به نام خدا

دانشگاه صنعتی شریف

دانشکده‌ی مهندسی کامپیوتر

پایان‌نامه‌ی کارشناسی

گرایش نرم‌افزار

عنوان:

مسئله‌ی راه فرار مستطیل‌ها

نگارش:

احسان امام‌جمعه‌زاده

سپهر اسدی

استاد راهنما:

دکتر حمید ضرابی‌زاده

تیر ماه ۱۳۹۲

دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

پایان‌نامه‌ی کارشناسی
گرایش نرم‌افزار

عنوان:

مسئله‌ی راه فرار مستطیل‌ها

نگارش:

احسان امام‌جمعه‌زاده

سپهر اسدی

استاد راهنما:

دکتر حمید ضرابی‌زاده

امضای استاد ممتحن: امضای استاد راهنما: نمره:

چکیده

در این نوشتار به بررسی *مسئله‌ی راه فرار مستطیل‌ها* می‌پردازیم که در مسیردهی بر روی بردهای دیجیتال مورد نیاز است. مسئله‌ی راه فرار مستطیل‌ها را می‌توان این‌گونه تعریف کرد: مجموعه‌ی S شامل n مستطیل داده شده که همگی بر روی یک ناحیه‌ی مستطیلی بزرگ به نام *قاب* قرار گرفته‌اند، به شکلی که مرزهای مستطیل‌ها همگی موازی اضلاع قاب هستند. هدف، فراری دادن هر کدام از این مستطیل‌های عضو S به یکی از چهار جهت اصلی است، به گونه‌ای که بیشینه چگالی نقاط قاب، کمینه شود. برای هر نقطه‌ی p از قاب، چگالی نقطه‌ی p، تعداد مستطیل‌هایی است که در فرارشان از آن نقطه عبور می‌کنند.

در این مقاله، ابتدا سختی این مسئله را بررسی می‌کنیم و نشان می‌دهیم که با فرض $P \neq NP$، الگوریتمی چندجمله‌ای با ضریب تقریب بهتر از $\frac{۳}{۲}$ برای این مسئله وجود ندارد. سپس برای حالتی که بیشینه چگالی مجاز برابر با یک است، یک الگوریتم چندجمله‌ای با زمان اجرای $O(n^۴)$ ارائه می‌کنیم که در مقایسه با الگوریتم پیشین این مسئله با زمان اجرا $O(n^۶)$، از زمان اجرای بهتری برخوردار است. در پایان، به ازای هر مقدار $\epsilon > ۰$، با این شرط که مقدار جواب به اندازه‌ی کافی بزرگ باشد، یک الگوریتم تقریبی احتمالی با ضریب تقریب $۱ + \epsilon$ ارائه می‌دهیم. این درحالی است که ضریب تقریب بهترین الگوریتم تقریبی قبلی ۴ است.

فهرست مطالب

لیست تصاویر

فصل ۱

مقدمات

در این بخش، به معرفی برخی از مفاهیمی می‌پردازیم که در این مقاله استفاده شده‌اند.

۱-۱ کلاس‌های پیچیدگی

مهم‌ترین کلاس‌های پیچیدگی در ادامه معرفی شده‌اند. پیش از آن، گفتنی است به یک مسئله که پاسخ آن *بلی* یا خیر باشد، مسئله‌ی *تصمیم‌گیری*[۱] گفته می‌شود.

کلاس پیچیدگی P: مجموعه‌ای از مسئله‌های تصمیم‌گیری که برای آن‌ها الگوریتم (قطعی) با زمان اجرای چندجمله‌ای وجود دارد.

کلاس پیچیدگی NP: مجموعه‌ای از مسئله‌های تصمیم‌گیری که وقتی به ازای یک ورودی، پاسخ *بلی* باشد، این ادعا را می‌توان در زمان چندجمله‌ای ثابت کرد.

با توجه به تعاریف ارائه شده، بدیهی است که $P \subseteq NP$. یک پرسش بسیار مشهور است این است که آیا $P = NP$ یا خیر؟ در حال حاضر، حدس بسیار قوی در رابطه با پاسخ این پرسش این است که NP با P برابر نیست.

[۱] Decision Problem

کلاس پیچیدگی NP–سخت[۲]: مجموعه‌ی مسئله‌هایی است که در صورت حل شدن یکی از آن‌ها در زمان چندجمله‌ای، همه‌ی مسئله‌های NP در زمان چندجمله‌ای حل خواهند شد. پس با فرض $NP \neq P$، هیچ یک از مسئله‌های NP–سخت، الگوریتم چندجمله‌ای ندارند. شایان ذکر است که برخی از مسئله‌های NP–سخت ممکن است مسئله‌ی تصمیم‌گیری نباشند.

کلاس پیچیدگی NP–کامل[۳]: مجموعه‌ی مسئله‌هایی است که هم NP و هم NP–سخت باشند.

۱–۲ الگوریتم‌های تقریبی

مسئله‌های بهینه‌سازی گسسته، مسئله‌هایی هستند که در آن‌ها، هدف تصمیم‌گیری به منظور کمینه کردن هزینه (یا بیشینه کردن سود) است. اکثر مسئله‌های مورد توجه در این مجموعه، مسئله‌های هستند که NP–سخت بودن آن‌ها نشان داده شده‌است و در نتیجه، با این فرض که $NP \neq P$، نمی‌توان برای یافتن پاسخ بهینه‌ی چنین مسئله‌هایی، راه حلی کارا ارائه کرد.

نظر به دشواری یافتن پاسخ بهینه‌ی چنین مسئله‌هایی، یک رویکرد رایج، ارائه‌ی الگوریتم‌های کارایی است که پاسخی نه چندان دور از پاسخ بهینه می‌یابند. به چنین الگوریتم‌هایی، *الگوریتم‌های تقریبی*[۴] گفته می‌شود. یک الگوریتم تقریبی برای یک مسئله‌ی کمینه‌سازی دارای *ضریب تقریب*[۵] α است اگر به ازای همه‌ی ورودی‌ها، پاسخی حداکثر α برابر پاسخ بهینه بیابد.

۱–۳ تعاریف و قضایای احتمال

از آن‌جایی که بخشی از نتایج این نوشتار، بر پایه‌ی الگوریتم‌های تصادفی[۶] و نامساوی‌های احتمالاتی به دست آمده‌اند، به بیان چند تعریف پایه و سپس قضایای مورد استفاده می‌پردازیم.

[۲] NP-Hard

[۳] NP-Complete

[۴] Approximation Algorithms

[۵] Approximation Factor

[۶] Randomized Algorithms

- **فرایند تصادفی**[۷]: به هر فرایندی که نتیجه آن نه به صورت قطعی که به صورت احتمالاتی مشخص می‌شود، فرایند تصادفی گفته می شود. به عنوان مثال، پرتاب یک سکه، یک فرایند تصادفی است؛ چراکه نتیجه‌ی آن به صورت احتمالاتی شیر یا خط است.

- **فضای نمونه**[۸]: به مجموعه‌ی نتیجه‌هایی که یک فرایند تصادفی می‌تواند اتخاذ کند، فضای نمونه‌ی آن فرایند تصادفی گفته می‌شود. به عنوان مثال، فضای نمونه‌ی فرایند تصادفی پرتاب سکه، مجموعه {شیر، خط} است.

- **متغیر تصادفی**[۹]: تابعی از فضای نمونه‌ی یک فرایند تصادفی به مجموعه‌ی اعداد حقیقی، یک متغیر تصادفی است. به عنوان مثال، متغیر تصادفی x_i را می‌توان معادل با شیر آمدن سکه پس از پرتاب iم تعریف کرد که در صورت شیر آمدن سکه، مقدار ۱ و در غیر این صورت، مقدار ۰ می‌گیرد. در این نوشتار، برد همه‌ی متغیرهای تصادفی، زیرمجموعه‌ای متناهی از مجموعه‌ی اعداد صحیح است.

- **متغیر تصادفی شناسه**[۱۰]: یک متغیر تصادفی که برد آن $\{۰,۱\}$ است (یک زیرمجموعه از فضای نمونه را به ۱ و باقی را به ۰ می‌برد)، یک متغیر تصادفی شناسه است. به بیانی دیگر، می‌توان گفت که ۱ بودن یک متغیر تصادفی شناسه معادل با اتفاق افتادن عضوی از زیرمجموعه‌ی مورد نظر است.

- **امید ریاضی**[۱۱]: امید ریاضی یک متغیر تصادفی که برد آن، مجموعه‌ی متناهی از $\mathbb{R}$ است، به صورت زیر تعریف می‌شود:

$$E[X] = \Sigma_{x\in R} P(X=x) \times x$$

برای یک متغیر تصادفی شناسه بنا به تعریف داریم:

$$E[X] = P(X=۱) \times ۱ + P(X=۰) \times ۰ = P(X=۱)$$

[۷] Stochastic Process

[۸] Sample Space

[۹] Random Variable

[۱۰] Indicator Random Variable

[۱۱] Expected Value

به بیان دیگر، برای یک متغیر تصادفی شناسه، امید ریاضی برابر است با احتمال ۱ شدن مقدار آن متغیر.

- **واریانس**[۱۲]: واریانس معیاری است برای نشان دادن فاصله‌ی مقادیر یک متغیر تصادفی از امید ریاضی آن. به طور رسمی واریانس به صورت زیر تعریف می‌شود:

$$var(X) = E[X^2] - (E[X])^2$$

هم‌چنین $\sigma(X)$ را برابر با $\sqrt{var(X)}$ تعریف می‌کنند و آن را *انحراف معیار*[۱۳] متغیر تصادفی X می‌نامند.

در ادامه به چند قضیه‌ی مهم و اساسی احتمالاتی اشاره می‌کنیم:

قضیه ۱ (خطی بودن امید ریاضی) *امید ریاضی خطی است. به بیان دقیق‌تر، برای هر دو متغیر تصادفی X و Y و عدد ثابت c، روابط زیر برقرارند:*

$$E[cX] = cE[x]$$

$$E[X + Y] = E[X] + E[Y]$$

لازم به یادآوری است که تساوی دوم در قضیه‌ی ۱ به ازای هر دو متغیر تصادفی X و Y برقرار است؛ چه مستقل باشند و چه نباشند.

قضیه ۲ (نامساوی‌های احتمالاتی) *در این قضیه، چند نامساوی احتمالاتی معروف که در تحلیل الگوریتم‌های احتمالی کاربرد دارند، بیان می‌شوند.*

- *اگر $X = \{x_1, \ldots, x_k\}$ مجموعه‌ای از متغیرهای تصادفی باشد که دامنه‌ی همه‌ی آن‌ها $\{0, 1\}$ است، احتمال آن‌که مقدار حداقل یک $x_i \in X$ برابر ۱ شود، حداکثر $\Sigma_{x_i \in X} P(x_i = 1)$ است. دقت کنید که $\Sigma_{x_i \in X} P(x_i = 1)$، مجموع احتمال ۱ شدن این متغیرهاست.*

- ***نامساوی مارکوف***[۱۴]*:*

[۱۲] Variance

[۱۳] Standard Deviation

[۱۴] Markov Inequality

اگر برد متغیر تصادفی X، زیرمجموعه‌ای از اعداد طبیعی باشد، آن‌گاه به ازای هر $a > ۰$، داریم:

$$P(X \geq a) \leq \frac{E[X]}{a}$$

یا به طور معادل، می‌توان نوشت به ازای هر $c > ۰$:

$$P(X \geq cE[X]) \leq \frac{۱}{c}$$

به بیان دیگر، احتمال آن‌که مقدار X از c برابر مقدار امید ریاضی بیش‌تر شود، حداکثر $\frac{۱}{c}$ است.

- ***نامساوی چبیچو***[۱۵]*:*

اگر برد متغیر تصادفی X، زیرمجموعه‌ای از اعداد صحیح باشد، آن‌گاه به ازای هر $c > ۰$ داریم:

$$P(|X - E[X]| \geq c\sigma(X)) \leq \frac{۱}{c^۲}$$

که در آن $\sigma(X) = \sqrt{var(X)}$.

- ***نامساوی چرنوف***[۱۶]*:*

اگر $x_۱, \ldots, x_n$ متغیرهای تصادفی مستقل از هم با دامنه‌ی $\{۰, ۱\}$ باشند و متغیر تصادفی X به صورت $X = \Sigma_{i=۱}^{n} x_i$ تعریف گردد، آن گاه داریم:

$$P\{X \geq (۱+\epsilon)E[X]\} < \left(\frac{e^{\epsilon}}{(۱+\epsilon)^{(۱+\epsilon)}}\right)^{E[X]}$$

۱-۴ برنامه‌ریزی خطی

هدف از یک برنامه‌ریزی خطی[۱۷]، یافتن یک بردار با مؤلفه‌های نامنفی است که در نامساوی‌های خطی داده‌شده صدق کند و مقدار یک عبارت خطی را نیز کمینه نماید. این عبارت خطی، *تابع*

[۱۵] Chebyshev Inequality

[۱۶] Chernoff Inequality

[۱۷] Linear Programming (LP)

هدف[۱۸] نامیده می‌شود. فرض کنید $x_۱, \ldots, x_n$ متغیرهایی با دامنه‌ی اعداد گویا باشند و

$$c_j \in \mathbb{Q}, \quad b_i \in \mathbb{Q}, \quad a_{i,j} \in \mathbb{Q} \quad (۱ \leq i \leq m, ۱ \leq j \leq n)$$

یک برنامه‌ریزی خطی در حالت کلی به شکل زیر است:

$$\begin{aligned}
&\text{minimize} && \sum_{j=1}^{n} c_j x_j \\
&\text{subject to} && \sum_{j=1}^{n} a_{i,j} x_j \geq b_i && \forall\ 1 \leq i \leq m \\
& && x_j \geq 0 && \forall\ 1 \leq j \leq n
\end{aligned}$$

در این برنامه‌ریزی خطی، عبارت خطی $\sum_{j=۱}^{n} c_j x_j$، تابع هدف است. بدیهی است که اگر قصد بیشینه کردن تابع هدف را داشته‌باشیم، کافی است همه‌ی ضرایب c_j در تابع هدف را قرینه کنیم. به این ترتیب، تابع هدفی به دست می‌آید که می‌خواهیم مقدار آن کمینه شود.

به یک بردار *امکان‌پذیر* گفته می‌شود اگر در همه‌ی محدودیت‌های برنامه‌ریزی صدق کند. منظور از حل یک برنامه‌ریزی خطی، یافتن برداری امکان‌پذیر است که مقدار تابع هدف را کمینه کند. الگوریتم‌های کارایی برای حل کردن برنامه‌ریزی خطی وجود دارند. یکی از بهترین الگوریتم‌های چندجمله‌ای که برای حل کردن برنامه‌ریزی خطی ارائه شده، الگوریتم interior-point است که در سال ۱۹۸۴ توسط کارمارکار پیشنهاد شده‌است [۱]. زمان اجرای این الگوریتم از $O(n^{۳/۵}L)$ است که n تعداد متغیرهای برنامه‌ریزی خطی است و L تعداد بیت‌هایی است که برای نمایش متغیرها نیاز است.

[۱۸] Objective Function

۱-۵ برنامه‌ریزی صحیح

برنامه‌ریزی صحیح[۱۹] یک برنامه‌ریزی خطی است که در آن، دامنه‌ی متغیرها محدود به اعداد صحیح است. هرچند برنامه‌ریزی خطی در زمان چندجمله‌ای قابل حل است، ولی در مورد برنامه‌ریزی صحیح می‌توان نشان داد که مسئله‌ی حل کردن برنامه‌ریزی صحیح یک مسئله‌ی NP-سخت است. گفتنی است یک حالت خاص از مسئله‌ی برنامه‌ریزی صحیح که در آن دامنه‌ی متغیرها تنها مجموعه‌ی دو عضوی $\{۰,۱\}$ است، یکی از ۲۱ مسئله‌ای است که کارپ به عنوان مسئله‌های NP-سخت معرفی کرده‌است [۲].

۱-۶ الگوریتم تقریبی بر پایه‌ی برنامه‌ریزی خطی

برای بسیاری از مسئله‌های کمینه‌سازی هزینه، می‌توان یک برنامه‌ریزی صحیح معادل نوشت که دامنه‌ی متغیرهای آن $\{۰,۱\}$ است، ولی همان گونه که اشاره شد، با فرض $NP \neq P$، الگوریتم کارآیی برای حل برنامه‌ریزی صحیح وجود ندارد تا با کمک آن، مسئله‌ی بهینه‌سازی مورد نظر نیز حل شود. در چنین مواردی، یک راهکار رایج برای به دست آوردن یک الگوریتم تقریبی، این است که برنامه‌ریزی صحیح متناظر با مسئله‌ی مورد نظر به یک برنامه‌ریزی خطی با همان محدودیت‌ها تبدیل گردد و حل شود. بدیهی است که در برنامه‌ریزی خطی متناظر، دامنه‌ی متغیرها به جای مجموعه‌ی دوعضوی $\{۰,۱\}$، مجموعه‌ی همه‌ی اعداد گویای بازه‌ی $[۰,۱]$ خواهد بود. فرض کنید جواب این برنامه‌ریزی خطی را بردار OPT_{LP} بنامیم. یک ایده برای به دست آوردن الگوریتمی تقریبی آن است که پس از به دست آوردن OPT_{LP}، با گرد کردن مؤلفه‌های بردار OPT_{LP} به یکی از دو روش زیر، پاسخی تقریبی برای برنامه‌ریزی صحیح مورد نظر به دست آید.

- **گرد کردن قطعی**

در این روش، یک عدد $k > ۱$ انتخاب می‌شود. سپس در برنامه‌ریزی صحیح، مقدار هر متغیر که مقدار آن در OPT_{LP} کم‌تر از $\frac{۱}{k}$ است، ۰ و مقدار سایر متغیرها ۱ در نظر گرفته می‌شود. آن‌چه در این جا اهمیت دارد، انتخاب مقدار مناسبی برای k است، به گونه‌ای که بردار حاصل

[۱۹] Integer Programming (IP)

از گرد کردن، یک بردار امکان‌پذیر برای برنامه‌ریزی صحیح باشد. از آن‌جایی که در بردار جدید به دست آمده، مقدار هر متغیر حداکثر k برابر شده‌است، مقدار تابع هزینه نیز حداکثر k برابر خواهد شد و به این ترتیب، یک الگوریتم با ضریب تقریب k خواهیم داشت.

- **گرد کردن تصادفی**

در این روش، مقدار هر متغیر، نه به صورت قطعی بلکه با احتمالی به ۱ گرد می‌شود که این احتمال، خود تابعی است از مقدار همان متغیر در OPT_{LP}. بدیهی است که در این صورت، باید پذیرفت بردار جدید ممکن است امکان‌پذیر نباشد. در تحلیل یک الگوریتم تقریبی که بر پایه‌ی گرد کردن تصادفی طراحی شده، باید هم احتمال به دست آمدن برداری امکان‌پذیر بررسی شود و هم امید ریاضی مقدار تابع هزینه.

فصل ۲

معرفی مسئله

۲-۱ تعریف دقیق مسئله

در این نوشتار، مسئله‌ی راه فرار مستطیل‌ها[۱] را بررسی خواهیم کرد. این مسئله را می‌توان این گونه تعریف کرد:

مسئله ۱ (راه فرار مستطیل‌ها) *یک ناحیه‌ی مستطیلی با اضلاع موازی محورهای مختصات به نام قاب داده شده‌است که درون آن n مستطیل با اضلاع موازی محورها قرار گرفته‌اند. هدف فراری دادن این مستطیل‌ها - هر یک به یکی از چهار جهت اصلی - است، به گونه‌ای که بیشینه چگالی[۲] نقاط قاب کمینه شود. چگالی یک نقطه از قاب، تعداد مستطیل‌هایی تعریف می‌شود که روی آن نقطه قرار گرفته‌اند یا در فرارشان از آن نقطه عبور می‌کنند.*

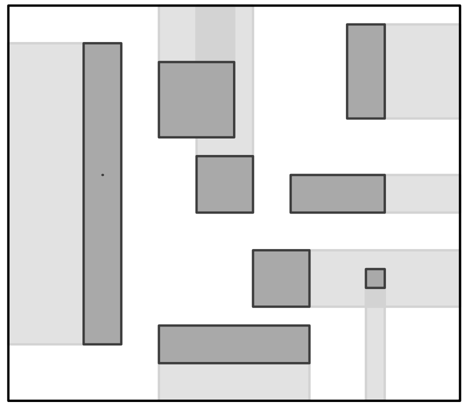
در این مقاله، برای سادگی، مسئله‌ی تصمیم‌گیری زیر را به ازای هر عدد طبیعی k تعریف می‌کنیم:

مسئله ۲ *یک نمونه از مسئله‌ی راه فرار مستطیل‌ها داده شده‌است. آیا می‌توان مستطیل‌ها را به گونه‌ای فراری داد که چگالی هیچ نقطه‌ای از قاب بیش‌تر از k نشود؟*

---

[۱] Rectange Escape Problem (REP)

[۲] Density



شکل ۲-۱: یک نمونه از مسئله‌ی راه فرار مستطیل‌ها.

بدیهی است که به ازای یک ورودی از مسئله‌ی راه فرار مستطیل‌ها، پاسخ مسئله‌ی ۱ برابر است با کم‌ترین عدد k که پاسخ مسئله‌ی ۲ به ازای آن، *بلی* شود.

مسئله‌ی راه فرار مستطیل‌ها که در [۳] تعریف شده‌است، در مسیردهی روی بردهای دیجیتال[۳] مورد نیاز است. تراشه‌های مستطیلی شکلی را در نظر بگیرید که روی یک برد قرار گرفته‌اند، به گونه‌ای که اضلاع تراشه‌ها موازی اضلاع برد است. می‌خواهیم هر یک از تراشه‌ها را در یکی از چهار جهت اصلی از طریق یک گذرگاه[۴] (مطابق شکل ۲-۱) به دیواره‌ی برد متصل کنیم. هدف کمینه کردن بیش‌ترین تعداد گذرگاهی است که در یک نقطه برخورد می‌کنند تا تعداد لایه‌های لازم برای اتصال تراشه‌ها به دیواره‌های قاب کمینه شود.

مسئله‌ی کمینه‌سازی تعداد لایه‌ها، پیش‌تر نیز مورد بررسی قرار گرفته‌اند [۴، ۵، ۶، ۷، ۸، ۹، ۱۰، ۱۱، ۱۲]. گفتنی است که [۴] برای مسئله‌ی زیر الگوریتمی با زمان اجرای $O(n^6)$ ارائه کرده‌است:

مسئله ۳ *بر روی یک قاب مستطیلی، n مستطیل با اضلاع موازی اضلاع قاب داده شده‌اند، به گونه‌ای که از هر مستطیل، حداقل یک ضلع روی مرز قاب قرار گرفته‌است. بیش‌ترین تعداد مستطیل از میان این n مستطیل را بیابید که هم‌پوشانی نداشته باشند.*

[۳] Printed Circuit Board (PCB) Routing

[۴] Bus

به سادگی می‌توان دید که با کمک مسئله‌ی ۳ می‌توان مسئله‌ی ۲ به ازای $k = ۱$ را حل کرد. کافی است برای یک نمونه از مسئله‌ی راه فرار مستطیل‌ها، همه‌ی n مستطیل داده‌شده را به هر چهار جهت تا مرز قاب گسترش دهیم. به این ترتیب، $۴n$ مستطیل خواهیم داشت که حداقل یک ضلع از هر کدام از آن‌ها روی مرز قاب قرار گرفته‌است. اکنون کافی است از میان این $۴n$ مستطیل، بیش‌ترین تعداد مستطیل بدون هم‌پوشانی را بیابیم. این تعداد برابر n است اگر و تنها اگر مستطیل‌ها بتوانند با بیشینه چگالی ۱ فرار کنند.

۲-۲ کارهای پیشین

درباره‌ی مسئله‌ی راه فرار مستطیل‌ها پیش از این ثابت شده‌است:

- مسئله‌ی راه فرار مستطیل‌ها یک مسئله‌ی NP-سخت است. به طور دقیق‌تر، مسئله‌ی ۲ به ازای $k = ۳$ در کلاس پیچیدگی NP-کامل قرار دارد [۳].
- هرچند مسئله‌ی راه فرار مستطیل‌ها در حالت کلی NP-سخت است، ولی همان گونه که توضیح داده‌شد، برای مسئله‌ی ۲ به ازای $k = ۱$، یک الگوریتم با زمان اجرای $O(n^۶)$ در [۴] ارائه شده‌است.
- در [۳]، الگوریتمی تقریبی با زمان اجرای چندجمله‌ای و ضریب تقریب ۴ برای مسئله‌ی ۱ ارائه شده‌است.

۲-۳ نتایج ما

نتایجی که در این نوشتار به دست آورده‌ایم، به شرح زیر است:

- مسئله‌ی ۲ به ازای $k = ۲$ در کلاس پیچیدگی NP-کامل قرار دارد. گفتنی است که این نتیجه، حتی برای حالت خاص‌تر مسئله که در آن هیچ دو مستطیلی از n مستطیل ورودی هم‌پوشانی ندارند هم برقرار است.
- با فرض $P \neq NP$، برای مسئله‌ی ۱ نمی‌توان الگوریتمی تقریبی با ضریب تقریب بهتر از $\frac{۳}{۲}$ یافت.
- مسئله‌ی ۲ وقتی $k = ۱$، در زمان $O(n^۴)$ قابل حل است.
- به ازای هر عدد ثابت $\epsilon > ۰$، در حالتی که جواب مسئله‌ی ۱ به اندازه‌ی کافی بزرگ باشد، الگوریتمی احتمالی با ضریب تقریب $۱ + \epsilon$ برای آن وجود دارد.

فصل ۳

سختی مسئله

در این بخش، ابتدا سختی مسئله‌ی ۲ به ازای $k = ۲$ را بررسی می‌کنیم و نشان می‌دهیم که این مسئله در کلاس پیچیدگی NP-کامل قرار دارد. این نتیجه را برای حالت خاص‌تری از مسئله که در آن هیچ دو مستطیلی از n مستطیل ورودی هم‌پوشانی ندارند، ثابت می‌کنیم. سپس نتیجه می‌گیریم با فرض $NP \neq P$، برای مسئله‌ی ۱ الگوریتمی تقریبی با ضریب تقریب بهتر از $\frac{۳}{۲}$ نمی‌توان یافت.

۳-۱ سختی مسئله برای چگالی ۲

همان گونه که اشاره شد، سختی مسئله‌ی ۲ برای چگالی ۳ پیش‌تر در [۳] بررسی شده‌است. در این جا، سختی این مسئله را به ازای $k = ۲$ نشان خواهیم داد. پیش از بررسی سختی مسئله‌ی ۲، ابتدا مسئله‌ی مشهور ۳-ارضاپذیری [۱] را معرفی می‌کنیم:

مسئله ۴ (۳-ارضاپذیری) *برای مجموعه‌ی $X = \{x_۱, \dots, x_n\}$ از متغیرهایی با دامنه‌ی $\{۰, ۱\}$، m عبارت[۲] به شکل $C_i = l_{i,۱} \vee l_{i,۲} \vee l_{i,۳}$ داده‌شده‌اند که در آن، هر $l_{i,\lambda}$ یکی از n متغیر عضو X یا نقیض یکی آن‌هاست. آیا می‌توان متغیرهای مجموعه‌ی X را به گونه‌ای مقداردهی کرد که مقدار*

[۱] ۳-SAT

[۲] Clause

همه‌ی این m عبارت منطقی، برابر ۱ شود؟

مسئله‌ی ۴ یکی از مشهورترین مسئله‌های NP-کامل شناخته‌شده است [۲]. ما در این بخش با فرض NP-کامل بودن مسئله‌ی ۴، قضیه‌ی زیر را ثابت می‌کنیم:

قضیه ۳ *برای $k = 2$، مسئله‌ی ۲ حتی با این محدودیت که هیچ دو مستطیلی هم‌پوشانی ندارند، یک مسئله‌ی NP-کامل است.*

اثبات. این که مسئله‌ی مورد نظر در کلاس پیچیدگی NP قرار می‌گیرد، بدیهی است. پس برای نشان دادن NP-کامل بودن این مسئله برای $k = 2$، کافی است NP-سخت بودن آن را ثابت کنیم. این کار را با کاهش[۳] از مسئله‌ی ۴ انجام می‌دهیم. به بیان ساده‌تر، ثابت می‌کنیم که اگر مسئله‌ی ۲ به ازای $k = 2$ در حالتی که هم‌پوشانی وجود ندارد، الگوریتمی با زمان اجرای چندجمله‌ای داشته‌باشد، آن‌گاه مسئله‌ی ۴ نیز در زمان چندجمله‌ای قابل حل است. به این منظور، برای یک نمونه از مسئله‌ی ۳-ارضاپذیری ، یک نمونه از مسئله‌ی راه فرار مستطیل‌ها به شکل زیر می‌سازیم:

یک قاب مستطیلی در نظر می‌گیریم و آن را به صورت مجازی به چهار ناحیه تقسیم می‌کنیم. این ناحیه‌ها را مطابق شکل ۳-۱، ناحیه‌ی *بالا*[۴]، ناحیه‌ی چپ[۵]، ناحیه‌ی *متغیرها*[۶] و ناحیه‌ی *عبارت‌ها*[۷] می‌نامیم.

- به ازای هر متغیر x_j، دو مستطیل با نام‌های v_j و $\bar{v_j}$ متناظر با x_j و $\bar{x_j}$ همان گونه که در شکل ۳-۱ نشان داده‌شده، در ناحیه‌ی متغیرها به صورت افقی کنار هم قرار می‌دهیم. این مستطیل‌های متناظر با متغیرها باید به گونه‌ای قرار بگیرند که یک خط عمودی یا افقی، دو مستطیل مربوط به دو متغیر مختلف را قطع نکند. فرض کنید این مستطیل‌ها را مستطیل‌های *نوع یک* بنامیم. علاوه بر مستطیل‌های نوع یک، یک مستطیل بلند نیز به صورت عمودی در سمت راست ناحیه‌ی متغیرها قرار می‌گیرد. شکل ۳-۱ را ببینید.

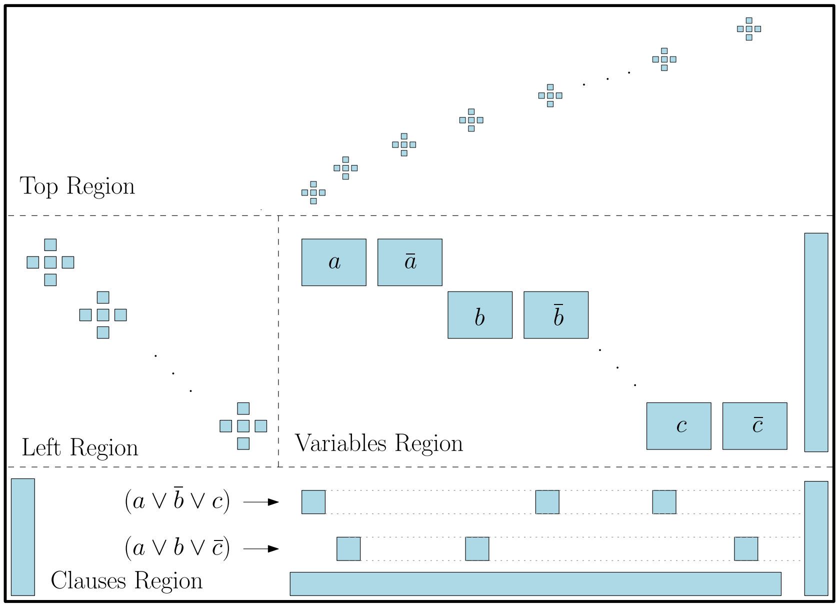
[۳] Reduction

[۴] Top Region

[۵] Left Region

[۶] Variable Region

[۷] Clause Region



شکل ۳-۱: کاهش از مسئله‌ی ۳-ارضاپذیری.

- برای هر عبارت $C_i = l_{i,۱} \vee l_{i,۲} \vee l_{i,۳}$، سه مستطیل در بخش عبارت‌ها قرار می‌دهیم. این سه مستطیل روی یک خط افقی به گونه‌ای قرار می‌گیرند که زیر مستطیل‌های نوع یک متناظر با $l_{i,i}$، $l_{i,۲}$ و $l_{i,۳}$ باشند. همه‌ی این مستطیل‌ها را نیز مستطیل‌های *نوع دو* می‌نامیم. همان گونه که در مثال شکل ۳-۱ نشان داده‌شده، یک خط عموی یا افقی نباید هیچ دو مستطیلی را که وابسته به دو عبارت مختلف هستند، قطع نماید.

  علاوه بر این مستطیل‌های نوع دو، دو مستطیل بلند به صورت عمودی در سمت چپ و سمت راست این ناحیه و هم‌چنین یک مستطیل بلند افقی در پایین آن قرار می‌دهیم.

- برای هر متغیر، در سمت چپ مستطیل‌های نوع یک متناظر با آن، ۵ مربع کوچک که شکلی صلیبی ساخته‌اند، در ناحیه‌ی چپ قرار می‌گیرند. این شکل‌های صلیبی را *سد* می‌نامیم. دقت کنید که ۵ مربع یک سد به هر روشی که فرار کنند، چگالی نقطه‌ای روی یکی از آن‌ها حداقل ۲ خواهد شد. سدهای ناحیه‌ی چپ به گونه‌ای قرار می‌گیرند که با یک خط افقی یا عمودی نتوان دو سد مختلف را قطع کرد.

- بالای هر مستطیل نوع دو، یک سد در ناحیه‌ی بالا قرار می‌گیرد. سپس اگر بالای یک مستطیل

نوع یک هیچ سدی قرار نگرفت، برای آن مستطیل نیز یک سد در ناحیه‌ی بالا قرار می‌دهیم. سدهای ناحیه‌ی بالا را نیز به شکلی قرار می‌دهیم که یک خط عمودی یا افقی نتواند دو سد مختلف را قطع کند.

اکنون فرض کنید در نمونه‌ی ساخته‌شده، مستطیل‌ها بتوانند به گونه‌ای فرار کنند که چگالی هیچ نقطه‌ای از قاب بیش‌تر از ۲ نشود. می‌خواهیم ثابت کنیم یک مقداردهی برای متغیرهای $\{x_۱, \ldots, x_n\}$ وجود دارد که به ازای آن مقداردهی، همه‌ی عبارت‌ها مقدار ۱ می‌گیرند.

با توجه به مکان قرارگیری سدها در ناحیه‌ی چپ و ناحیه‌ی بالا، هیچ مستطیل نوع یکی نمی‌تواند به چپ یا بالا فرار کرده‌باشد، پس هر کدام از مستطیل‌های نوع یک به راست یا پایین فرار کرده‌اند. از طرفی، با توجه به قرارگیری مستطیل بلندی در سمت راست ناحیه‌ی متغیرها، دو مستطیل نوع یکی که متناظر با متغیر یکسانی هستند، نمی‌توانند هر دو به راست فرار کرده‌باشند؛ چراکه در این صورت چگالی نقطه‌ای روی مستطیل بلند سمت راست ناحیه‌ی متغیرها، بیش‌تر از ۲ می‌شد. برای هر متغیر x_i، اگر v_i به سمت راست فرار کرده‌بود، مقدار x_i را برابر ۱ قرار می‌دهیم و اگر $\bar{v_i}$ به سمت راست فرار کرده‌بود، مقدار $\bar{x_i}$ را برابر با ۱ در نظر می‌گیریم. اگر هم هیچ یک از این دو مستطیل به سمت راست فرار نکرده‌بودند (در واقع فرار هر دو به سمت پایین بود)، مقدار x_i را ۱ و در نتیجه مقدار $\bar{x_i}$ را ۰ می‌گیریم. ادعا می‌کنیم که با این مقداردهی، همه‌ی عبارت‌ها برابر ۱ خواهند شد.

به خاطر سدهای ناحیه‌ی بالا، هیچ یک از مستطیل‌های نوع دو نمی‌تواند به سمت بالا فرار کند. هم‌چنین با توجه به قرارگیری دو مستطیل بلند سمت چپ و راست ناحیه‌ی عبارت‌ها، برای هر عبارت $C_i = l_{i,۱} \vee l_{i,۲} \vee l_{i,۳}$، حداکثر یکی از سه مستطیل نوع دو وابسته به این عبارت می‌تواند به چپ فرار کند و حداکثر هم یکی به راست. بنابراین، حداقل یکی از این سه مستطیل نوع دو به سمت پایین فرار کرده‌است. فرض کنید که مستطیل متناظر با $l_{i,\lambda}$ به پایین فرار کرده‌باشد ($\lambda \in \{۱,۲,۳\}$). مستطیل نوع یکی که بالای این مستطیل قرار گرفته‌است، باید به سمت راست فرار کرده‌باشد، چون اگر به پایین فرار کرده‌باشد، چگالی نقطه‌ای روی مستطیل بلند پایین ناحیه‌ی عبارت‌ها بیش از ۲ خواهد شد. پس با توجه به نحوه‌ی مقداردهی متغیرها، مقدار $l_{i,\lambda}$ و در نتیجه مقدار عبارت C_i برابر ۱ است. به این ترتیب ثابت شد که با مقداردهی ارائه‌شده، مقدار همه‌ی عبارت‌ها برابر ۱ خواهد شد.

در سوی دیگر، باید ثابت کنیم که اگر برای نمونه‌ی داده‌شده از مسئله‌ی ۳-ارضاپذیری ، یک

مقداردهی وجود داشته‌باشد که همه‌ی عبارت‌ها را ۱ کند، آن‌گاه مستطیل‌ها می‌توانند به گونه‌ای فرار کنند که بیشینه چگالی نقاط قاب ۲ باشد. به این منظور، برای هر متغیر x_i، اگر $x_i = ۱$، آن‌گاه جهت فرار v_i را سمت راست و جهت فرار $\bar{v_i}$ را پایین در نظر در نظر می‌گیریم. در غیر این صورت، v_i را به سمت پایین و $\bar{v_i}$ را به سمت راست فراری می‌دهیم.

برای هر عبارت $C_i = l_{i,۱} \vee l_{i,۲} \vee l_{i,۳}$، حداقل یک $\lambda \in \{۱, ۲, ۳\}$ وجود دارد که مقدار $l_{i,\lambda}$ برابر ۱ باشد. برای فرار مستطیل نوع دو متناظر با $l_{i,\lambda}$ جهت پایین در نظر گرفته می‌شود و از بین دو مستطیل نوع دو دیگری که وابسته به همین عبارت هستند، مستطیل سمت چپ به سمت چپ و مستطیل دیگر به راست فرار می‌کند.

هم‌چنین مستطیل بلند سمت راست ناحیه‌ی متغیرها به بالا، مستطیل‌های بلند سمت چپ و سمت راست ناحیه‌ی عبارت‌ها هر دو به پایین و مستطیلی که در پایین این ناحیه قرار گرفته‌است به سمت راست فراری داده می‌شود. در مورد سدها هم کافی است از هر سد، مربع‌های وسطی و پایینی به سمت چپ فرار کنند و سه مستطیل دیگر به سمت بالا. به این ترتیب، همه‌ی مستطیل‌ها می‌توانند به گونه‌ای فرار کنند که بیشینه چگالی نقاط قاب برابر ۲ باشد. گفتنی است که با این شیوه‌ی فرار، چگالی هیچ نقطه‌ای روی اضلاع قاب بیش‌تر از ۱ نخواهد شد.

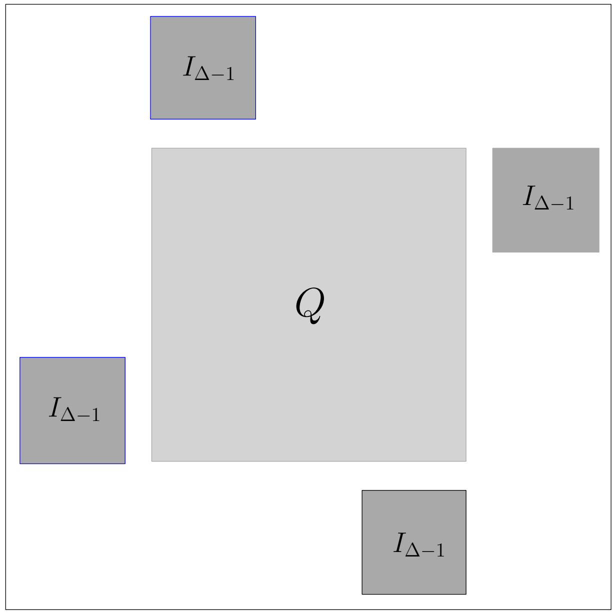
بنابر آن‌چه گفته‌شد، در صورت حل مسئله‌ی ۲ به ازای $k = ۲$ در زمان چندجمله‌ای، مسئله‌ی ۳-ارضاپذیری نیز در زمان چندجمله‌ای قابل حل خواهد بود. پس مسئله‌ی ۲ برای $k = ۲$ در کلاس پیچیدگی NP-کامل قرار دارد.

□

۳-۲ سختی مسئله برای چگالی بیش‌تر از ۲

پیش‌تر، سختی مسئله‌ی ۲ به ازای $k = ۲$ بررسی شد. در این‌جا، نتیجه‌ی مشابهی را با استفاده از استقرا به ازای هر عدد طبیعی ثابت $k \geq ۲$ به دست می‌آوریم.

قضیه ۴ *برای هر عدد طبیعی $k \geq ۲$، مسئله‌ی ۲ حتی برای حالتی که هیچ دو مستطیلی هم‌پوشانی نداشته‌باشند، در کلاس NP-کامل قرار دارد.*



شکل ۳-۲: ساختن I_{Δ} با استفاده از چهار نمونه از $I_{\Delta-1}$ برای مسئله‌ی ۲.

اثبات. پایه‌ی استقرا معادل قضیه‌ی ۳ است که ثابت شد. اکنون فرض کنید که متناظر با نمونه‌ی داده‌شده‌ای از مسئله‌ی ۴، $I_{\Delta-1}$ نمونه‌ای از مسئله‌ی ۲ به ازای $k = \Delta - 1$ باشد ($\Delta > 2$). برای ساختن I_{Δ} همان گونه که در شکل ۳-۲ نشان داده‌شده، یک مربع بزرگ Q در وسط می‌گذاریم و چهار نمونه از $I_{\Delta-1}$ را در اطراف آن به گونه‌ای قرار می‌دهیم که یک خط عمودی یا افقی نتواند قاب هیچ دوتایی از آن‌ها را قطع کند.

ادعا می‌کنیم که پاسخ مسئله‌ی ۲ به ازای $k = \Delta$ برای نمونه‌ی I_{Δ} *بلی* است اگر و تنها اگر جواب همین مسئله به ازای $k = \Delta - 1$ برای نمونه‌ی $I_{\Delta-1}$ *بلی* باشد. ابتدا فرض کنید مستطیل‌های I_{Δ} به شکلی فرار کرده‌اند که چگالی هیچ نقطه‌ای از قاب بیش‌تر از Δ نیست. با توجه به این که Q به یکی از چهار جهت فرار کرده‌است و در نتیجه از روی یکی از چهار نمونه‌ی $I_{\Delta-1}$ عبور کرده‌است، می‌توان نتیجه گرفت که اگر $I_{\Delta-1}$ به تنهایی در نظر گرفته‌شود، مستطیل‌ها می‌توانند به گونه‌ای فرار کنند که بیشینه چگالی، $\Delta - 1$ شود. در دیگر سوی، باید ثابت کنیم که اگر مستطیل‌ها در $I_{\Delta-1}$ بتوانند با بیشینه چگالی $\Delta - 1$ فرار کنند، آن‌گاه در I_{Δ}، مستطیل‌ها می‌توانند به گونه‌ای فرار کنند که بیشینه چگالی Δ شود. دقت کنید که مستطیل‌های چهار نمونه‌ی $I_{\Delta-1}$ در I_{Δ} باید به گونه‌ای فرار

کنند که چگالی هیچ نقطه‌ای روی مکان اولیه‌ی Q از Δ بیش‌تر نشود. مشاهده‌ی زیر اثبات قضیه‌ی ۴ را کامل می‌کند.

مشاهده ۱ *به ازای هر $\Delta \geq 2$، اگر مستطیل‌های I_Δ بتوانند به گونه‌ای فرار کنند که چگالی هیچ نقطه‌ای بیش از Δ نشود، آن‌گاه این مستطیل‌ها را می‌توان به گونه‌ای فراری داد که*

- *چگالی نقاط ضلع بالایی قاب کم‌تر از Δ (حداکثر $\Delta - 1$) باشد.*
- *چگالی هر نقطه‌ای روی سه ضلع دیگر قاب حداکثر ۱ باشد.*

در زیربخش قبلی، این نتیجه برای $\Delta = 2$ به دست آمده‌بود. از طرفی دیگر، فرض کنید مستطیل‌های هر چهار $I_{\Delta-1}$ در I_Δ به گونه‌ای فرار کرده‌باشند که برای هر یک از این چهار نمونه، بیشینه چگالی روی ضلع بالایی قاب $\Delta - 2$ باشد و چگالی نقاط سایر اضلاع قاب هم حداکثر ۱. به سادگی می‌توان دید که چگالی هیچ نقطه‌ای از مکان اولیه‌ی Q بیش از Δ نیست. هم‌چنین اگر خود Q به سمت بالا فرار کند، آن‌گاه محدودیت‌های گفته‌شده در مشاهده برای چگالی نقاط روی قاب I_Δ رعایت شده‌است.

□

۳-۳ تقریب‌ناپذیری

با استفاده از قضیه‌ی ۳، می‌توان قضیه‌ی زیر را ثابت کرد:

قضیه ۵ (تقریب‌ناپذیری) *برای مسئله‌ی ۱ الگوریتمی تقریبی با زمان چندجمله‌ای و ضریب تقریب بهتر از $\frac{3}{2}$ نمی‌توان یافت مگر آن‌که $P = NP$.*

اثبات. گفتنی است که این نتیجه‌ی تقریب‌ناپذیری را نیز برای حالت بدون هم‌پوشانی می‌توان ثابت کرد. اگر الگوریتمی با ضریب تقریب کم‌تر از $\alpha < \frac{3}{2}$ وجود داشته‌باشد، آن‌گاه به ازای یک ورودی از مسئله‌ی راه فرار مستطیل‌ها، در زمان چندجمله‌ای می‌توان دریافت که آیا مستطیل‌ها می‌توانند به گونه‌ای فرار کنند که چگالی هیچ نقطه‌ای از قاب بیش از ۲ نشود: مستطیل‌ها را به عنوان ورودی به الگوریتم تقریبی دارای ضریب تقریب α می‌دهیم و پاسخ به دست آمده را با عدد ۳ مقایسه می‌نماییم.

- اگر پاسخ مسئله‌ی ۱ حداکثر ۲ باشد، آن‌گاه الگوریتم تقریبی باید پاسخی کم‌تر $۲ \times \frac{۳}{۲} = ۳$ بیابد.
- اگر پاسخ مسئله‌ی ۱ حداقل ۳ باشد، پاسخی که الگوریتم تقریبی مورد نظر به دست می‌آورد نیز حداقل ۳ خواهد بود.

$\square$

فصل ۴

الگوریتم دقیق برای چگالی واحد

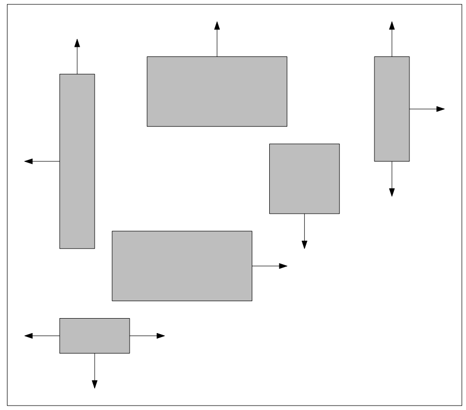
در این بخش یک الگوریتم با زمان اجرای $O(n^۴)$ برای مسئله‌ی ۲ به ازای $k = ۱$ ارائه می‌کنیم. به بیان دیگر، یک نمونه از مسئله‌ی راه فرار مستطیل‌ها داده شده‌است که در آن مکان اولیه‌ی هیچ دو مستطیلی هم‌پوشانی ندارند و هدف فراری دادن مستطیل‌هاست با این شرط که چگالی هیچ نقطه‌ای از قاب بیش‌تر از یک نشود. همان‌گونه که پیش‌تر گفته‌شد، زمان اجرای بهترین الگوریتم قبلی برای این مسئله، الگوریتمی با زمان $O(n^۶)$ بوده که در [۴] ارائه شده‌است. در این بخش، با کمک برنامه‌نویسی پویا[۱]، الگوریتمی با زمان اجرای $O(n^۴)$ برای این مسئله به دست خواهیم‌آورد. به این منظور، ابتدا این مسئله‌ی بهینه‌سازی را که آن را بیشینه فرار مجزا[۲] می‌نامیم، در نظر بگیرید:

مسئله ۵ (بیشینه فرار مجزا) *یک نمونه از مسئله‌ی راه فرار مستطیل‌ها داده شده‌است که در آن مکان هیچ دو مستطیلی هم‌پوشانی ندارند. بیش‌ترین تعداد مستطیلی را بیابید که با چگالی یک می‌توانند فرار کنند.*

لازم به ذکر است که در مسئله‌ی ۵، مکان اولیه‌ی مستطیل‌های فرار نکرده هم اهمیت دارد: یک مستطیل در مسیر فرار خود نمی‌تواند با مکان اولیه‌ی هیچ مستطیل دیگری برخورد کند، حتی

[۱] Dynamic Programming

[۲] Maximum Disjoint Escaping



شکل ۴-۱: جهت‌های آزاد برای هر یک از مستطیل‌ها در مسئله‌ی ۲ به ازای $k = 1$.

مستطیل‌های فرار نکرده.

مستطیل‌های $R_1, \ldots, R_n$ را در نظر بگیرید به گونه‌ای که اضلاع این مستطیل‌ها موازی محورهای مختصات هستند و هیچ دو مستطیلی هم‌پوشانی ندارند. بدیهی است که قاب را می‌توان هر ناحیه‌ی مستطیلی دلخواهی با مرزهای موازی محورها در نظر گرفت که همه‌ی این n مستطیل را در بر بگیرد. فرض کنید که این n مستطیل بر حسب ضلع پایینی خود مرتب شده‌اند، به گونه ای که ضلع پایینی R_i از ضلع پایینی R_{i+1} بالاتر نیست ($1 \leq i < n$).

برای مستطیل R_i و $d \in \{left, right, up, down\}$، جهت d را برای مستطیل R_i *آزاد* می‌نامیم، اگر R_i در صورت فرار در جهت d از روی مکان اولیه‌ی هیچ مستطیلی عبور نکند. بنا بر تعریف، آزاد بودن یک جهت برای یک مستطیل وابسته به جهت فرار هیچ مستطیلی نیست. شکل ۴-۱ را ببینید. برای هر مستطیل R_i، جهت‌های آزاد این مستطیل را می‌توان به سادگی در زمان $O(n)$ به دست آورد، پس با یک پیش‌پردازش[3] در زمان $O(n^2)$ جهت‌های آزاد همه‌ی مستطیل‌های داده‌شده را می‌توان یافت. گذشته از این، مجموعه‌ی $\{v_1, \ldots, v_k\}$ را مجموعه‌ی همه‌ی خط‌های عمودی در نظر می‌گیریم که ضلع‌های عمودی مستطیل‌ها بر روی آن‌ها قرار گرفته‌اند. فرض کنید این خط‌ها از چپ به راست مرتب شده‌اند. این خط‌ها را نیز در پیش‌پردازش با همان زمان $O(n^2)$ می‌توان به دست آورد.[4]

[3] Preprocess

[4] همه‌ی این پیش‌پردازش‌ها را در زمان $O(n \log n)$ هم می‌توان انجام داد.

برای حل مسئله‌ی ۵، ابتدا دو زیر مسئله‌ی ساده‌ی زیر را در نظر می‌گیریم:

- $One\text{-}Direction(i, l, r)$

  بیش‌ترین تعداد مستطیل را از میان مستطیل‌های $R_1, \ldots, R_i$ که بین دو خط عمودی v_l و v_r قرار گرفته‌اند و می‌توانند به سمت بالا فرار کنند.

- $Two\text{-}Direction(i, l, r)$

  بیش‌ترین تعداد مستطیل را از میان مستطیل‌های $R_1, \ldots, R_i$ که بین دو خط عمودی v_l و v_r قرار گرفته‌اند و می‌توانند به سمت بالا یا پایین فرار کنند.

این دو زیر مسئله به سادگی با الگوریتم‌های حریصانه[5] قابل حل هستند. برای یافتن مقدار $One\text{-}Direction(i, l, r)$ کافی است از میان مستطیل‌های $R_1, \ldots, R_i$ که بین دو خط v_l و v_r قرار گرفته‌اند، تعداد مستطیل‌هایی را بیابیم که جهت بالا برای آن‌ها آزاد است. هم‌چنین برای $Two\text{-}Direction(i, l, r)$ باید تعداد مستطیل‌هایی در بین مستطیل‌های گفته‌شده را یافت که برای آن‌ها جهت بالا یا پایین، آزاد باشد. توجه کنید که اگر دو مستطیل در جهت عمودی (بالا یا پایین) فرار کنند به گونه‌ای که جهت فرارشان آزاد باشد، آن‌گاه مسیر فرارشان برخورد نخواهد داشت.

بنا به آن‌چه گفته‌شد، مقادیر $One\text{-}Direction$ و $Two\text{-}Direction$ را می‌توان در زمان $O(n^4)$ برای همه‌ی سه‌تایی‌های (i, l, r)به دست آورد[6] و در آرایه‌هایی ذخیره نمود.

اکنون زیرمسئله‌های زیر را در نظر بگیرید:

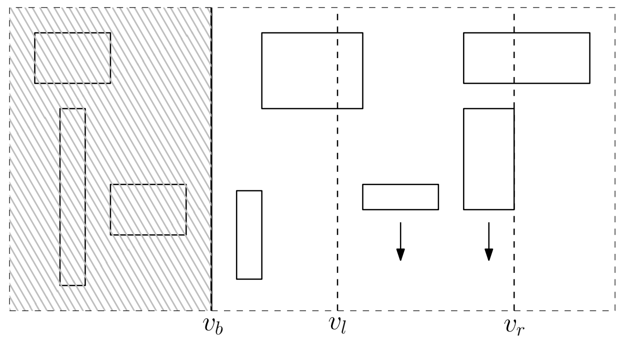
مسئله ۶ (فرار به سه جهت) *زیرمسئله‌ی $No\text{-}Left\text{-}Escape(i, b, l, r)$ بیش‌ترین تعداد مستطیل در میان مستطیل‌های $R_1, \ldots, R_i$ تعریف می‌شود که با شرایط زیر می‌توانند فرار کنند:*

- *هیچ مستطیلی نمی‌تواند به سمت چپ فرار کند.*
- *فقط مستطیل‌هایی که سمت راست خط عمودی v_b قرار دارند، می‌توانند فرار کنند.*
- *فقط مستطیل‌هایی که بین خط‌های v_l و v_r قرار دارند، می‌توانند به سمت پایین فرار کنند.*

---

[5] Greedy

[6] این مقادیر با کمک برنامه‌نویسی پویا در زمان $O(n^3)$ نیز قابل محاسبه هستند.



شکل ۴-۲: یک نمونه از مسئله‌ی فرار به سه جهت.

شکل ۴-۲ را ببینید.

زیرمسئله‌ی $No\text{-}Right\text{-}Escape(i,b,l,r)$ نیز به طریق مشابه قابل تعریف است، با این تفاوت که در $No\text{-}Right\text{-}Escape$ ، هیچ مستطیلی به سمت راست نمی‌تواند فرار کند.

برای یافتن مقدار $No\text{-}Left\text{-}Escape(i,b,l,r)$ به صورت بازگشتی، می‌توان همه‌ی گزینه‌های ممکن برای R_i (که پایین‌ترین مستطیل در بین مستطیل‌های مورد نظر است) را در نظر گرفت. نخستین گزینه این است که R_i فرار نکند. در این صورت، بیش‌ترین تعداد مستطیلی که با رعایت محدودیت‌های مسئله می‌توانند فرار کنند، برابر است با $No\text{-}Left\text{-}Escape(i-۱,b,l,r)$ که به صورت بازگشتی قابل محاسبه است. سایر گزینه‌ها، فرار R_i به یکی از سه جهت پایین، بالا و راست هستند که به صورت جداگانه در زیر بررسی شده‌اند. در همه‌ی حالت‌ها فرض بر این است که جهت مورد بررسی برای R_i آزاد است و فرار در آن جهت با همه‌ی محدودیت‌های مسئله سازگاری دارد.

در ضمن فرض کنید خطوط v_α و v_β خط‌هایی هستند که ضلع‌های عمودی R_i روی آن‌ها قرار گرفته‌اند و $\alpha < \beta$ (به بیان دیگر، ضلع سمت چپ روی v_α و ضلع سمت راست روی v_β قرار گرفته‌است.)

- فرار به سمت پایین

اگر R_i به سمت پایین فرار کند، هیچ محدودیت جدیدی برای فرار مستطیل‌های $R_۱,\ldots,R_{i-۱}$ ایجاد نخواهد شد. بنابراین، بیش‌ترین تعداد مستطیل از بین این $i-۱$ مستطیل که می‌توانند با رعایت محدودیت‌های مسئله فرار کنند، برابر است با $No\text{-}Left\text{-}Escape(i-۱,b,l,r)$ که

به صورت بازگشتی قابل محاسبه است.

- فرار به سمت *بالا*

با فرار R_i به سمت بالا، یک محدودیت بر فرار $i-1$ مستطیل دیگر افزوده می‌شود: مستطیل‌هایی که در سمت راست v_β قرار ندارند، نمی‌توانند به سمت راست فرار کنند، چراکه در این صورت، با مسیر فرار R_i برخورد خواهند کرد. با توجه به این نکته، مستطیل‌هایی را در که بین v_b و v_β قرار دارند، در نظر بگیرید. بسته به جایگاه قرارگیری v_l و v_r، بیش‌ترین تعداد مستطیل از بین این مستطیل‌ها که می‌توانند فرار کنند، با استفاده از زیر مسئله‌های $One\text{-}Direction$ و $Two\text{-}Direction$ به دست می‌آید. از سوی دیگر، بیش‌ترین تعداد مستطیل از $R_1, \ldots, R_{i-1}$ که سمت راست v_β قرار دارند و می‌توانند فرار کنند، برابر است با

$$No\text{-}Left\text{-}Escape(i-1, max(b, \beta), l, r)$$

که به صورت بازگشتی قابل محاسبه است.

- فرار به سمت *راست*

اگر R_i به سمت راست فرار کند، از بین $i-1$ مستطیل دیگر، مستطیل‌هایی که سمت چپ v_α قرار ندارند، نمی‌توانند به پایین فرار کنند. پس اگر مستطیلی بخواهد به پایین فرار کند، باید نه تنها سمت چپ v_r که سمت چپ v_α نیز قرار داشته‌باشد. به این ترتیب، بیش‌ترین تعداد مستطیل از میان این $i-1$ مستطیل که می‌توانند فرار کنند، برابر است با:

$$No\text{-}Left\text{-}Escape(i-1, b, l, min\{r, \alpha\})$$

زیرمسئله‌ی $No\text{-}Left\text{-}Escape$ مطابق آن‌چه گفته شد، با در نظر گرفتن همه‌ی گزینه‌های ممکن برای R_i به صورت بازگشتی قابل حل است. همین الگوریتم بازگشتی را می‌توان به الگوریتمی مبتنی بر برنامه‌نویسی پویا تبدیل کرد. این ترتیب مقدار $No\text{-}Left\text{-}Escape$ برای همه‌ی چهارتایی‌هایی (i, b, l, r) در زمان $O(n^4)$ به دست می‌آید. هم‌چنین برای زیرمسئله‌ی $No\text{-}Right\text{-}Escape$ نیز با الگوریتم مشابهی می‌توان پاسخ را به ازای همه‌ی چهارتایی‌های (i, b, l, r) به دست آورد. با داشتن این مقادیر، مسئله‌ی کلی زیر را حل خواهیم کرد:

مسئله ۷ *به ازای اعداد $0 \le i \le n$ و $1 \le l, r \le k$، بیش‌ترین تعداد مستطیل از بین مستطیل‌های*

Algorithm ۱ $Max\text{-}Route(i, l, r)$

if $i = 0$ **then**
  Return 0
$ans_n \leftarrow Max\text{-}Route(i-1, l, r)$
$ans_d \leftarrow 0, ans_u \leftarrow 0, ans_l \leftarrow 0, ans_r \leftarrow 0$
$\alpha, \beta \leftarrow$ indices of the vertical lines through the left and the right sides of R_i
if *down* is feasible for R_i **then**
  $ans_d \leftarrow Max\text{-}Route(i-1, l, r) + 1$
if *left* is feasible for R_i **then**
  $ans_l \leftarrow Max\text{-}Route(i-1, \max\{l, \beta\}, r) + 1$
if *right* is feasible for R_i **then**
  $ans_r \leftarrow Max\text{-}Route(i-1, l, \min\{r, \alpha\}) + 1$
if *up* is feasible for R_i **then**
  $ans_u \leftarrow No\text{-}Right\text{-}Escape(i-1, \alpha, l, r) + No\text{-}Left\text{-}Escape(i-1, \beta, l, r) + 1$
Return $\max\{ans_n, ans_d, ans_u, ans_l, ans_r\}$

$R_1, \ldots, R_i$ را بیابید که با این محدودیت می‌توانند فرار کنند: تنها مستطیل‌هایی که سمت راست v_l و سمت چپ v_r قرار دارند، می‌توانند به سمت پایین فرار کنند.

این مسئله که حالت کلی‌تری از مسئله‌ی ۵ است، مشابه دو زیرمسئله‌ی گفته‌شده با در نظر گرفتن همه‌ی گزینه‌های ممکن برای R_i و به صورت بازگشتی قابل حل است. الگوریتم ۱ را ببینید.

الگوریتم بازگشتی ۱ را نیز می‌توان با بهره‌گیری از برنامه‌نویسی پویا بازنویسی کرد و به این ترتیب، مقدار $Max\text{-}Route$ برای همه‌ی سه‌تایی‌های (i, l, r) در زمان $O(n^3)$ به دست می‌آید.

در پایان، برای یافتن پاسخ مسئله‌ی ۵ تنها داشتن مقدار $Max\text{-}Route(n, 1, k)$ کافی است که در آن، ۱ اندیس چپ‌ترین و k اندیس راست‌ترین خط عمودی است. بدیهی است که اگر این مقدار برابر n باشد، به معنی آن است که همه‌ی این n مستطیل می‌توانند در چگالی ۱ فرار کنند. اکنون قضیه‌ی زیر را می‌توان نتیجه گرفت:

قضیه ۶ *به ازای $k = 1$، مسئله‌ی ۲ در زمان $O(n^4)$ قابل حل است که در آن، n تعداد مستطیل‌هاست.*

اثبات. کافی است ابتدا بررسی شود که آیا n مستطیل داده‌شده هم‌پوشانی دارند یا نه. اگر دو مستطیل هم‌پوشانی داشته‌باشند، بدیهی است که مستطیل‌ها نمی‌توانند به گونه‌ای فرار کنند که چگالی همه‌ی نقاط قاب حداکثر ۱ باشد، چراکه برخی ار نقاط قاب را بیش از یک مستطیل پوشانده‌اند. در غیر این صورت، کافی است پاسخ مسئله‌ی ۵ با عدد n مقایسه شود.

$\square$

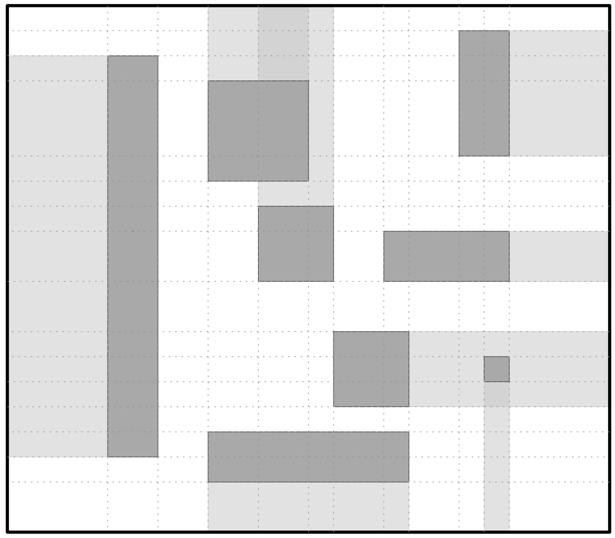
فصل ۵

الگوریتم تقریبی

در این بخش، دو الگوریتم تقریبی برای مسئله‌ی راه فرار مستطیل‌ها را بررسی خواهیم‌کرد. از آن‌جایی که الگوریتم‌های تقریبی این نوشتار مبتنی بر برنامه‌ریزی صحیح و برنامه‌ریزی خطی است، ابتدا یک برنامه‌ریزی صحیح برای مسئله ارائه می‌کنیم.

پیش از ارائه‌ی برنامه‌ریزی صحیح معادل با مسئله‌ی ۱ ، ابتدا فرض کنید همه‌ی اضلاع مستطیل‌ها را از دو طرف گسترش دهیم تا اضلاع قاب را قطع کنند. به این ترتیب، قاب به شکل مشبک در خواهد آمد. شکل ۵-۱ را ببینید. به سادگی می‌توان دید که مستقل از چگونگی فرار مستطیل‌ها، چگالی همه‌ی نقاط درون یک *سلول* از این شبکه با هم برابر است، پس به جای چگالی نقاط قاب، می‌توان چگالی را به یک سلول نسبت داد. از سوی دیگر، بدیهی است که تعداد سلول‌ها از $O(n^{۲})$ خواهد بود.

اکنون می‌توان یک برنامه‌ریزی صحیح نوشت که معادل مسئله‌ی ۱ باشد. مستطیل‌های ورودی را $R_{۱}, \dots, R_n$ در نظر بگیرید. به ازای هر $۱ \leq i \leq n$، چهار متغیر $x_{i,l}$، $x_{i,r}$، $x_{i,u}$ و $x_{i,d}$ با دامنه‌ی $\{۰,۱\}$ تعریف می‌کنیم. منظور از ۱ بودن مقدار $x_{i,l}$، فرار کردن R_i به سمت چپ است. به طریق مشابه، ۱ بودن مقدار $x_{i,r}$، $x_{i,u}$ و $x_{i,d}$ را به ترتیب معادل فرار R_i به راست، بالا و پایین در نظر می‌گیریم. فرض کنید این متغیرها را *متغیرهای جهت* نام‌گذاری کنیم.



شکل ۵-۱: سلول‌ها برای یک نمونه از مسئله‌ی راه فرار مستطیل‌ها.

برای هر سلول c، مجموعه‌ای به نام P_c به این شکل تعریف می‌کنیم: به ازای هر $1 \leq i \leq n$ و هر جهت $\lambda \in \{l, r, u, d\}$، دوتایی (i, λ) در P_c قرار دارد، اگر و تنها اگر مستطیل R_i در صورت فرار در جهت λ، مسیر فرارش از سلول c عبور کند. بنابر تعریف، اگر مکان اولیه‌ی R_i روی سلول c قرار گرفته‌باشد، آن‌گاه هر چهار زوج (i, l)، (i, r)، (i, u) و (i, d) در P_c قرار دارند و در غیر این صورت، حداکثر یکی از این چهار زوج، عضو P_c است.

اکنون می‌توان محدودیت‌های خطی زیر را با این فرض که بیشینه چگالی نقاط برد برابر Z باشد، نوشت:

- به ازای هر i، باید $x_{i,l} + x_{i,r} + x_{i,u} + x_{i,d} \geq 1$. به بیان دیگر، هر مستطیل باید حداقل به یکی از چهار جهت فرار کند.
- چگالی همه‌ی سلول‌ها باید کم‌تر یا مساوی Z باشد، پس به ازای هر سلول c، محدودیت $\Sigma_{(i,\lambda) \in P_c} x_{i,\lambda} \leq Z$ وجود دارد.

بنا بر آن‌چه گفته‌شد، برنامه‌ریزی صحیح زیر معادل مسئله‌ی ۱ است. دقت کنید که در این برنامه‌ریزی صحیح، تعداد محدودیت‌ها از $O(n^2)$ است.

$$
\begin{aligned}
\text{minimize} \quad & Z \\
\text{subject to} \quad & \sum_{(i,\lambda)\in P_c} x_{i,\lambda} \le Z && \forall c \\
& x_{i,l} + x_{i,r} + x_{i,u} + x_{i,d} \ge 1 && \forall\ 1 \le i \le n
\end{aligned}
$$

همان گونه که پیش‌تر اشاره شد، دامنه‌ی همه‌ی متغیرهای جهت در برنامه‌ریزی صحیح داده‌شده، مجموعه‌ی $\{۰,۱\}$ است. دامنه‌ی متغیر Z را نیز می‌توان مجموعه‌ی همه‌ی اعداد طبیعی در نظر گرفت، هرچند می‌دانیم که مقدار Z در جواب بهینه، عضو $\{۱,\dots,n\}$ خواهد بود. برنامه‌ریزی خطی متناظر با این برنامه‌ریزی صحیح را می‌توان این گونه تعریف کرد که همه‌ی محدودیت‌ها مطابق برنامه‌ریزی صحیح باشند، ولی دامنه‌ی متغیرهای جهت در آن، همه‌ی اعداد گویای بازه‌ی $[۰,۱]$ در نظر گرفته‌شود و دامنه‌ی Z هم همه‌ی اعداد گویای بازه‌ی $[۱,n]$.

هرچند با فرض $NP \neq P$، برای حل برنامه‌ریزی صحیح مورد نظر هیچ الگوریتم کارایی وجود ندارد، ولی برنامه‌ریزی خطی متناظر را می‌توان در زمان چندجمله‌ای حل کرد. پاسخ برنامه‌ریزی خطی را OPT_{LP} می‌نامیم. فرض کنید که مقدار متغیر $x_{i,\lambda}$ در OPT_{LP} را ${x^*}_{i,\lambda}$ و مقدار متغیر Z در OPT_{LP} را نیز Z^* بنامیم. لازم به ذکر است که هر بردار امکان‌پذیر برای برنامه‌ریزی صحیح داده‌شده یک بردار امکان‌پذیر برای برنامه‌ریزی خطی متناظر نیز هست، پس برای برنامه‌ریزی صحیحی مورد نظر، بردار امکان‌پذیری نمی‌توان یافت که در آن مقدار متغیر Z کم‌تر از Z^* باشد. به بیان دیگر، کمینه چگالی ممکن برای فراری دادن مستطیل‌ها نمی‌تواند کم‌تر از Z^* باشد.

اکنون با داشتن OPT_{LP} می‌توان الگوریتم‌هایی تقریبی برای مسئله‌ی ۱ به دست آورد. در این بخش ابتدا یک الگوریتم تقریبی ساده با ضریب تقریب ۴ ارائه خواهد شد. این الگوریتم پیش‌تر در [۳] معرفی شده‌است. سپس الگوریتمی احتمالی ارائه خواهیم کرد که به ازای هر $\epsilon > ۰$، در صورتی که جواب مسئله راه فرار مستطیل‌ها به اندازه‌ی کافی بزرگ باشد، با احتمال بسیار بالایی پاسخی با ضریب تقریب $۱ + \epsilon$ به دست می‌دهد.

۵-۱ ضریب تقریب ۴

با داشتن OPT_{LP} و با بهره‌گیری از روش گرد کردن قطعی به سادگی می‌توان یک الگوریتم تقریبی با ضریب تقریب ۴ برای مسئله به دست آورد. برای این کار، با داشتن OPT_{LP}، یک بردار امکان‌پذیر برای برنامه‌ریزی صحیح اولیه، مطابق با آن‌چه در ادامه می‌آید، به دست می‌آوریم.

- برای هر مستطیل i، جهت λ را به گونه‌ای انتخاب می‌کنیم که مقدار $x^*{}_{i,\lambda}$ در بین چهار متغیر جهت مربوط به این مستطیل بیشینه باشد. اگر بیش از یک جهت با این ویژگی وجود داشت، یک جهت به دلخواه انتخاب می‌شود. سپس در برنامه‌ریزی صحیح معادل با مسئله‌ی ۱، مقدار متغیر $x_{i,\lambda}$ را برابر ۱ و مقدار سه متغیر جهت دیگری که مربوط به R_i هستند را ۰ در نظر می‌گیریم. این مقداردهی به متغیرها معادل با این است که مستطیل R_i در جهت λ فرار کند.
- مقدار متغیر Z در برنامه‌ریزی صحیح مورد نظر را $\lfloor ۴Z^* \rfloor$ قرار می‌دهیم.

پیش از هر چیز نشان می‌دهیم بردار صحیحی که این گونه به دست می‌آید، یک بردار امکان‌پذیر است. برای این منظور دقت کنید که اگر مستطیل R_i در جهت λ فرار کند، در OPT_{LP} مقدار $x^*{}_{i,\lambda}$ حداقل $\frac{1}{4}$ بوده‌است، چراکه $x_{i,l} + x_{i,r} + x_{i,u} + x_{i,d} \geq 1$. بنابراین، در نتیجه‌ی گرد کردن، مقدار هر متغیر جهت، حداکثر ۴ برابر شده‌است. پس اگر مقدار متغیر Z در برنامه‌ریزی صحیح مورد نظر برابر $\lfloor ۴Z^* \rfloor$ قرار داده‌شود، همه‌ی محدودیت‌های به شکل $\Sigma_{x \in P_c} x \leq Z$ رعایت خواهند شد. پس بردار به دست آمده یک بردار امکان‌پذیر برای برنامه‌ریزی صحیحی است که ارائه شده‌است.

از سوی دیگر، مقدار تابع هدف به ازای این بردار به دست آمده برابر همان $\lfloor ۴Z^* \rfloor$ است و به این ترتیب می‌توان نتیجه گرفت که الگوریتم گفته‌شده، الگوریتمی تقریبی برای مسئله‌ی راه فرار مستطیل‌ها با ضریب تقریب ۴ است.

۵-۲ ضریب تقریب هر اندازه نزدیک به ۱

اکنون به معرفی یک الگوریتم تقریبی با ضریب تقریبِ به میزان دلخواه نزدیک به ۱ در هنگامی که پاسخ بهینه به اندازه‌ی کافی بزرگ باشد، می‌پردازیم. به طور دقیق‌تر، الگوریتم ارائه‌شده در این بخش یک الگوریتم تصادفی است که اگر Z^* (پاسخ بهینه‌ی برنامه‌ریزی خطی) از $c_\epsilon \log n$، بیش‌تر باشد، آن گاه با احتمال زیاد جواب بدست آمده از $1+\epsilon$ برابر جواب بهینه بیش‌تر نیست. دقت کتید که c_ϵ عدد ثابتی است وابسته به ϵ (و مستقل از n یا هر پارامتر دیگری).

این الگوریتم، بر پایه‌ی گرد کردن تصادفی است که در بخش های پیشین به طور مختصر معرفی شده‌است. یک نکته‌ی مهم در این این الگوریتم، مستقل نبودن متغیرهای تصادفی حاصل از گرد کردن متغیرهای برنامه‌ریزی خطی است. به بیان دیگر، در روشی که ارائه می‌کنیم، بر خلاف روشی که توضیح داده شده، متغیرها به صورت مستقل از هم گرد نمی‌شوند، بلکه در گرد کردن آن‌ها وابستگی وجود دارد. ابتدا لم زیر را برای گرد کردن متغیرهای گویا به اعداد صحیح ارائه می‌دهیم.

لم ۱ *فرض کنید متغیرهای $x_i \in [0,1]$ داده شده‌اند و داریم $\Sigma x_i = 1$. می‌توان متغیرهای x_i را به یکی از اعداد ۰ یا ۱ گرد کرد به گونه‌ای که:*

- *احتمال گرد شدن متغیر x_i به یک، برابر با مقدار x_i باشد.*
- *یک و تنها یکی از متغیرهای داده شده ۱ شود و باقی همگی به ۰ گرد شوند.*

اثبات. لازم به یادآوری است که روش گرد کردن متغیرها به صورت مستقل و با احتمال x_i به ازای هر متغیر، در این جا کاربرد ندارد؛ چرا که در این روش، شرط دوم لم در برآورده نمی‌شود.

برای آن که بتوان هر دو شرط را به طور هم‌زمان برآورده کرد، روش مقابل را پیشنهاد می دهیم: به هرکدام از متغیرهای x_i یک زیربازه از بازه‌ی $[0,1)$ با طولی برابر با مقدار x_i نسبت می‌دهیم به طوری که هیچ دو بازه‌ای اشتراک نداشته باشند. به این ترتیب، از آن جایی که $\Sigma x_i = 1$، اجتماع این بازه‌های مجزا، کل $[0,1)$ را می‌پوشاند.

پس از این کار، یک عدد تصادفی به صورت یکنواخت در بازه‌ی $[0,1)$ انتخاب می‌کنیم و آن متغیر x_i را که نقطه‌ی انتخاب شده در بازه‌ی نسبت داده‌شده به آن باشد، به ۱ گرد می‌کنیم. سایر

متغیرها را نیز به ۰ گرد می‌نماییم. با توجه به این که اندازه‌ی بازه‌ی نسبت داده‌شده به هر متغیر x_i برابر است با مقدار آن متغیر، احتمال آن که یک متغیر به ۱ گرد شود، مساوی با مقدار x_i است. از طرف دیگر، با توجه به این که اجتماع این بازه‌های مجزا تمام بازه‌ی $[0,1)$ را می‌پوشانند و نقطه‌ی انتخاب شده در یک و تنها یک بازه خواهد بود، شرط دوم لم با این روش برآورده می‌شود.

□

اکنون با استفاده از لم بالا، الگوریتم تقریبی زیر را برای مسئله‌ی ۱ ارائه می‌دهیم.

- برای هر مستطیل i داریم $x^*_{i,l} + x^*_{i,r} + x^*_{i,u} + x^*_{i,d} = 1$. مطابق لم بالا، این چهار متغیر جهت را گرد می کنیم. فرض کنید که متغیر $\hat{x}_{i,\lambda}$ را برابر با مقدار گرد شده متغیر ${x^*}_{i,\lambda}$ در نظر بگیریم. به این ترتیب، از بین چهار متغیر $\hat{x}_{i,l}$، $\hat{x}_{i,r}$، $\hat{x}_{i,u}$ و $\hat{x}_{i,d}$، مقدار یکی برابر ۱ و مقدار سه‌تای دیگر ۰ است.

- مستطیل R_i را به جهتی فرار می‌دهیم که مقدار $\hat{x}_{i,\lambda}$ متناظر با آن جهت برابر ۱ باشد.

قضیه ۷ *الگوریتم گفته‌شده یک الگوریتم تقریبی با ضریب تقریب $1 + \epsilon$ برای مسئله‌ی راه فرار مستطیل‌ها است، هنگامی که $Z^* \geq (9/\epsilon^2)\ln n$.*

اثبات. همان گونه که توضیح داده‌شد، این الگوریتم به هر مستطیل یک و تنها یک جهت برای فرار نسبت می‌دهد. حال چگالی نقطه‌های قاب را با جواب Z^* مقایسه می‌کنیم. به ازای هر سلول در قاب مانند c، متغیر d_c را برابر با چگالی آن سلول وقتی مستطیل‌ها طبق الگوریتم گفته‌شده فرار می‌کنند، تعریف می‌کنیم. بنابر این داریم:

$$d_c = \Sigma_{(i,\lambda)\in P_c}\hat{x}_{i,\lambda}$$

طبق روش گرد کردن می‌توان گفت:

$$E[d_c] = E[\Sigma_{(i,\lambda)\in P_c}\hat{x}_{i,\lambda}] = \Sigma_{(i,\lambda)\in P_c}E[\hat{x}_{i,\lambda}] = \Sigma_{(i,\lambda)\in P_c}P[\hat{x}_{i,\lambda}] = \Sigma_{(i,\lambda)\in P_c}{x^*}_{i,\lambda} \leq Z^*$$

در عبارت بالا، تساوی دوم بر اساس خطی بودن امید ریاضی به دست آمده، تساوی سوم بر اساس قسمت شرط نخست بالا و تساوی چهارم بر اساس تعریف امید ریاضی برای متغیرهای

تصادفی شناسه[1]. نامساوی پایانی نیز نتیجه‌ی محدودیت‌های برنامه‌ریزی خطی است. بنابر آن چه به دست آمد، امید ریاضی چگالی هر سلول کم‌تر از یا مساوی با Z^* است. اکنون باید فاصله‌ی مقدار چگالی یک سلول از مقدار امید ریاضی آن را بررسی کنیم. ابزاری که برای این کار استفاده می‌کنیم، نامساوی چرنوف است که پیش از این مطرح شده‌است.

مطابق با آن‌چه در نامساوی چرنوف گفته شده، در این جا، متغیر تصادفی d_c مجموع تعدادی متغیر تصادفی است که دامنه‌ی آنها $\{۰,۱\}$ است. تنها نکته‌ای که باید در نظر داشت این است که در نامساوی چرنوف، شرط مستقل بودن متغیرها وجود دارد، در حالی که روش گرد کردن متغیرها در الگوریتم بالا مستقل نیست. در این باره می‌توان گفت:

- اگر مکان اولیه‌ی مستطیل R_i روی سلول c قرار داشته‌باشد، آن گاه هر چهار زوج (i,l)، (i,r)، (i,u) و (i,d) عضو P_c هستند. در این حالت، به جای چهار متغیر وابسته به R_i در عبارت $d_c = \Sigma_{(i,\lambda)\in P_c}\hat{x}_{i,\lambda}$، می‌توان عدد ۱ قرار داد؛ چرا که همواره یک و تنها یکی از این متغیرها مقدار ۱ خواهد داشت.

- اگر مکان اولیه‌ی مستطیل R_i روی سلول c قرار داشته‌باشد، آن گاه حداکثر یکی از چهار زوج (i,l)، (i,r)، (i,u) و (i,d) در P_c است. از سوی دیگر، به سادگی می‌توان دید که متغیرهای $\hat{x}_{i,\lambda}$ و $\hat{x}_{i',\lambda'}$ به ازای $i \neq i'$ مستقل از هم هستند.

بنابر آن چه گفته شد، برای هر سلول c، می‌توان فاصله‌ی d_c از امید ریاضی آن را با کمک نامساوی چرنوف بررسی کرد.

$$P(d_c \geq (۱+\epsilon)E[d_c]) \leq (\frac{e^\epsilon}{(۱+\epsilon)^{(۱+\epsilon)}})^{Z^*}$$

با کمک لم ۲، می‌توان کران بالای ساده‌تری نسبت به آن چه در نامساوی چرنوف آمده‌است، به دست آورد.

لم ۲ *به ازای* $۰ \leq \epsilon \leq ۱$ *داریم:*

$$(\frac{e^\epsilon}{(۱+\epsilon)^{(۱+\epsilon)}})^{Z^*} \leq e^{-Z^*\epsilon^۲/۳}$$

[1] به فصل مقدمات رجوع کنید

دقت کنید پاسخی که الگوریتم ما برای مسئله‌ی راه فرار مستطیل‌ها تولید می‌کند، برابر است با بیشینه مقدار d_c یا به بیان دقیق‌تر $max_c\{d_c\}$. هم‌چنین تعداد سلول‌های قاب حداکثر برابر است با $(۲n)^۲$. حال فرض کنید که به ازای یک ثابت عددی وابسته به ϵ مانند c_ϵ داریم:

$$Z^* \geq c_\epsilon \ln n$$

در نتیجه خواهیم داشت:

$$P\{max_c\{d_c\} \geq (۱+\epsilon)Z^*\} \leq \Sigma_c P\{d_c \geq (۱+\epsilon)Z^*\} \leq (۲n)^۲ \times n^{-c_\epsilon \epsilon^۲/۳}$$

بنا بر آن چه گفته شد، اگر ثابت عددی c_ϵ برابر با $۹/\epsilon^۲$ در نظر گرفته‌شود، احتمال آن که بیشینه چگالی در الگوریتم ما از $۱+\epsilon$ برابر پاسخ برنامه‌ریزی خطی بیش‌تر باشد، از $\frac{۴}{n}$ کم‌تر است. از آن جایی که Z^* کران پایینی برای مسئله‌ی ۱ است، می‌توان ادعا کرد وقتی $Z^* \geq c_\epsilon \ln n$، آن گاه با احتمال بالا خروجی الگوریتم ما حداکثر $۱+\epsilon$ برابر پاسخ بهینه است.

□

فصل ۶

نتیجه‌گیری

در این نوشتار، نتایج جدیدی در مورد مسئله‌ی راه فرار مستطیل‌ها به دست آمد که به طور خلاصه به شرح زیر هستند:

- ثابت شد مسئله‌ی ۲ به ازای $k = ۲$ در کلاس پیچیدگی NP-کامل قرار دارد. این در حالی است که پیش از این، NP-کامل بودن این مسئله به ازای $k \geq ۳$ در [۳] نشان داده شده‌بود و از سوی دیگر برای $k = ۱$، [۴] الگوریتمی چندجمله‌ای ارائه کرده‌بود. به این ترتیب، وضعیت مسئله‌ی ۲ برای هر عدد k مشخص شد.
- با فرض $NP \neq P$، نمی‌توان الگوریتمی تقریبی با زمان چندجمله‌ای و ضریب تقریب بهتر از $\frac{۳}{۲}$ برای مسئله‌ی ۱ یافت.
- پاسخ مسئله‌ی ۲ به ازای $k = ۱$ را می‌توان در زمان $O(n^۴)$ یافت که در مقایسه با الگوریتم ارائه شده در [۴] از زمان اجرای بهتری برخوردار است و در عمل می‌تواند بسیار قابل استفاده‌تر باشد.
- به ازای هر عدد $۰ < \epsilon$، برای مسئله‌ی ۱ الگوریتمی تقریبی و احتمالی با ضریب تقریب $۱ + \epsilon$ ارایه شد، ولی به شرط آن که پاسخ از عدد مشخصی (وابسته به ϵ) بیش‌تر باشد.

علی‌رغم الگوریتم تقریبی ارائه شده در این مقاله، هنوز یک پرسش جذاب در رابطه با این مسئله بی‌پاسخ مانده‌است: آیا می‌توان الگوریتمی تقریبی با ضریب تقریب بهتر از ۴ برای مسئله‌ی راه فرار مستطیل‌ها در حالت کلی یافت؟

سپاس

از استاد بزرگوارمان، دکتر ضرابی‌زاده که با کمک‌ها و راهنمایی‌های بی‌دریغشان، ما را در انجام این پروژه یاری داده‌اند، تشکر و قدردانی می‌کنیم.

هم‌چنین از آقای حسام منفرد که این مسئله را به ما پیشنهاد دادند و دوست عزیزمان، آقای صدرا یزدانبد که در به دست آوردن نتایج این مقاله با ما همکاری داشتند، صمیمانه سپاس‌گزاریم.

Bibliography

[1] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[2] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[3] Qiang Ma, Hui Kong, Martin D. F. Wong, and Evangeline F. Y. Young. NP-completeness and an approximation algorithm for rectangle escape problem with application to PCB routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(9):1356–1365, 2012.

[4] Hui Kong, Qiang Ma, Tan Yan, and Martin D. F. Wong. An optimal algorithm for finding disjoint rectangles and its application to PCB routing. In *Proceedings of the 47th ACM/EDAC/IEEE Design Automation Conference*, DAC '10, pages 212–217, 2010.

[5] Hui Kong, Tan Yan, and Martin D. F. Wong. Automatic bus planner for dense PCBs. In *Proceedings of the 46th ACM/EDAC/IEEE Design Automation Conference*, DAC '09, pages 326–331, 2009.

[6] Qiang Ma, Hui Kong, Martin D. F. Wong, and Evangeline F. Y. Young. A provably good approximation algorithm for rectangle escape problem with application to PCB routing. In *Proceedings of the 16th Asia South Pacific Design Automation Conference*, ASPDAC '11, pages 843–848, 2011.

[7] Qiang Ma, E.F.Y. Young, and M. D F Wong. An optimal algorithm for layer assignment of bus escape routing on PCBs. In *Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference*, pages 176–181, 2011.

[8] Muhammet Mustafa Ozdal, Martin D. F. Wong, and Philip S. Honsinger. An escape routing framework for dense boards with high-speed design constraints. In *Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '05, pages 759–766, 2005.

[9] Pei-Ci Wu, Qiang Ma, and Martin D.F. Wong. An ILP-based automatic bus planner for dense PCBs. In *Proceedings of the 18th Asia South Pacific Design Automation Conference*, ASPDAC '13, pages 181–186, 2013.

[10] Jin Tai Yan and Zhi Wei Chen. Direction-constrained layer assignment for rectangle escape routing. In *Proceedings of the 2012 IEEE Internat. System-on-Chip Conf.*, SOCC '12, pages 254–259, 2012.

[11] Jin Tai Yan, Jun Min Chung, and Zhi Wei Chen. Density-reduction-oriented layer assignment for rectangle escape routing. In *Proceedings of the Great Lakes Sympos. VLSI*, GLSVLSI '12, pages 275–278, 2012.

[12] Tan Yan, Hui Kong, and Martin D. F. Wong. Optimal layer assignment for escape routing of buses. In *Proceedings of the 2009 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '09, pages 245–248, 2009.

abstract

Motivated by a bus routing application, we study the following *rectangle escape* problem: Given a set S of n rectangles inside a rectangular region named *board*, extend each rectangle in S toward one of the four borders of the board so that the maximum density over the board is minimized, where the density of each point p of the board is defined as the number of extended rectangles containing p.

We show that the problem is hard to approximate to within any factor better than $\frac{3}{2}$. When the desired density is 1, we provide an exact algorithm that finds an optimal solution in $O(n^4)$ time, improving upon the current best $O(n^6)$-time algorithm. When the optimal density is sufficiently large, for any constant $\epsilon > 0$, we provide a randomized algorithm that achieves an approximation factor of $1 + \epsilon$ with high probability improving upon the current best 4-approximation algorithm available for the problem.

In the name of God

Sharif University of Technology

Computer Engineering Department

B.Sc. Thesis

Software Engineering

Title:

The Rectangle Escape Problem

By:

Ehsan Emamjomeh-Zadeh

Sepehr Assadi

Supervisor:

Dr. Hamid Zarrabi-Zadeh

June 2013