





دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

پروژه‌ی کارشناسی
مهندسی کامپیوتر - نرم افزار

عنوان:

مسئله‌ی راه فرار مستطیل‌ها

نگارش:

احسان امام‌جمعه‌زاده

سپهر اسدی

استاد راهنما:

دکتر حمید ضرابی‌زاده

تیرماه ۱۳۹۲



دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

پروژه‌ی کارشناسی
مهندسی کامپیوتر - نرم افزار

عنوان:

مسئله‌ی راه فرار مستطیل‌ها

نگارش:

احسان امام جمعه‌زاده

سپهر اسدی

استاد راهنما:

دکتر حمید ضرابی‌زاده

نمره:

امضای استاد راهنما:

امضای استاد ممتحن:

چکیده

در این رساله به بررسی مسئله‌ی راه فرار مستطیل‌ها می‌پردازیم که در مسیره‌ی بر روی برده‌های دیجیتال مورد نیاز است. مسئله‌ی راه فرار مستطیل‌ها را می‌توان این گونه 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 اضلاع مستطیل‌ها همگی موازی مرزهای 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 رساله، ابتدا سختی این مسئله را بررسی می‌کنیم و نشان می‌دهیم که با فرض $P \neq NP$ ، الگوریتمی چندجمله‌ای با ضریب تقریب بهتر از برای این مسئله وجود ندارد. سپس برای حالتی که بیشینه چگالی مجاز برابر با یک است، یک الگوریتم چندجمله‌ای با زمان اجرای $O(n)$ ارائه می‌کنیم که در مقایسه با الگوریتم پیشین این مسئله با زمان اجرا $O(n)$ ، از زمان اجرای بهتری برخوردار است. در پایان، به ازای هر مقدار $\epsilon > 0$ ، با این شرط که مقدار جواب به اندازه‌ی کافی بزرگ باشد، یک الگوریتم تقریبی احتمالی با ضریب تقریب $\epsilon + 1$ ارائه می‌دهیم. این درحالی است که ضریب تقریب بهترین الگوریتم تقریبی قبلی است.

فهرست مطالب

۸	۱	مقدمات
۸	۱-۱	کلاس‌های پیچیدگی
۹	۲-۱	الگوریتم‌های تقریبی
۹	۳-۱	تعاریف و قضایای احتمال
۱۱	۴-۱	برنامه‌ریزی خطی
۱۲	۵-۱	برنامه‌ریزی صحیح
۱۲	۶-۱	الگوریتم تقریبی بر پایه‌ی برنامه‌ریزی خطی
۱۴	۲	معرفی مسئله
۱۴	۱-۲	تعریف دقیق مسئله
۱۵	۲-۲	کارهای پیشین
۱۷	۳-۲	نتایج ما
۱۸	۳	سختی مسئله
۱۸	۱-۳	سختی مسئله برای چگالی ۲
۲۱	۲-۳	سختی مسئله برای چگالی بیش‌تر از ۲
۲۲	۳-۳	تقریب‌ناپذیری
۲۴	۴	الگوریتم دقیق برای چگالی واحد
۲۹	۵	الگوریتم تقریبی
۳۱	۱-۵	ضریب تقریب

۲-۵ ضریب تقریب هر اندازه نزدیک به ۳۱

۶ نتیجه گیری ۳۵

فهرست شکل‌ها

۱۵	۱-۲ یک نمونه از مسئله‌ی راه فرار مستطیل‌ها.
۱۹	۱-۳ کاهش از مسئله‌ی -ارضایپذیری.
۲۲	۲-۳ ساختن I_{Δ} با استفاده از چهار نمونه از I_{Δ} برای مسئله‌ی ۲.
۲۵	۱-۴ جهت‌های آزاد برای هر یک از مستطیل‌ها در مسئله‌ی ۲ به ازای $k =$
۲۶	۲-۴ یک نمونه از مسئله‌ی فرار به سه جهت.
۳۰	۱-۵ سلول‌ها برای یک نمونه از مسئله‌ی راه فرار مستطیل‌ها.

فصل ۱

مقدمات

این جمله توسط مینا اضافه شده است.
در این بخش، به معرفی برخی از مفاهیمی می‌پردازیم که در این رساله 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.

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.

^۱ Decision Problem

^۲ NP-Hard

کلاس پیچیدگی NP -کامل^۳: مجموعه‌ی مسئله‌هایی است که هم NP و هم NP -سخت باشند.

۲-۱ الگوریتم‌های تقریبی

مسئله‌های بهینه‌سازی گسسته، مسئله‌هایی هستند که در آن‌ها هدف، 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.

۳-۱ تعاریف و قضایای احتمال

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

- **فرایند تصادفی**^۷: به هر فرایندی که نتیجه آن نه به صورت قطعی که به صورت احتمالاتی مشخص می‌شود، فرایند تصادفی گفته می‌شود. 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، مجموعه‌ی {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 i م تعریف کرد که در صورت شیر آمدن سکه، مقدار و در غیر این صورت، مقدار می‌گیرد. در این رساله، برد همه‌ی متغیرهای تصادفی، زیرمجموعه‌ای متناهی از مجموعه‌ی اعداد صحیح است.

^۳ NP -Complete

^۴ Approximation Algorithms

^۵ Approximation Factor

^۶ Randomized Algorithms

^۷ Stochastic Process

^۸ Sample Space

^۹ Random Variable

- متغیر تصادفی شناسه^{۱۰}: یک متغیر تصادفی که برد آن $\{, \}$ است (یک زیرمجموعه از فضای نمونه را به و باقی را به می‌برد)، یک متغیر تصادفی شناسه است. به بیانی دیگر، می‌توان گفت که 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 R است، به صورت زیر تعریف می‌شود:

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

برای یک متغیر تصادفی شناسه بنا به تعریف داریم:

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

به بیان دیگر، برای یک متغیر تصادفی شناسه، امید ریاضی برابر است با احتمال 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:

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

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:

قضیه‌ی 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 تساوی‌های قضیه‌ی ۱ به ازای هر دو متغیر تصادفی X و Y برقرار است؛ چه مستقل باشند و چه نباشند.

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

- اگر $X = \{x, \dots, x_k\}$ مجموعه‌ای از متغیرهای تصادفی باشد که برد همه‌ی آن‌ها $\{, \}$ است، احتمال آن‌که مقدار حداقل یک $x_i \in X$ برابر شود، حداکثر $\sum_{x_i \in X} P(x_i =)$ است. دقت کنید که $\sum_{x_i \in X} P(x_i =)$ مجموع احتمال شدن این متغیرهاست.

^{۱۰} Indicator Random Variable

^{۱۱} Expected Value

^{۱۲} Variance

^{۱۳} Standard Deviation

- نامساوی مارکوف^{۱۴}: اگر برد متغیر تصادفی X ، زیرمجموعه‌ای از اعداد حقیقی نامنفی باشد، آن‌گاه به ازای هر $a > 0$ ، داریم:

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

یا به طور معادل، می‌توان نوشت به ازای هر $c > 0$:

$$P(X \geq cE[X]) \leq \frac{1}{c}$$

به بیان دیگر، احتمال آن‌که مقدار X از c برابر مقدار امید ریاضی بیش‌تر شود، حداکثر $\frac{1}{c}$ است.

- نامساوی چبیشوف^{۱۵}: برای متغیر تصادفی X با امید ریاضی و واریانس متناهی، به ازای هر $c > 0$ داریم:

$$P(|X - E[X]| \geq c\sigma(X)) \leq \frac{1}{c^2}$$

که در آن $\sigma(X) = \sqrt{\text{var}(X)}$.

- نامساوی چرنوف^{۱۶}: اگر x, \dots, x_n متغیرهای تصادفی مستقل از هم با دامنه‌ی $\{0, 1\}$ باشند و متغیر تصادفی $X = \sum_{i=1}^n x_i$ تعریف گردد، آن‌گاه داریم:

$$P\{X \geq (1 + \epsilon)E[X]\} < \left(\frac{e^\epsilon}{(1 + \epsilon)^{(1 + \epsilon)}}\right)^{E[X]}$$

هم‌چنین در ادامه‌ی این نامساوی، داریم:

$$\left(\frac{e^\epsilon}{(1 + \epsilon)^{(1 + \epsilon)}}\right)^{E[X]} \leq e^{-E[X]\epsilon/3}$$

۴-۱ برنامه‌ریزی خطی

هدف از یک برنامه‌ریزی خطی^{۱۷}، یافتن یک بردار با مؤلفه‌های حقیقی است که در نامساوی‌های خطی داده‌شده در مسئله صدق کند و مقدار یک عبارت خطی را نیز کمینه (یا بیشینه) نماید. این عبارت خطی، تابع هدف^{۱۸} نامیده می‌شود. فرض کنید x, \dots, x_n متغیرهایی با دامنه‌ی اعداد گویا باشند و

$$c_j \in \mathbb{Q}, \quad b_i \in \mathbb{Q}, \quad a_{i,j} \in \mathbb{Q} \quad (1 \leq i \leq m, 1 \leq j \leq n)$$

یک برنامه‌ریزی خطی در حالت کلی به شکل زیر است:

^{۱۴} Markov Inequality

^{۱۵} Chebyshev Inequality

^{۱۶} Chernoff Inequality

^{۱۷} Linear Programming (LP)

^{۱۸} Objective Function

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

در این برنامه‌ریزی خطی، عبارت خطی $\sum_{j=1}^n c_j x_j$ ، تابع هدف است. بدیهی است که اگر قصد بیشینه کردن تابع هدف را داشته باشیم، 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 برنامه‌ریزی خطی وجود دارند. یکی از بهترین الگوریتم‌های چندجمله‌ای که برای حل برنامه‌ریزی خطی ارائه شده، الگوریتم interior-point است که در سال ۱۹۸۴ توسط کارمارکار پیشنهاد شده است [۴]. زمان اجرای این الگوریتم از $O(nL)$ است که n تعداد متغیرهای برنامه‌ریزی خطی است و L تعداد بیت‌هایی است که برای نمایش متغیرها و محدودیت‌ها نیاز است.

۵-۱ 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 می‌توان نشان داد که مسئله‌ی حل 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 $\{0, 1\}$ است، یکی از ۲۱ مسئله‌ای است که کارپ به عنوان مسئله‌های NP -سخت معرفی کرده است [۴].

۶-۱ الگوریتم تقریبی بر پایه‌ی برنامه‌ریزی خطی

برای بسیاری از مسئله‌های کمینه‌سازی هزینه، می‌توان یک برنامه‌ریزی صحیح معادل نوشت که دامنه‌ی متغیرهای آن $\{0, 1\}$ است، ولی همان گونه که اشاره شد، با فرض $NP \neq P$ ، الگوریتم کارایی برای حل برنامه‌ریزی صحیح وجود ندارد تا با کمک آن، مسئله‌ی بهینه‌سازی مورد نظر نیز حل شود. در چنین مواردی، یک راهکار رایج، این است که برنامه‌ریزی صحیح متناظر با مسئله‌ی مورد نظر به یک برنامه‌ریزی خطی با همان محدودیت‌ها تبدیل

^{۱۹} Integer Programming (IP)

گردد و حل شود. بدیهی است که در برنامه‌ریزی خطی متناظر، دامنه‌ی متغیرها به جای مجموعه‌ی دو عضوی $\{, \}$ ، مجموعه‌ی همه‌ی اعداد گویای بازه‌ی $[,]$ 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 $\alpha < \alpha < 1$ انتخاب می‌شود. سپس هر متغیر که مقدار آن در OPT_{LP} کمتر از α است، در برنامه‌ریزی صحیح مقدار می‌گیرد و مقدار سایر متغیرها، همگی مقدار α در این جا اهمیت دارد، انتخاب مقدار مناسبی برای α است، به گونه‌ای که بردار حاصل از گرد کردن، یک بردار امکان‌پذیر برای برنامه‌ریزی صحیح باشد. از آنجایی که در بردار جدید به دست آمده، مقدار هر متغیر حداکثر α برابر شده‌است، مقدار تابع هزینه نیز حداکثر α برابر خواهد شد و به این ترتیب، یک الگوریتم با ضریب تقریب α خواهیم داشت.

- گرد کردن تصادفی

در این روش، مقدار هر متغیر، نه به صورت قطعی بلکه با احتمالی به 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

۱-۲ تعریف دقیق مسئله

در این فصل، مسئله‌ی راه فرار مستطیل‌ها^۱ را معرفی خواهیم کرد. این مسئله را می‌توان این گونه تعریف کرد:

مسئله‌ی ۱ (راه فرار مستطیل‌ها) یک ناحیه‌ی مستطیلی با مرزهای 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 رساله، برای سادگی، مسئله‌ی تصمیم‌گیری زیر را نیز تعریف می‌کنیم.

مسئله‌ی ۲ یک نمونه از مسئله‌ی راه فرار مستطیل‌ها و عدد طبیعی k داده شده‌اند. آیا می‌توان مستطیل‌ها را به گونه‌ای فراری داد که چگالی هیچ نقطه‌ای از قاب بیش‌تر از k نشود؟

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.

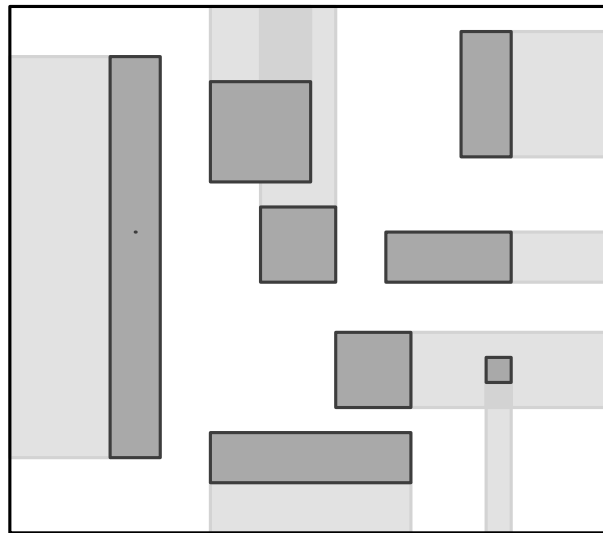
مسئله‌ی راه فرار مستطیل‌ها که در [۴] تعریف شده‌است، در مسیره‌ی روی بردهای دیجیتال^۳ مورد نیاز است. تراشه‌های مستطیلی شکلی را در نظر بگیرید که روی یک برد قرار گرفته‌اند، به گونه‌ای که اضلاع تراشه‌ها موازی اضلاع برد است. می‌خواهیم هر یک از تراشه‌ها را در یکی از چهار جهت اصلی از طریق یک گذرگاه^۴ (مطابق

^۱Rectangle Escape Problem (REP)

^۲Density

^۳Printed Circuit Board (PCB) Routing

^۴Bus



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

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، ۴، ۴، ۴، ۴، ۴، ۴، ۴، ۴، ۴]. گفتنی است که [۴] برای مسئله‌ی زیر الگوریتمی با زمان اجرای $O(n)$ ارائه کرده‌است:

مسئله‌ی 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.

به سادگی می‌توان دید که با کمک مسئله‌ی ۳ می‌توان مسئله‌ی ۲ به ازای k 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، n مستطیل خواهیم داشت که حداقل یک ضلع از هر کدام از آن‌ها روی مرز قاب قرار گرفته‌است. اکنون کافی است از میان این n مستطیل، بیش‌ترین تعداد مستطیل بدون هم‌پوشانی را بیابیم. این تعداد برابر n است اگر و تنها اگر مستطیل‌ها بتوانند با بیشینه چگالی 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 در کلاس پیچیدگی NP -کامل قرار دارد [۴].

- هرچند مسئله‌ی راه فرار مستطیل‌ها در حالت کلی NP -سخت است، ولی همان گونه که توضیح داده شد، برای مسئله‌ی ۲ به ازای $k =$ یک الگوریتم با زمان اجرای $O(n)$ در [؟] ارائه شده است.
- در [؟]، الگوریتمی تقریبی با زمان اجرای چندجمله‌ای و ضریب تقریب برای مسئله‌ی ۱ ارائه شده است.

۲-۳ نتایج ما

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

- مسئله‌ی ۲ به ازای $k =$ 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 ۱ یافت.
- مسئله‌ی ۲ وقتی $k =$ ، در زمان $O(n)$ قابل حل است.
- به ازای هر عدد ثابت $\epsilon >$ ، در حالتی که جواب مسئله‌ی ۱ به اندازه‌ی کافی بزرگ باشد، الگوریتمی احتمالی با ضریب تقریب $\epsilon +$ برای مسئله‌ی ۱ وجود دارد.

فصل ۳

سختی مسئله

در این بخش، ابتدا سختی مسئله ۲ به ازای k 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 مسئله ۱ الگوریتمی تقریبی با ضریب تقریب بهتر از نمی‌توان یافت.

۳-۱ سختی مسئله برای چگالی ۲

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

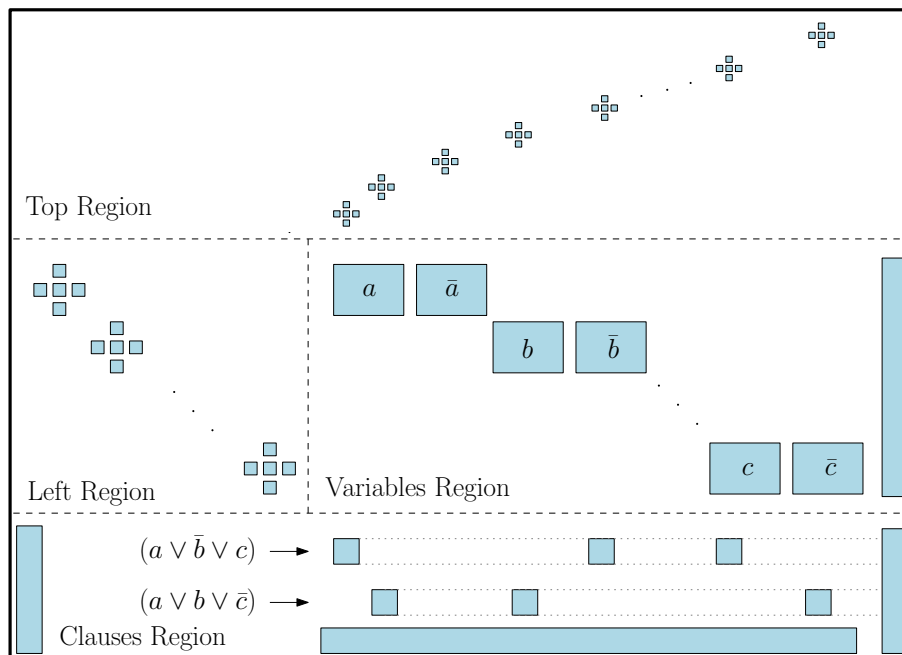
مسئله ۴ (-ارضاپذیری) برای مجموعه $X = \{x_1, \dots, x_n\}$ از متغیرهایی با دامنه‌ی $\{0, 1\}$ ، عبارت ۲ به شکل $C_i = l_{i,1} \vee l_{i,2} \vee \dots \vee l_{i,k}$ داده شده‌اند که در آن، هر $l_{i,\lambda}$ یکی از n متغیر عضو X یا نقیض یکی آن‌هاست. آیا می‌توان متغیرهای مجموعه X را به گونه‌ای مقداردهی کرد که مقدار همه‌ی این m عبارت منطقی، برابر شود؟

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

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

^۱-SAT

^۲Clause



شکل ۳-۱: کاهش از مسئله ی -ارضاپذیری .

اثبات. این که مسئله ی مورد نظر در کلاس پیچیدگی NP قرار می گیرد، بدیهی است. پس برای نشان دادن NP -کامل بودن این مسئله برای $k = 3$ ، کافی است NP -سخت بودن آن را ثابت کنیم. این کار را با کاهش^۳ از مسئله ی ۴ انجام می دهیم. به بیان ساده تر، ثابت می کنیم که اگر مسئله ی ۲ به ازای $k = 3$ در حالتی که هم پوشانی وجود ندارد، الگوریتمی با زمان اجرای چند جمله ای داشته باشد، آن گاه مسئله ی ۴ نیز در زمان چند جمله ای قابل حل است. به این منظور، برای یک نمونه از مسئله ی -ارضاپذیری، یک نمونه از مسئله ی راه فرار مستطیل ها به شکل زیر می سازیم: یک قاب مستطیلی در نظر می گیریم و آن را به صورت مجازی به چهار ناحیه تقسیم می کنیم. این ناحیه ها را مطابق شکل ۳-۱، ناحیه ی بالا^۴، ناحیه ی چپ^۵، ناحیه ی متغیرها^۶ و ناحیه ی عبارت ها^۷ می نامیم.

- به ازای هر متغیر x_j ، دو مستطیل با نام های v_j و \bar{v}_j متناظر با x_j و \bar{x}_j همان گونه که در شکل ۳-۱ نشان داده شده، در ناحیه ی متغیرها به صورت افقی کنار هم قرار می دهیم. این مستطیل های متناظر با متغیرها باید به گونه ای قرار بگیرند که یک خط عمودی یا افقی، دو مستطیل مربوط به دو متغیر مختلف را قطع نکند. فرض کنید این مستطیل ها را مستطیل های نوع یک بنامیم. علاوه بر مستطیل های نوع یک، یک مستطیل بلند نیز به صورت عمودی در سمت راست ناحیه ی متغیرها قرار می گیرد. شکل ۳-۱ را ببینید.

- برای هر عبارت $C_i = l_i \vee \bar{l}_i \vee l_i$ ، سه مستطیل در بخش عبارت ها قرار می دهیم. این سه مستطیل روی

^۳ Reduction

^۴ Top Region

^۵ Left Region

^۶ Variable Region

^۷ Clause Region

یک خط افقی به گونه‌ای قرار می‌گیرند که زیر مستطیل‌های نوع یک متناظر با l_i ، $l_{i,i}$ و l_i باشند. همه‌ی این مستطیل‌ها را نیز مستطیل‌های نوع دو می‌نامیم. همان گونه که در مثال شکل ۳-۱ نشان داده شده، یک خط عمودی 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، مربع کوچک که شکلی صلیبی ساخته‌اند، در ناحیه‌ی چپ قرار می‌گیرند. این شکل‌های صلیبی را سد می‌نامیم. دقت کنید که مربع یک سد به هر روشی که فرار کنند، چگالی نقطه‌ای روی یکی از آن‌ها حداقل 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 ساخته شده، مستطیل‌ها بتوانند به گونه‌ای فرار کنند که چگالی هیچ نقطه‌ای از قاب بیش‌تر از نشود. می‌خواهیم ثابت کنیم یک مقداردهی برای متغیرهای $\{x, \dots, x_n\}$ وجود دارد که به ازای آن مقداردهی، همه‌ی عبارت‌ها مقدار می‌گیرند.

با توجه به مکان قرارگیری سدها در ناحیه‌ی چپ و ناحیه‌ی بالا، هیچ مستطیل نوع یکی نمی‌تواند به چپ یا بالا فرار کرده باشد، 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 کرده باشند؛ چراکه در این صورت چگالی نقطه‌ای روی مستطیل بلند سمت راست ناحیه‌ی متغیرها، بیش‌تر از می‌شد. برای هر متغیر x_i ، اگر v_i به سمت راست فرار کرده بود، مقدار x_i را برابر قرار می‌دهیم و اگر \bar{v}_i به سمت راست فرار کرده بود، مقدار \bar{x}_i را برابر با در نظر می‌گیریم. اگر هم هیچ یک از این دو مستطیل به سمت راست فرار نکرده بودند (در واقع فرار هر دو به سمت پایین بود)، مقدار x_i را و در نتیجه مقدار \bar{x}_i را می‌گیریم. ادعا می‌کنیم که با این مقداردهی، همه‌ی عبارت‌ها برابر 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 $C_i = l_i \vee l_i, \vee l_i$ ، 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 کرده است. فرض کنید که مستطیل متناظر با $l_{i,\lambda}$ به پایین فرار کرده باشد ($\lambda \in \{1, \dots\}$). مستطیل نوع یکی که بالای این مستطیل قرار گرفته است، باید به سمت راست فرار کرده باشد، چون اگر به پایین فرار کرده باشد، چگالی نقطه‌ای روی مستطیل بلند پایین ناحیه‌ی عبارت‌ها بیش از خواهد شد. پس با توجه به نحوه‌ی مقداردهی متغیرها، مقدار $l_{i,\lambda}$ و در نتیجه مقدار عبارت C_i برابر است. به این ترتیب ثابت شد که با مقداردهی ارائه شده، مقدار همه‌ی عبارت‌ها برابر خواهد شد.

در سوی دیگر، باید ثابت کنیم که اگر برای نمونه‌ی داده شده از مسئله‌ی -رضاپذیری، یک مقداردهی وجود

داشته باشد که همه‌ی عبارت‌ها را کند، آن‌گاه مستطیل‌ها می‌توانند به گونه‌ای فرار کنند که بیشینه چگالی نقاط قاب باشد. به این منظور، برای هر متغیر x_i ، اگر $x_i =$ 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 $C_i = l_i \vee l_i, \vee l_i$ ، حداقل یک $\lambda \in \{, , \}$ وجود دارد که مقدار $l_{i,\lambda}$ برابر 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 سمت 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 گرفته است 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 باشد. گفتنی است که با این شیوه‌ی فرار، چگالی هیچ نقطه‌ای روی اضلاع قاب بیش‌تر از نخواهد شد.

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

□

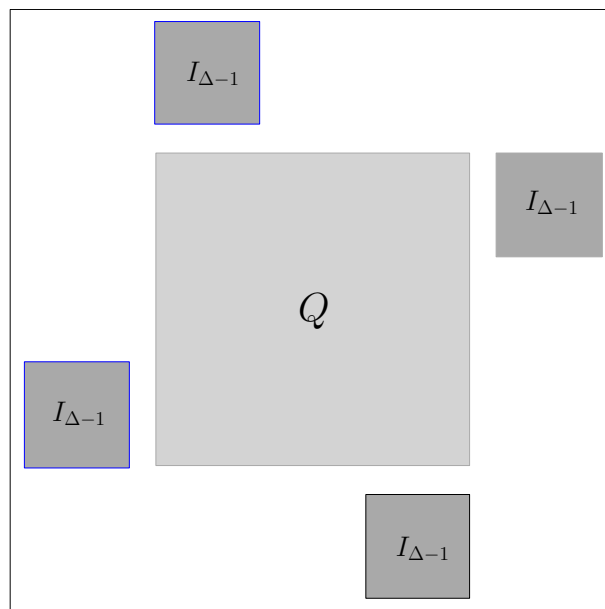
۲-۳ سختی مسئله برای چگالی بیش‌تر از ۲

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

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

اثبات. پایه‌ی استقرا معادل قضیه‌ی ۳ است که ثابت شد. اکنون فرض کنید که متناظر با نمونه‌ی داده‌شده‌ای از مسئله‌ی ۴، $I_{\Delta-}$ نمونه‌ای از مسئله‌ی ۲ به ازای $k = \Delta -$ باشد ($\Delta > 0$). برای ساختن I_{Δ} همان گونه که در شکل ۲-۳ نشان داده شده، یک مربع بزرگ Q در وسط می‌گذاریم و چهار نمونه از $I_{\Delta-}$ 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 $k = \Delta -$ برای نمونه‌ی $I_{\Delta-}$ 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 کرده است و در نتیجه از روی یکی از چهار نمونه‌ی $I_{\Delta-}$ عبور کرده است، می‌توان نتیجه گرفت که اگر $I_{\Delta-}$ به تنهایی در نظر گرفته شود، مستطیل‌ها می‌توانند به گونه‌ای فرار کنند که بیشینه چگالی، $\Delta -$ شود. در دیگر سوی، باید ثابت کنیم که اگر مستطیل‌ها در $I_{\Delta-}$ بتوانند با بیشینه چگالی $\Delta -$ فرار کنند، آن‌گاه در I_{Δ} ، مستطیل‌ها می‌توانند به گونه‌ای



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

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.

مشاهده‌ی ۱ به ازای هر $\Delta \geq$ ، 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 Δ) باشد.

- چگالی هر نقطه‌ای روی سه ضلع دیگر قاب حداکثر باشد.

در زیربخش قبلی، این نتیجه برای $\Delta =$ به دست آمده بود. از طرفی دیگر، فرض کنید مستطیل‌های هر چهار $I_{\Delta-1}$ در I_{Δ} به گونه‌ای فرار کرده باشند که برای هر یک از این چهار نمونه، بیشینه چگالی روی ضلع بالایی قاب $\Delta -$ باشد و چگالی نقاط سایر اضلاع قاب هم حداکثر. 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 شده است.

□

۳-۳ تقریب‌ناپذیری

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

قضیه ۵ (تقریب ناپذیری) برای مسئله ۱ الگوریتمی تقریبی با زمان چندجمله‌ای و ضریب تقریب بهتر از نمی‌توان یافت مگر آن‌که $NP = P$.

اثبات. گفتنی است که این نتیجه‌ی تقریب ناپذیری را نیز برای حالت بدون هم‌پوشانی می‌توان ثابت کرد. اگر الگوریتمی با ضریب تقریب کمتر از α 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 نشود: مستطیل‌ها را به عنوان ورودی به الگوریتم تقریبی دارای ضریب تقریب α می‌دهیم و پاسخ به دست آمده را با عدد مقایسه می‌نماییم.

- اگر پاسخ مسئله ۱ حداکثر باشد، آن‌گاه الگوریتم تقریبی باید پاسخی کمتر \times بیابد.

- اگر پاسخ مسئله ۱ حداقل باشد، پاسخی که الگوریتم تقریبی مورد نظر به دست می‌آورد نیز حداقل خواهد بود.

□

فصل ۴

الگوریتم دقیق برای چگالی واحد

در این فصل، یک الگوریتم با زمان اجرای $O(n)$ برای مسئله ۲ به ازای $k = ۲$ 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 $O(n)$ بوده که در [۴] ارائه شده‌است. در این بخش، با کمک برنامه‌ریزی پویا^۱، الگوریتمی با زمان اجرای $O(n)$ 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.

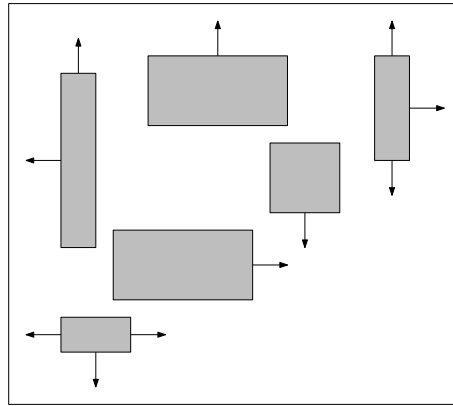
لازم به ذکر است که در مسئله ۵، مکان اولیه‌ی مستطیل‌های فرار نکرده هم اهمیت دارد: یک مستطیل در مسیر فرار خود نمی‌تواند با مکان اولیه‌ی هیچ مستطیل دیگری - حتی مستطیل‌های فرار نکرده - برخورد کند.

مستطیل‌های R_1, \dots, 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 مستطیل 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 گونه‌ای که ضلع پایینی R_i از ضلع پایینی R_{i+1} بالاتر نیست ($i < n$).

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

^۱Dynamic Programming

^۲Maximum Disjoint Escaping



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

این مستطیل را می‌توان به سادگی در زمان $O(n)$ به دست آورد، پس با یک پیش‌پردازش^۳ در زمان $O(n)$ جهت‌های آزاد همه‌ی مستطیل‌های داده‌شده را می‌توان یافت. گذشته از این، مجموعه‌ی $\{v, \dots, v_k\}$ 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 $O(n)$ می‌توان به دست آورد.^۴

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

One-Direction(i, l, r) -

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

Two-Direction(i, l, r) -

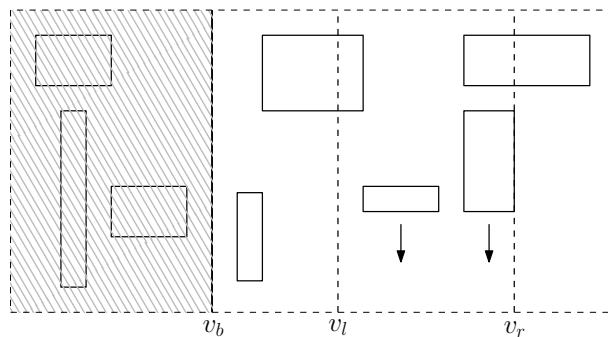
بیش‌ترین تعداد مستطیل از میان مستطیل‌های R_i, \dots, R_r که بین دو خط عمودی v_l و v_r قرار گرفته‌اند و می‌توانند به سمت بالا یا پایین فرار کنند.

این دو زیرمسئله به سادگی با الگوریتم‌های حریصانه^۵ قابل حل هستند. برای یافتن مقدار *One-Direction*(i, l, r) کافی است از میان مستطیل‌های R_i, \dots, R_r 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-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.

^۳Preprocess

^۴همه‌ی این پیش‌پردازش‌ها در زمان $O(n \log n)$ هم می‌توان انجام داد.

^۵Greedy



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

بنا به آنچه گفته شد، مقادیر $One-Direction$ و $Two-Direction$ را می‌توان در زمان $O(n)$ برای همه‌ی سه‌تایی‌های (i, l, r) به دست آورد^۶ و در آرایه‌هایی ذخیره نمود.

اکنون زیرمسئله‌های زیر را در نظر بگیرید:

مسئله‌ی ۶ (فرار به سه جهت) زیرمسئله‌ی $No-Left-Escape(i, b, l, r)$ بیش‌ترین تعداد مستطیل در میان مستطیل‌های R_1, \dots, 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.

شکل ۴-۲ را ببینید.

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

برای یافتن مقدار $No-Left-Escape(i, b, l, r)$ 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 $No-Left-Escape(i-1, b, l, r)$ 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).

^۶ این مقادیر با کمک برنامه‌ریزی پویا در زمان $O(n)$ نیز قابل محاسبه هستند.

- فرار به سمت پایین

اگر R_i به سمت پایین فرار کند، هیچ محدودیت جدیدی برای فرار مستطیل‌های R_i, \dots, R_{i-1} ایجاد نخواهد شد. بنابراین، بیش‌ترین تعداد مستطیل از بین این i مستطیل که می‌توانند با رعایت محدودیت‌های مسئله فرار کنند، برابر است با $No-Left-Escape(i-1, b, l, r)$ که به صورت بازگشتی قابل محاسبه است.

- فرار به سمت بالا

با فرار R_i به سمت بالا، یک محدودیت بر فرار i 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 $One-Direction$ و $Two-Direction$ به دست می‌آید. از سوی دیگر، بیش‌ترین تعداد مستطیل از R_i, \dots, R_{i-1} که سمت راست v_β قرار دارند و می‌توانند فرار کنند، برابر است با

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

که به صورت بازگشتی قابل محاسبه است.

- فرار به سمت راست

اگر R_i به سمت راست فرار کند، از بین i 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 مستطیل که می‌توانند فرار کنند، برابر است با:

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

زیرمسئله $No-Left-Escape$ 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 برنامه‌ریزی پویا تبدیل کرد. این ترتیب مقدار $No-Left-Escape$ برای همه‌ی چهارتایی‌هایی (i, b, l, r) در زمان $O(n)$ به دست می‌آید. هم‌چنین برای زیرمسئله $No-Right-Escape$ 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:

مسئله‌ی ۷ به ازای اعداد $1 \leq i \leq n$ و $1 \leq l, r \leq k$ ، بیش‌ترین تعداد مستطیل از بین مستطیل‌های R_i, \dots, R_1 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 برنامه‌ریزی پویا بازنویسی کرد و به این ترتیب، مقدار $Max-Route$ برای همه‌ی سه‌تایی‌های (i, l, r) در زمان $O(n)$ به دست می‌آید.

Algorithm 1 $Max-Route(i, l, r)$

1. **if** $i = 0$ **then**
 2. Return 0
 3. $ans_n \leftarrow Max-Route(i - 1, l, r)$
 4. $ans_d \leftarrow 0, ans_u \leftarrow 0, ans_l \leftarrow 0, ans_r \leftarrow 0$
 5. $\alpha, \beta \leftarrow$ indices of the vertical lines through the left and the right sides of R_i
 6. **if** *down* is feasible for R_i **then**
 7. $ans_d \leftarrow Max-Route(i - 1, l, r) + 1$
 8. **if** *left* is feasible for R_i **then**
 9. $ans_l \leftarrow Max-Route(i - 1, \max\{l, \beta\}, r) + 1$
 10. **if** *right* is feasible for R_i **then**
 11. $ans_r \leftarrow Max-Route(i - 1, l, \min\{r, \alpha\}) + 1$
 12. **if** *up* is feasible for R_i **then**
 13. $ans_u \leftarrow No-Right-Escape(i - 1, \alpha, l, r) + No-Left-Escape(i - 1, \beta, l, r) + 1$
 14. Return $\max\{ans_n, ans_d, ans_u, ans_l, ans_r\}$
-

در پایان، برای یافتن پاسخ مسئله‌ی ۵ تنها داشتن مقدار $Max-Route(n, k)$ کافی است که در آن، 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 فرار کنند. اکنون قضیه‌ی زیر را می‌توان نتیجه گرفت:

قضیه‌ی ۶ به ازای $k = 2$ ، مسئله‌ی ۲ در زمان $O(n)$ قابل حل است که در آن، تعداد مستطیل‌های ورودی است.

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

□

فصل ۵

کارهای پیشین

در این فصل به بررسی کارهای پیشین انجام شده در زمینه‌ی مسیریابی شبکه‌های مبتنی بر داده‌های نام‌گذاری شده می‌پردازیم. لازم به ذکر است

فصل ۶

نتیجه گیری

در این رساله، 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 $k =$ در کلاس پیچیدگی NP -کامل قرار دارد. این در حالی است که پیش از این، NP -کامل بودن این مسئله به ازای $k \geq$ در [۱] نشان داده شده بود و از سوی دیگر برای $k =$ ، [۲] الگوریتمی چندجمله‌ای ارائه کرده بود. به این ترتیب، وضعیت مسئله‌ی ۲ برای هر عدد k مشخص شد.
- با فرض $P \neq NP$ ، نمی‌توان الگوریتمی تقریبی با زمان چندجمله‌ای و ضریب تقریب بهتر از برای مسئله‌ی ۱ یافت.
- پاسخ مسئله‌ی ۲ به ازای $k =$ را می‌توان در زمان $O(n)$ یافت که در مقایسه با الگوریتم ارائه شده در [۳] از زمان اجرای بهتری برخوردار است و در عمل می‌تواند بسیار قابل استفاده‌تر باشد.
- به ازای هر عدد $\epsilon >$ ، برای مسئله‌ی ۱ الگوریتمی تقریبی و احتمالی با ضریب تقریب $\epsilon +$ ارائه 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 مانده است: آیا می‌توان الگوریتمی تقریبی با ضریب تقریب بهتر از برای مسئله‌ی راه فرار مستطیل‌ها در حالت کلی یافت؟

سپاس

از استاد بزرگوارمان، دکتر ضرابی زاده که با کمک‌ها و راهنمایی‌های بی دریغشان، 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.

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.



Sharif University of Technology
Computer Engineering Department

B.Sc. Thesis
Computer Engineering - Software

Title:

Rectangle Escape Problem

By:

Ehsan Emamjomeh-Zadeh

Sepehr Assadi

Supervisor:

Dr. Hamid Zarrabi-Zadeh

June 2013