور

۴-۴- الگوریتم تولید چندضلعی چندبخشی

در این بخش یک الگوریتم ابتکاری برای تولید چندضلعی‌های تصادفی چندبخشی ارائه می‌شود که از الگوریتم مطرح شده بخش ۴-۱ استفاده می‌کند. پیچیدگی زمانی این الگوریتم نیز برابر با $O(nlogn)$ است. در این الگوریتم مانند الگوریتم قبل ابتدا طبق الگوریتم حریصانه گراهام، $CH(S)$ فرضی روی مجموعه نقاطS به دست می‌آید. سپس برای هر مجموعه نقاط باقیمانده S تا وقتی که n مخالف صفر باشد مجددا الگوریتم گراهام فراخوانی می‌شود و همه پوسته‌های محدب تو در توی فرضی به دست می‌آید.

اگر تعداد نقاط روی داخلی‌ترین لایه پوسته محدب فرضی یعنی l_1، n_i باشد می‌توان مجموعه نقاط S را به قسمت n_i افراز کرد. این افراز به این صورت است که با شروع از چپ‌ترین نقطه روی l_1 و در خلاف جهت عقربه‌های ساعت، هر یال l_1 از نقطه دوم امتداد می‌یابد. به این ترتیب n_i مجموعه نقاط به دست می‌آید. پس از افراز مجموعه نقاط S برای هر مجموعه نقاط افراز شده حاصل، دو نقطه متوالی داخلی‌ترین لایه که در آن ناحیه واقع شده‌اند نیز جزء این نقاط به حساب می‌آیند. بنابراین هر نقطه داخلی‌ترین پوسته محدب، در هر دو ناحیه مجاورش محسوب می‌شود. سپس برای هر مجموعه نقاط افراز شده الگوریتم قبلی فراخوانی می‌شود با این تفاوت که در لایه l_k (بیرونی‌ترین لایه) یال مربوط به داخلی‌ترین لایه پوسته محدب در این ناحیه نباید به عنوان یال دلخواه انتخاب شود. با این فرض یک یال دیگر از این ناحیه انتخاب می‌شود و الگوریتم ۱ برای هر بخش افراز شده فراخوانی می‌شود. در پایان اگر مجموعه نقاط یك ناحیه بیشتر از ۲ باشد، یال مربوط به داخلی ترین لایه پوسته محدب یعنی l_1 در آن ناحیه حذف می‌شوند. به این ترتیب در هر بخش افراز شده یک چندضلعی مانند چندضلعی‌های تصادفی حاصل از الگوریتم ۱ ایجاد می‌شود.

یکی از ویژگی‌های بارز چندضلعی تصادفی حاصل این است که می‌تواند جهت ارزیابی بسیاری از مسائل هندسی مانند روشن‌سازی و موزه هنری در بدترین حالت به کار رود. روند کلی این الگوریتم پیشنهادی به ترتیب زیر است:

گام ۱- تا زمانی که n مخالف صفر باشد، همه لایه‌های $CH(S)$ فرضی طبق الگوریتم گراهام روی مجموعه نقاط S محاسبه می‌شود.

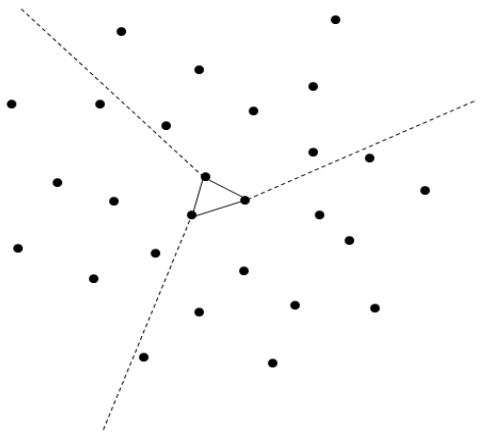
گام ۲- تعداد نقاط پوسته محدب فرضی l_1 شمارش شده و n_i فرض می‌شود.

گام ۳- هر لایه فرضی v_jv_{j+1} $(1 \le j \le n_i)$ روی l_1 از راس v_{j+1} امتداد می‌یابد و در نهایت $v_1v_{n_i}$ از راس v_1 امتداد می‌یابد. بنابراین مجموعه نقاط به n_i مجموعه نقطه افراز می‌شوند(هر نقطه از داخلی‌ترین لایه پوسته محدب جزء مجموعه نقاط دو ناحیه افراز شده مجاورش محسوب می‌شود).

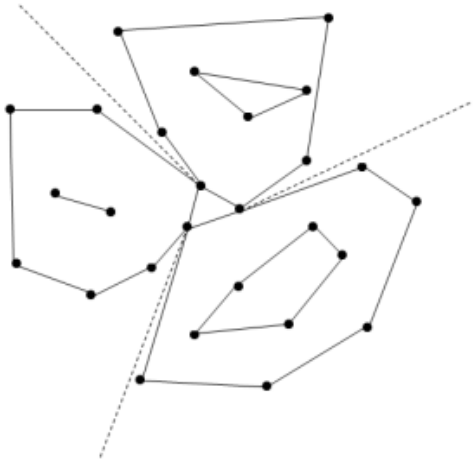
گام ۴- برای هر مجموعه نقاط حاصل از افراز نقاط الگوریتم بخش ۴-۱فراخوانی می‌شود ولی با این تفاوت که در لایه l_k (بیرونی‌ترین لایه) یال مربوط به داخلی‌ترین لایه پوسته محدب فرضی اولیه در این ناحیه نباید به عنوان یال دلخواه انتخاب شود. با این فرض یک یال دیگر از این ناحیه انتخاب می‌شود و الگوریتم مطرح شده بخش ۴-۱برای هر بخش افراز شده فراخوانی می‌شود.

گام۵- برای هر مجموعه نقاط افراز شده اگر با فرض موجود در گام ۳، تعداد نقاط آن ناحیه بیشتر از ۲ باشد، یال مربوط به داخلی‌ترین پوسته محدب اولیه حذف می‌شود.(شکل۴، شکل ۵ و شکل ۶).

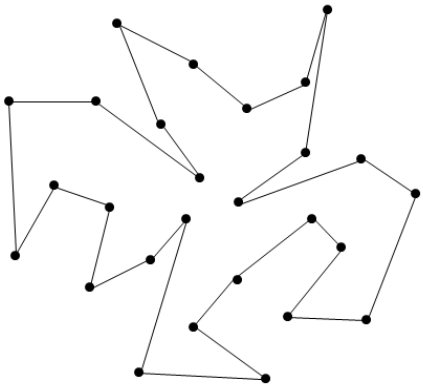
در حالت خاص که $n_i = 1$، چندضلعی حاصل همان چندضلعی حاصل از الگوریتم بخش ۴-۱ خواهد بود. در بخش بعدی بررسی کارایی الگوریتم پیشنهادی مورد بررسی قرار می‌گیرد.



شکل (٤): افراز مجموعه نقاطS با امتداد دادن راس دوم هر یال l_1



شکل (۵): پیدا کردن لایه‌های محدب تو در تو در هر ناحیه افراز شده



شکل (٦): ساخت یک چند ضلعی تصادفی چندبخشی با فراخوانی الگوریتم بخش ۴-۱ در هر بخش افراز شده

۴-۵- بررسی کارایی الگوریتم چندضلعی چندبخشی

در این بخش کارایی الگوریتم پیشنهادی تولید چندضلعی چندبخشی با محاسبه پیچیدگی زمانی آن بررسی می‌شود. در این الگوریتم زمان محاسبه بیرونی‌ترین لایه پوسته محدب طبق الگوریتم گراهام، $O(nlogn)$ است که چون روی n نقطه محاسبه می‌شود، زمان موردنیاز برای تشکیل پوسته محدب خواهد بود. افراز نقاط S به n_i مجموعه نقطه، $O(n_i)$ زمان نیاز دارد که در بدترین حالت یعنی حالتی که فقط یک بخش حاصل از افراز نقاط وجود داشته باشد برابر با $O(1)$ است.

فراخوانی الگوریتم بخش ۴-۱ برای هر بخش نقاط افراز شده در صورتی که پراکندگی مجموعه نقاط مناسب باشد یعنی به طور متوسط در هر بخش $\frac{n}{n_i}$ نقطه وجود داشته باشد $O\left(n\ log\frac{n}{n_i}\right)$ است و برای کل بخش‌ها $O\left(\frac{n}{n_i}log\frac{n}{n_i}\right)$ خواهد بود. در بدترین حالت که فقط یک بخش افراز شده وجود داشته باشد فراخوانی این الگوریتم $O(nlogn)$ زمان می‌برد.

بنابراین در مجموع پیچیدگی زمانی الگوریتم پیشنهادی $O(nlogn)$ می‌باشد.

۴-۶- بررسی صحت الگوریتم چندبخشی

در این بخش صحت الگوریتم ابتکاری مطرح شده بررسی می‌شود که این الگوریتم در حالت کلی برای هر مجموعه نقاط داده شده با تعداد لایه‌های پوسته محدب فرضی $k > 1$ درست عمل می‌کند. برای اثبات صحت درستی این الگوریتم روی هر مجموعه نقاط افراز شده و برقراری صحت الگوریتم پیشنهادی بخش ۴-۱ روی این نقاط بحث می‌شود. از آنجا که صحت الگوریتم پیشنهادی بخش ۴-۱ در بخش ۴-۳ اثبات شد برای هر ناحیه افراز شده، صحت درستی آن برقرار می‌باشد چرا که همه بخش‌های افراز شده کاملا از هم مستقل بوده و هیچ اشتراکی با هم ندارند بنابراین به سادگی قابل درک است که حتما یک چندضلعی تصادفی با الگوریتم پیشنهادی بر روی هر مجموعه نقاط داده با تعداد لایه‌های پوسته محدب فرضی $k > 1$ قابل ساخت است.

۵- نتیجه

الگوریتم‌های بسیاری برای تولید چندضلعی‌های تصادفی ارئه شده اند اما از آنجا هیچ الگوریتمی برای این مسأله با پیچیدگی زمانی خطی ارائه نشده است محققان از راهکارهای ابتکاری استفاده می‌کند. در این مقاله دو الگوریتم بهینه با پیچیدگی زمانی$O(nlogn)$ برای تولید دو نوع چندضلعی تصادفی خاص مارپیچی و چندبخشی پیشنهاد شد. از این الگوریتم‌ها می‌توان برای ارزیابی بسیاری از الگوریتم‌ها و مسائل گرافیک کامپیوتری و هندسه محاسباتی مانند روشن‌سازی و موزه هنری استفاده کرد. همچنین ثابت شده این الگوریتم بر روی هر مجموعه نقاط داده شد. با پوسته‌های محدب فرضی تو در تو (با تعداد لایه های $k > 1$) درست عمل کرده و چندضلعی‌های تصادفی خاصی تولید می‌کند.

مراجع

[1] M. D. Berg, "*Computational Geometry: Algorithms and Applications"*, 3rd , *Published by springer-verlag*, 2008.

[2] T.Auer, "*Heuristic for generation of polygons*". *In Proc.Canada. Conf. Compute. Geom* , pp. 34-44,1994.

[3] M.D. Atkinson and J.-R.Sack, "*Uniform generation of fore of restricted height*", *Inform.Process.Lett*, pp. 323-327, 1994.

[4] C. Zhu, G.Sundaram, J.Snoeyink and J.S.B. Mitchell, "*Generating random polygon with given vertices*", *Compute.Geom.:Theory Applic.,to appear* 1996.

[5] C. Sohler, "*Generating random star-shaped polygons*", *In Proc. 11th Canadian Conference on Computational Geometry(CCCG'99)*, pp. 174-177, 1999.

[6] C. Zhu, G. Sundaram, J. Soneyink, and J. S. B. Mitchell, "*Generating Random Polygon with Given Vertices*", *Comput. Geom. Theory and Appl.*, Vol 6, Issues 5, pp. 277-290, 1996.

[7] T. Auer, M. Held, "*Heuristics for the Generation of Random Polygons*", *Proc. 8th Canadian on Computational Geometry(CCCG'96)*, pp. 38-41, 1996.

[8] J. O'Rourke and M. Virmani, "*Generating Random Polygons*", *Technical Report 011, CS Dept. Smith Colege, Northhampton, MA 01063*, July 1991.