مجله علمی- پژوهشی
«عمـران مـــدرس»
دوره چهاردهم، شماره ۳، پاییز ۱۳۹۳

ارزیابی و ارتقای عملکردی روش‌های کشف تماس برای استفاده در روش المان مجزا در مکانیک سنگ

حمیدرضا پاسه[۱]، محمود یزدانی[۲*]، مصطفی شریف‌زاده[۳]

۱- دانشجوی دکتری مهندسی عمران، مکانیک خاک و پی، دانشگاه تربیت مدرس

۲- استادیار مهندسی عمران، مکانیک خاک و پی، دانشگاه تربیت مدرس

۳- دانشیار مهندسی معدن، مکانیک سنگ، دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)

myazdani@modares.ac.ir

تاریخ دریافت: ۹۲/۱۲/۱۳ تاریخ پذیرش: ۹۳/۰٤/۱۱

چکیده- الگوریتم‌های کشف تماس، در شبیه‌سازی‌های المان مجزا، برای دستیابی به فهرست تماس‌های ممکن بین ذرات استفاده می‌شود. از آنجا که بخش مهمی از تلاش‌های محاسباتی در روش‌های المان مجزا مرتبط با کشف تماس ذرات است، کارایی الگوریتم مورد استفاده در این روش‌ها، از اهمیت بسیاری برخوردار است. این مقاله، با هدف شناسایی مناسب‌ترین الگوریتم کشف تماس و برای پیاده‌سازی یک نرم‌افزار تحلیل عددی هیدرو مکانیکی در مسائل مکانیک سنگ به روش المان مجزا (DA2)، الگوریتم‌های موجود برای کشف تماس بلوک‌های نامدور و اندازه‌های ناهمسان را مطالعه و ارزیابی می‌کند. برای این منظور، الگوریتم‌های کشف تماس، شامل بازرسی مستقیم (DC)، مرتب‌سازی و به‌روزاوری افزایشی (ISU) و مرتب‌سازی فضایی دو انتهایی (DESS)، در قالب نرم‌افزار DA2 پیاده‌سازی و در محیطی همسان و برای مسائل رایج در مکانیک سنگ اجرا شده و نتایج زمان اجرای آنها مقایسه شده‌است. نتایج پژوهش نشان می‌دهد که الگوریتم ISU در مقایسه با الگوریتم‌های DC و DESS، به لحاظ معیار زمان اجرا، کارایی بهتری داشته و به تغییرات پارامترهای مساله مانند تعداد بلوک‌ها، نسبت ابعادی مدل، تفرق اندازه‌ی بلوک‌ها، زوایای ناپیوستگی‌ها و زاویه‌ی اصطکاک داخلی درزه‌ها حساسیت کمتری نشان می‌دهد. در پایان، برای افزایش کارایی الگوریتم ISU، دو راه‌کار به‌روزآوری تاخیری و موازی‌سازی در به‌روزاوری، شناسایی و پیشنهاد شده‌اند. نتایج پیاده‌سازی راه‌کارها نشان داده‌است که با به‌کارگیری آن‌ها می‌توان تا ۲۰ درصد سرعت الگوریتم ISU را افزایش داد.

واژگان کلیدی: مکانیک سنگ، تحلیل عددی، روش المان مجزا، نرم‌افزار DA2، الگوریتم کشف تماس.

۱- مقدمه

الگوریتم‌های کشف تماس، در شبیه‌سازی‌های المان مجزا، برای دستیابی به فهرست تماس‌های ممکن بین ذرات استفاده می‌شود. از آنجا که بخش مهمی از تلاش‌های محاسباتی در روش‌های المان مجزا مرتبط با کشف تماس ذرات است، کارایی الگوریتم مورد استفاده در این روش‌ها، از اهمیتی بسیار حیاتی برخوردار است [۱].

نرم‌افزارهای متعددی مانند TRUBALL (کاندال و استراک، ۱۹۷۹)، UDEC (کانـــدال و هـــارت، ۱۹۸۵)، PFC (گـــروه مشاوران آیتاسکا، ۱۹۹۵) و DIBS (والتون، ۱۹۸۰) توسعه داده

شده‌اند که روش المان مجزا را برای حل مسائل خود استفاده می‌کنند. هر یک از این نرم‌افزارها به نوبه‌ی خود یک یا چند روش کشف تماس را برای این مهم به کار می‌برند.

نویسندگان این مقاله، با هدف پیاده‌سازی یک نرم‌افزار تحلیل عددی مسائل مکانیک سنگ به روش المان مجزا (DA2[1])، سعی کرده‌اند که با مطالعه‌ی الگوریتم‌های موجود کشف تماس و بررسی و مقایسه‌ی آنها، مناسب‌ترین الگوریتم را انتخاب و راه‌های بهبود عملکرد آن را بررسی و ارائه کنند. الگوریتم بهبودیافته، در پایان، برای پیاده‌سازی بخش تحلیلی نرم‌افزار DA2 به کار رفته است.

۲- الگوریتم‌های کشف تماس

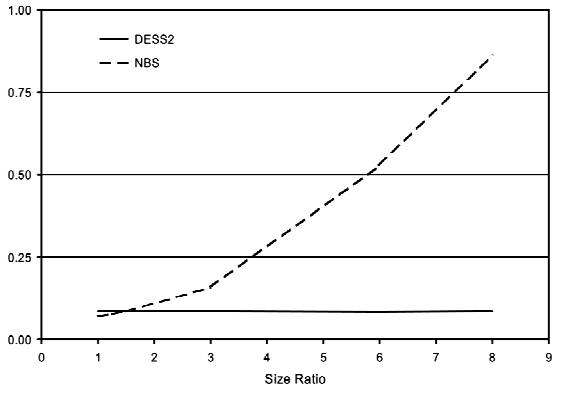
الگوریتم‌های کشف تماس، الگوریتم‌هایی است که به بررسی تماس‌ها و هم‌پوشانی‌های میان اشکال در صفحه یا فضا می‌پردازند. اشکال مورد بررسی در این الگوریتم‌ها، بیشتر مدور یا چندضلعی (یا چندوجهی) است (البته در تحلیل مسائل واقعی به اشکال پیچیده‌تر نیز بر خواهیم خورد). برخی از الگوریتم‌های کشف تماس به‌گونه‌ای طراحی شده‌اند که برای دسته‌ای از مسائل مناسب باشند، به‌طور مثال الگوریتم‌هایی که تنها به کشف تماس اجرام مدور می‌پردازند، و برخی دیگر جنبه‌ی عمومی دارند و روش آنها برای گونه‌های متفاوت شکل ذرات قابل استفاده است.

مسائل ژئوتکنیکی از جمله‌ی مسائلی است که شکل ذرات آنها دارای گوناگونی است. البته بیشتر از چندضلعی‌ها برای توصیف شکل ذرات استفاده می‌شود، پس الگوریتم‌های مورد بررسی در اینجا الگوریتم‌هایی است که به صورت عمومی قابل‌استفاده باشند.

بیشتر الگوریتم‌های کشف تماس که در مکانیک ذرات استفاده می‌شوند در دو دسته‌ی الگوریتم‌های مبتنی بر درهم‌سازی[2] و الگوریتم‌های مبتنی بر جعبه محیطی[3] قرار می‌گیرند.

الگوریتم‌های درهم‌سازی به طور عمده در تحلیل سیستم‌هایی به کار می‌روند که در آن ذرات دارای پراکندگی اندک در ابعاد باشند و در این نوع مسائل دارای کارایی بسیار بالایی هستند، ولی در صورتی که پراکندگی اندازه‌ی ذرات زیاد باشد، کارایی‌شان به شدت افت می‌کند [۲] (شکل ۱). پس در این پژوهش که نیاز به تحلیل سیستم‌هایی با پراکندگی دلخواه اندازه‌ی ذرات است، استفاده از الگوریتم‌های مبتنی بر درهم‌سازی مناسب نیست و از بررسی این گروه از الگوریتم‌ها چشم‌پوشی شده‌است.

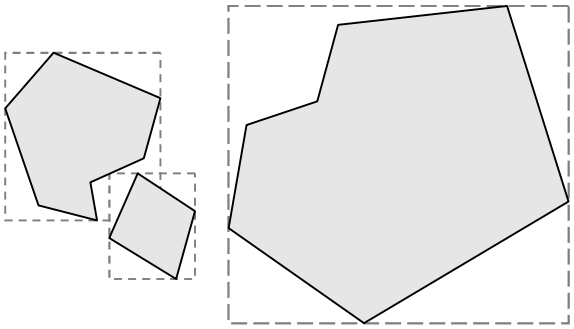
دسته‌ی دیگر، الگوریتم‌های مبتنی بر جعبه محیطی هستند (شکل ۲- راست). در این روش، حجمی ساده (مستطیل، دایره یا بیضی و در فضا یک مکعب، کره یا بیضی‌گون) در نظر گرفته می‌شود به صورتی که کوچک‌ترین حجم، در برگیرنده‌ی تمام حجم ذره باشد. استفاده از جعبه محیطی، اثر شکل ذرات را حذف می‌کند. بدین طریق تحلیل سیستم‌های ذرات با ابعاد متفرق با بازدهی مناسب قابل انجام خواهد بود [٤].



شکل (۱) مقایسه‌ی زمان اجرای الگوریتم‌های DESS (مبتنی بر جعبه محیطی) و الگوریتم NBS (مبتنی بر درهم‌سازی) نسبت تغییرات اندازه‌ی ذرات [۲].

1- Distinct Analysis 2D

2- Hashing
3- Bounding Box



شکل (۲) راست: یک ذره و جعبه محیطی آن. جعبه محیطی کوچکترین حجم در برگیرنده‌ی ذره را شامل می‌شود. چپ: دو جعبه محیطی که هم‌پوشانی دارند ولی ذره‌ها در تماس نیستند.

بررسی تماس حجم محیطی روش ساده‌ای دارد، پس با استفاده از آنها بررسی تماس بین ذرات بسیار ساده‌تر و سریع‌تر خواهد شد. البته تماس و هم‌پوشانی بین این حجم‌ها، به معنای تماس میان ذرات نیست و برای تحقق این امر و یافتن محل دقیق تماس‌ها نیاز به بررسی‌های بیشتری است (شکل ۲- چپ).

در این پژوهش، از این دسته، دو الگوریتم به همراه الگوریتم بازرسی مستقیم بررسی شده‌اند: الگوریتم مرتب‌سازی و به‌روزآوری افزایشی[1] و الگوریتم مرتب‌سازی فضایی دو انتهایی[2].

۲-۱- الگوریتم بازرسی مستقیم (DC[3])

در این روش، بررسی تماس میان بلوک‌ها برای هر جفت بلوک انجام می‌شود. معلوم است که زمان اجرای الگوریتم بازرسی مستقیم از مرتبه‌ی $O(N^2)$ است، که N تعداد بلوک‌ها و علامت O شاخص مرتبه‌ی زمانی تابع زمان است [۳]. از مزایای این روش این است که روشی برجا است، یعنی به حافظه‌ی اضافی برای اعمال این الگوریتم نیاز نیست.

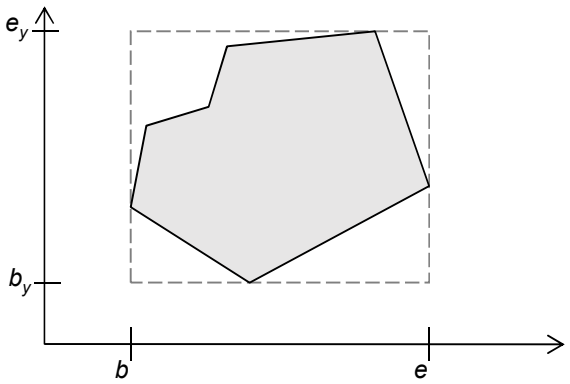
با وجود این‌که الگوریتم بازرسی مستقیم، یک الگوریتم مناسب و کارا در عمل نیست؛ این الگوریتم ساده‌ترین گزینه برای پیاده‌سازی است و در مواردی که تعداد بلوک‌ها کم باشد دارای عملکرد قابل قبولی است.

علاوه بر این، صحت خروجی این روش به سادگی قابل بررسی و تایید است و از خروجی آن می‌توان برای راستی‌آزمایی خروجی الگوریتم‌های دیگر استفاده کرد.

۲-۲- الگوریتم مرتب‌سازی و به‌روزآوری افزایشی (ISU)

در سال ۱۹۹۹، شینر[4]، الگوریتم مرتب‌سازی و به‌روزآوری افزایشی را برای حل مساله کشف تماس ارائه داد. برای این منظور، شینر از ایده‌ی جعبه‌های محیطی بهره جست [٤].

برای این الگوریتم، در مسائل دوبعدی، جعبه محیطی، مستطیلی با اضلاعی موازی با محورهای مختصات است و حدود آن به وسیله‌ی مقادیر b_x، b_y و e_x، e_y مشخص می‌شود (شکل ۳).



شکل (۳) ذره به همراه جعبه محیطی در صفحه‌ی مختصات و حدود آن در دستگاه مختصات

برای سادگی، ابتدا به صورت یک‌بعدی مساله پرداخته شده است و در ادامه، راهکار به ابعاد دیگر تعمیم داده شده است. در حالت یک بعدی، جعبه‌های محیطی به بازه‌هایی

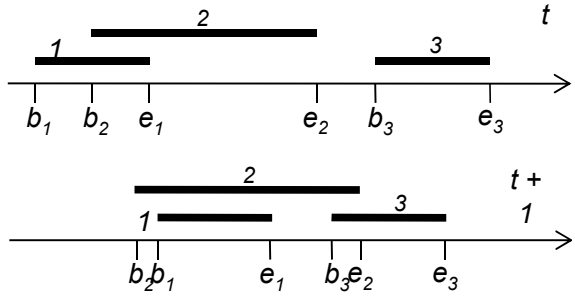
1- Incremental Sort and Update
2- Double-Ended Spatial Sorting
3- Direct Checking
4- Schinner

در محور x بدل می‌شوند. در شکل ٤، سه بازه مشاهده می‌شوند که در آن تنها بازه‌های ۱ و ۲ هم‌پوشانی دارند. نقاط ابتدا و انتهای بازه‌ها روی محور نشان‌گذاری شده‌اند. ترتیب قرارگیری این نشان‌ها بر محور، درست یکسان با ترتیب فهرست مرتب‌سازی شده‌ی مقادیر b_1، b_2، e_1، e_2، b_3 و e_3 است.

به طور کلی، اگر یکی از شرایط زیر اقناع شود:

- $b_m \le b_n \le e_m \le e_n$، یا
- $b_m \le b_n \le e_n \le e_m$، یا
- $b_n \le b_m \le e_m \le e_n$، یا
- $b_n \le b_m \le e_n \le e_m$

آنگاه دو بازه‌ی m و n در تماس هستند. از فهرست مرتب شده‌ی تمام b_iها و e_iها، می‌توان شرایط کافی و لازم برای تماس دو بازه و در نتیجه وضعیت هم‌پوشانی دو بازه را یافت. برای این منظور می‌توان از روش مرتب‌سازی و جاروب[1] برای ایجاد فهرست تماس‌ها در شروع کار کرد [٥ و ٦]. البته این روش خیلی مناسب نیست، زیرا برای تماس‌های فزاینده زمان‌بر بوده و کارایی ندارد [٤].



شکل (٤) بازه‌های در حال حرکت. تغییرات نقاط ابتدایی و انتهایی بازه‌ها قابل مشاهده است.

برای پرهیز از این مشکل، می‌توان در نظر گرفت که در یک سیستم فیزیکی، بلوک‌ها در گام‌های زمانی، فاصله‌ی کمی را طی می‌کنند. بدین‌صورت به جای ایجاد یک فهرست جدید از تماس‌ها برای هر گام زمانی، تنها فهرست تماس‌های پیشین می‌تواند اصلاح شود. در شکل ٤، می‌توان چنین شرایطی را زمانی که بازه‌ها به‌اندازه‌ی کوچکی حرکت کنند، مشاهده کرد. مقادیر b_iها و e_iها همه تغییر کرده‌اند، لیکن ترتیب آنها تغییر چندانی نکرده‌است.

جدول (۱) قوانین اصلاح وضعیت تماس‌ها هنگام مرتب‌سازی حدود

وضعیت تماس	t + 1	t
تماس در فهرست باقی می‌ماند.	b_nb_m	b_mb_n
تماس در فهرست باقی می‌ماند.	e_ne_m	e_me_n
تماس از فهرست حذف می‌شود.	e_nb_m	b_me_n
تماس به فهرست افزوده می‌شود.	b_ne_m	e_mb_n

هر تعویض در فهرست مرتب شده‌ی حدود می‌تواند وضعیت هم‌پوشانی دو بازه را تغییر دهد. قوانین این تعویض‌ها در جدول ۱ شرح داده شده‌اند. این قوانین باید برای هر تعویض اعمال شوند.

تعمیم این روش به دو بعد بسیار سر راست است. تصویر هر جعبه محیطی بر محور مختصات، دو بازه با نقاط ابتدایی و انتهایی b_x، b_y و e_x، e_y است. به طور مشابه وضعیت تماس زمانی تغییر می‌کند که حدود در دو محور مختصات عوض شوند. الگوریتم ISU برای هر دو محور به طور جداگانه اجرا می‌شود، اما قوانین اعمالی برای محور فرعی فقط زمانی اعمال می‌شوند که در محور اصلی هم‌پوشانی صورت گرفته باشد [٤].

۲-۳- الگوریتم مرتب‌سازی فضایی دوانتهایی (DESS)

این روش به وسیله‌ی پرکینز[2] و ویلیامز[3]، در سال ۲۰۰۱، ارائه شده است و اصلاحی بر یافتن تماس‌ها به روش جستجو در همسایگی است. در این روش نیز تعیین

1- Sort and Sweep

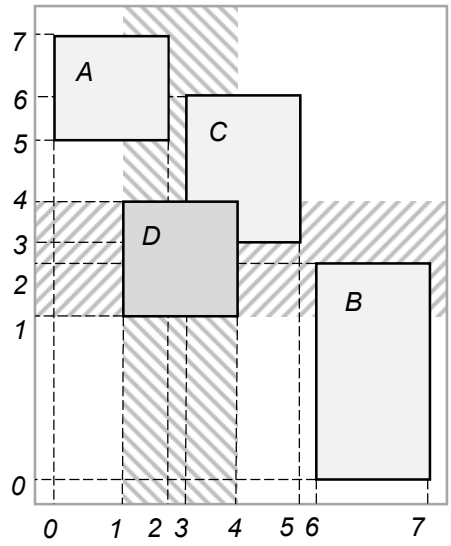
2- Perkins
3- Williams

همسایگی به وسیله‌ی جعبه محیطی به جای شکل بلوک انجام می‌شود. اصل روش جستجو در همسایگی، بر تعیین مجموعه‌ای دقیق از کاندیداهای تماس است که جفت بلوک‌هایی که جعبه‌های محیطی‌شان هم‌پوشانی نمی‌کنند را شامل نمی‌شود [۲].

هدف، در این الگوریتم، نزدیک کردن مجموعه‌ی کاندیداهای تماس به مجموعه‌ی دقیق کاندیداهای تماس و کمینه کردن زمان پردازش است. روش ارائه‌شده در این بخش، مبتنی بر روش قدیمی‌تر مرتب‌سازی فضایی است [٦]. این روش، مشابه روش ISU، دو انتهای تصویر جعبه محیطی ذره را بر محورهای مختصات در نظر می‌گیرد. زمانی که نقاط تصویر شده بر یک محور مرتب شوند، دو چیز را معین می‌کنند. اول: فهرست مرتب شده با مشخص کردن حدود هم‌پوشان که راهی را برای تعیین کاندیداهای تماس مهیا می‌کند؛ دوم: این فهرست یک روش جستجوی عددی برای مقایسه‌ی هم‌پوشانی میان جعبه‌های محیطی ارائه می‌دهد (شکل ۵- الف و ب).

روش DESS یک روش دو فازی است. فاز اول یا فاز مرتب‌سازی، یک نگاشت دوسویه بین حدود و رتبه‌شان در فهرست مرتب شده به دست می‌دهد. این نگاشت با در نظر گرفتن دو فهرست برای هر محور به دست می‌آید: یک فهرست مرتب شده بر اساس مقادیر، یک فهرست مرتب شده بر اساس شناسه. فهرست اول را فهرست شناسه می‌نامیم که رتبه‌ی مرتب‌شدگی را به شناسه نگاشت می‌کند. فهرست دوم را فهرست رتبه می‌نامیم که شناسه را به رتبه‌ی مرتب‌شدگی نگاشت می‌کند [۲].

هر دو فهرست با مرتب‌سازی فهرست شناسه بر اساس مقادیر و سپس به‌روز کردن فهرست رتبه به دست می‌آیند. این فرایند مرتب‌سازی و به روزآوری برای هر محور انجام می‌شود. برای هر محور نیاز به نگاه‌داری آرایه‌هایی با نسبت طول N است، پس حافظه‌ی مورد استفاده در این روش خطی است.



شکل (۵- الف) حدود و رتبه‌هایشان در محورهای اصلی و فرعی

شناسه x	
0	Al
1	Dl
2	Au
3	Cl
4	Du
5	Cu
6	Bl
7	Bu

رتبه x	
Al	0
Au	2
Bl	6
Bu	7
Cl	3
C	5
Dl	1
Du	4

شناسه y	
0	Bl
1	Dl
2	Bu
3	Cl
4	Du
5	Al
6	Cu
7	Au

رتبه y	
Al	5
Au	7
Bl	0
Bu	2
Cl	3
C	6
Dl	1
Du	4

شکل (۵- ب) فهرست‌های شناسه و رتبه‌ی محورهای x و y. علامت‌های l و u به‌ترتیب نشان دهنده‌ی حدود پایین و بالا هستند. از آرایه‌ی شناسه‌ی x، درمی‌یابیم که در این محور بلوک D با بلوک‌های A و C هم‌پوشانی دارد. به همین صورت، با توجه به آرایه‌ی شناسه‌ی y، بلوک D در محور y با بلوک‌های B و C هم‌پوشانی دارد.

در فاز دوم یا فاز به‌روزآوری، پس از به دست آمدن آرایه‌های رتبه و شناسه، یافتن کاندیداهای تماس تصویر شده برای هر یک از بلوک‌ها ساده است: رتبه‌ی حد پایین و بالای بلوک تصویر شده با مراجعه به فهرست رتبه یافته می‌شود. تمام شناسه‌هایی که در بازه‌ی رتبه‌ی حدود پایین

و بالای بلوک در فهرست شناسه قرار می‌گیرند، کاندیداهای تماس بلوک هستند.

برای تعمیم به حالت دوبعدی، با بررسی کاندیداهای تماس در محور فرعی، مجموعه‌ی بلوک‌های در تماس تعیین می‌شوند. لیکن در تعیین تماس‌ها در بررسی محور فرعی، از اعضایی که در محور اصلی کاندیدای تماس نیستند، چشم‌پوشی می‌شود[۲]. شکل ۵، فرایند کشف تماس را برای یک بلوک نشان می‌دهد.

کارایی الگوریتم DESS وابسته به نحوه‌ی انتخاب محور اصلی و توزیع فضایی بلوک‌هاست. پس نیاز است که راهکاری برای تعیین محور اصلی اندیشیده شود [۲].

۳- تحلیل ریاضی الگوریتم‌ها

برای مقایسه‌ی این الگوریتم‌ها از نظر زمان اجرا نیاز است که نخست آنها را از نظر پیچیدگی زمانی بررسی کنیم. برای این منظور به‌ترتیب شبه کدهای هر یک از الگوریتم‌ها ارائه و سپس گام‌های آن بررسی شده‌اند.

شبه کد الگوریتم ISU در شکل ٦ آمده‌است. در این شبه کد پارامترهای B، I و C به‌ترتیب حدود جعبه محیطی، رتبه‌ی مرتب‌شده‌ی حدود و فهرست تماس‌ها است. اندیس‌های x و y نشانگر محور مربوط به هر کدام از این پارامترها است.

شبه کد الگوریتم ISU نشان می‌دهد که این الگوریتم دارای دو گام یکسان Sort-And-Update-* برای محورهای X و Y است که مستقل از یکدیگر عمل می‌کنند، پس بررسی یکی از گام‌های مرتب‌سازی و به روزآوری می‌تواند در تعیین پیچیدگی کلی الگوریتم مفید باشد. گام Sort-And-Update-* در حقیقت یک پیاده‌سازی از الگوریتم مرتب‌سازی درجی[1] است که در هر تعویض آن، به‌روزآوری نیز گنجانده شده‌است.

1- Insertion Sort

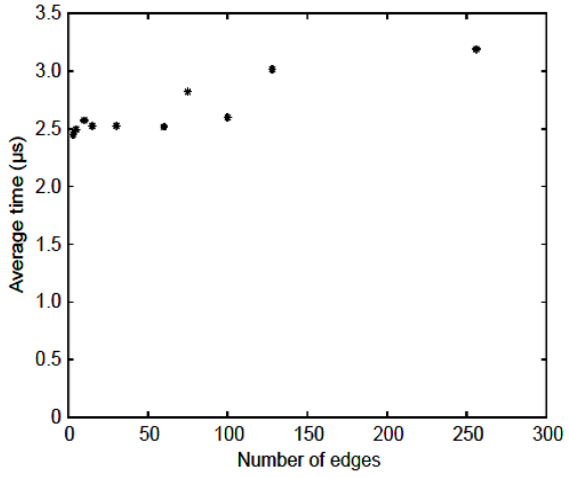
```
Incremental-Sort-And-Update(Bx, Ix, By, Iy, C)
Sort-And-Update-X(Bx, Ix, C)
Sort-And-Update-Y(By, Iy, C)

Sort-And-Update-X(Bx, Ix, C)
for i ← 2 to 2n do
j ← i – 1
while j ≥ 0 and Needs-Exchange(Bx, Ix, i, j) do
Exchange-X-And-Update(j, j + 1, Bx, Ix, C)
j ← j – 1

Sort-And-Update-Y(By, Iy, C)
for i ← 2 to 2n do
j ← i – 1
while j ≥ 0 and Needs-Exchange(By, Iy, i, j) do
Exchange-Y-And-Update(j, j + 1, By, Iy, C)
j ← j – 1
```

شکل (٦) شبه کد الگوریتم مرتب‌سازی و به‌روزآوری افزایشی

تعویض و به روزآوری، هر دو نیاز به زمان ثابتی برای انجام دارند؛ یعنی O(1) است. برای چنین الگوریتمی، رتبه‌ی پیچیدگی زمانی در بهترین حالت (برای فهرست مرتب‌شده) خطی O(N) و در بدترین حالت (زمانی که فهرست در جهت عکس مرتب شده باشد) درجه‌ی دو $O(N^2)$ است. رتبه‌ی زمانی میانگین $O(N^2)$ است، اما به دلیل آنکه در مسائل مورد بحث این مقاله، جابه‌جایی‌ها کوچک است، همواره فهرست مرتب‌شده و رتبه‌ی زمانی اجرای الگوریتم برای فهرست‌های تماس برابر O(N) است [۷] (شکل ۷).



شکل (۷) تغییرات خطی زمان اجرای الگوریتم ISU نسبت به افزایش ابعاد مساله [٤]

شبه‌کد الگوریتم DESS در شکل ۸ آمده‌است. در این شبه‌کد پارامترهای R، I و C به‌ترتیب فهرست رتبه‌ی مرتب‌شده‌ی حدود، فهرست شناسه‌ها و فهرست تماس‌ها هستند. اندیس‌های x و y نشانگر محور مربوط به هر کدام از این پارامترها است.

```
DoubleEnded-Spatial-Sorting(R_x, I_x, R_y, I_y, C)
Sort-X(R_x, I_x,)
Update-X(R_x, I_x, C)
Sort-Y(R_y, I_y)
Update-Y(R_y, I_y, C)

Sort-X(B_x, I_x)
for i ← 2 to 2n do
j ← i – 1
while j ≥ 0 and Needs-Exchange(B_x, I_x, i, j) do
Exchange-X (j, j + 1, B_x, I_x)
j ← j – 1

Update-X(R_x, I_x, C)
For each block, add blocks having boundaries between
current blocks lower and upper rank into C.

Sort-Y(B_y, I_y)
for i ← 2 to 2n do
j ← i – 1
while j ≥ 0 and Needs-Exchange(B_y, I_y, i, j) do
Exchange-Y (j, j + 1, B_y, I_y)
j ← j – 1

Update-Y(R_x, I_x, C)
For each block, remove blocks not having boundaries
between current blocks lower and upper rank from C.
```

شکل (۸) شبه کد الگوریتم الگوریتم مرتب‌سازی فضایی دوانتهایی

همان‌گونه که شبه‌کد الگوریتم DESS نشان می‌دهد، مانند الگوریتم ISU، این الگوریتم دارای دو گام یکسان *Sort- است. با این تفاوت که در آن عمل به‌روزآوری صورت نگرفته‌است، بلکه عمل به‌روزآوری در گام‌های مجزای *Update- انجام پذیرفته‌است. گام‌های *Sort- یک پیاده‌سازی از مرتب‌سازی درجی است که مطابق مطالب پیش‌گفته دارای رتبه‌ی زمانی O(N) برای فهرست‌های مرتب و رتبه‌ی زمانی بدترین حالت و میانگین $O(N^2)$ است.

گام *Update- به‌ازای هر بلوک، کاندیداهای تماس را از فهرست رتبه یافته و آن را به فهرست تماس می‌افزاید. اگر این بررسی فقط به ازای حدود پایین بلوک‌ها صورت گیرد، تضمین می‌شود که هر تماس تنها یک بار بررسی، و این گام با رتبه‌ی زمانی O(N) انجام شود. گام Update-Y تنها تماس‌هایی را بررسی می‌کند که در گام Update-X به فهرست افزوده شده‌اند، پس عملکرد آن وابسته به آن است. هرچه تعداد کاندیداهای تماس یافته‌شده در Update-X کمتر باشد، گام Update-Y سریع‌تر عمل خواهد کرد.

با توجه به موارد گفته‌شده، الگوریتم DESS نیز، مانند الگوریتم ISU، دارای رتبه‌ی زمانی $O(N^2)$ خواهد بود، اما به دلیل آن‌که در مسائل مورد بررسی این مقاله، تغییرات در فهرست‌ها کوچک است، این رتبه‌ی زمانی به O(N) نزدیک خواهد بود.

رتبه‌های زمانی حاصل از تحلیل ریاضی الگوریتم‌های ISU و DESS در جدول ۲ آمده‌است.

جدول (۲) رتبه‌های زمانی تحلیل ریاضی الگوریتم‌ها

الگوریتم	بهترین	بدترین	میانگین
ISU	O(N)	$O(N^2)$	$O(N^2)$
DESS	O(N)	$O(N^2)$	$O(N^2)$

٤- مقایسه‌ی عددی الگوریتم‌ها

در این پژوهش در مقام مقایسه‌ی الگوریتم‌های پیش‌گفته، ابتدا الگوریتم‌ها، در قالب نرم‌افزار DA2، پیاده‌سازی شده‌اند. برای پیاده‌سازی این الگوریتم‌ها از زبان F# و برای توسعه، ترجمه و اجرا از نرم‌افزار Microsoft Visual Studio 2012 و سیستم‌عامل ویندوز استفاده شده‌است و به دلیل مقایسه‌ی الگوریتم‌ها، ویژگی بهینه‌سازی کامپایلر غیرفعال شده است. همچنین، ویژگی‌های رایانه‌ی مورد استفاده برای مقایسه به شرح زیر است:

- Intel Core2 Duo P7530 @ 2.00 GHz
- 3 GB RAM
- 64-bit Operating System

- Windows 7

الگوریتم‌های پیاده‌سازی شده، برای به دست آوردن نتایج زمان اجرایشان، هر یک برای مدل‌های یکسان و برای ۳ ثانیه (زمان مکانیکی) اجرا شده‌اند. سپس از نتایج زمان اجرای گام‌هایشان میانگین‌گیری شده‌است. مدل نمونه، از شکستن یک بلوک سنگی اصلی (به ابعاد ۱۰×۱۰ متر) با خطوط ناپیوستگی مستقیم ایجاد شده‌است. پارامترهای متفاوتی که در ایجاد مدل دخالت داده شده‌اند به شرح جدول ۳ است.

جدول (۳) شرح پارامترهای موثر در مسائل مورد بررسی و نمادهای متناظر

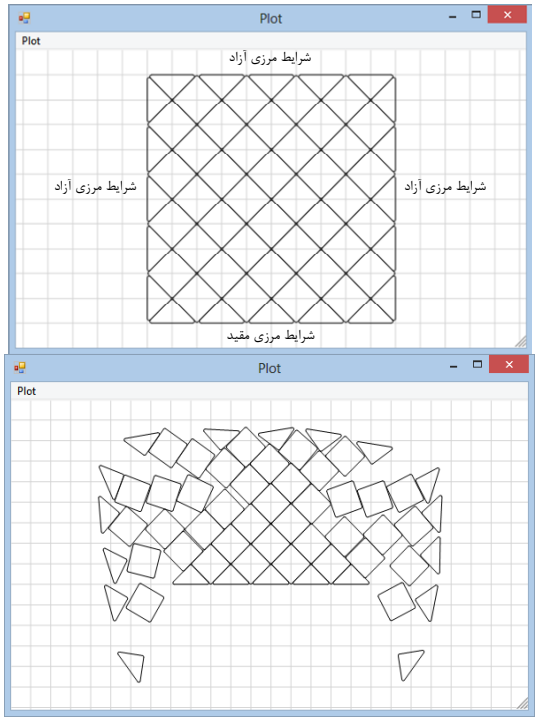
نماد	شرح پارامتر	واحد
N	تعداد بلوک‌ها	–
ρ	چگالی بلوک‌ها (=۲۵۰۰)	کیلوگرم بر مترمکعب
ϕ	زاویه‌ی اصطکاک داخلی	درجه
α_m	زاویه‌ی میانگین خط ناپیوستگی نسبت به محور x (برای توزیع احتمالی یکنواخت)	درجه
α_d	بیشینه‌ی انحراف زاویه‌ی ناپیوستگی از مقدار میانگین	درجه
s_m	فاصله‌ی افقی میانگین خطوط ناپیوستگی با یکدیگر (برای توزیع احتمالی یکنواخت)	متر
s_d	بیشینه‌ی انحراف فاصله‌ی خطوط ناپیوستگی از مقدار میانگین	متر
W/H	نسبت ابعاد مساله که به صورت نسبت عرض به ارتفاع بلوک اصلی تعیین می‌شوند.	–
σ	تفرق اندازه‌ی بلوک‌ها که به صورت انحراف معیار مساحت بلوک‌ها در نظر گرفته می‌شود.	–

٤-١- تعداد بلوک‌ها

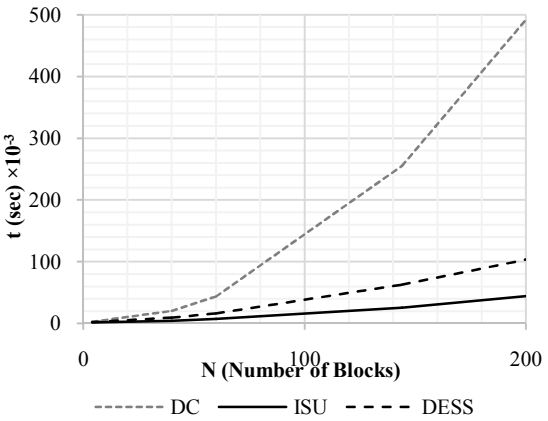
اولین گام برای مقایسه‌ی الگوریتم‌ها، مقایسه‌ی نتایج زمان اجرای آنها برای تعداد مختلف بلوک‌های موجود در مساله است. شکل ۹، نمونه‌ای از گراف‌های خروجی نرم‌افزار DA2 را نشان می‌دهد.

نتایج زمان محاسبه‌ی این الگوریتم‌ها برای تعداد بلوک‌های مختلف در شکل ۱۰ آمده است. آن‌گونه که این شکل نشان می‌دهد، الگوریتم DC بسیار زمان‌بر بوده، به گونه‌ای که برای مسائلی با تعداد بلوک بیشتر از ۳۰ عدد بسیار کند می‌شود و در اینجا تنها در مقام مقایسه بدان بسنده شده‌است. از طرفی مطابق شکل ۱۰، الگوریتم ISU دو برابر سریع‌تر از الگوریتم DESS عمل می‌کند و هر دو نسبت به DC بسیار بهتر و نزدیک به زمان خطی عمل می‌کنند.

در مقایسه‌ی الگوریتم‌های ISU و DESS که اهداف اصلی این پژوهش است، الگوریتم ISU نتایج زمان اجرای بهتری نسبت به DESS دارد و این امر به این دلیل قابل پیش‌بینی است که الگوریتم ISU با ادغام گام‌های مرتب‌سازی و به‌روزآوری موجب شده است که داده‌ها تنها یک بار پویش شوند، اما الگوریتم DESS در دو گام تمام داده‌ها را پویش می‌کند.



شکل (۹) خروجی برنامه‌ی DA2: پیش از اجرای شبیه‌سازی و پس از اجرای آن ($W = 10$ m، $H = 10$ m، $\alpha = 45$ و $\phi = 20$)



شکل (۱۰) زمان اجرای الگوریتم‌های مورد مطالعه نسبت به تعداد بلوک‌های مساله. (W = 10 m، H = 10 m، α = 45 و ϕ= 20)

برای بررسی بهتر این موضوع، برای هر کدام از این الگوریتم‌ها درصد زمان صرف شده در هر یک از فازها محاسبه شده و در جدول ٤ آورده شده‌است. مشاهده می‌شود که بخش به‌روزآوری الگوریتم DESS زمان قابل‌توجهی را به خود اختصاص داده است.

نزدیک بودن زمان به‌روزاوری به زمان مرتب‌سازی باعث می‌شود که پیاده‌سازی الگوریتم‌های مبتنی بر مرتب‌سازی و به‌روزاوری به شکل موازی دارای توجیه مناسبی باشد [۸]. در الگوریتم DESS انتخاب محور اصلی اختیاری است، ولی پیشنهاد شده است که محوری که دارای میانگین طول تماس کمتری است به عنوان محور اصلی انتخاب شود [۲]. البته این موضوع در مساله‌ای که مدلش دارای تقارن است، چندان قابل‌توجه نیست.

جدول (٤) درصد زمان صرف شده در بخش‌های الگوریتم‌های ISU و DESS نسبت به کل زمان

ISU	مرتب‌سازی	٥٥
	به‌روزآوری	٣١
	سایر	١٤
DESS	مرتب‌سازی	٣٨
	به‌روزآوری	٤٨
	سایر	١٤

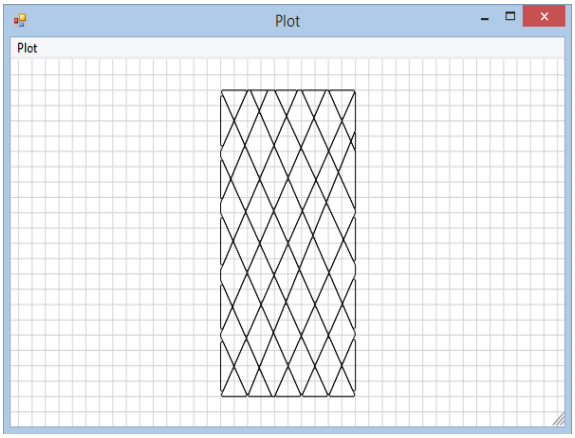
در ادامه، به دلیل زمان‌بر بودن الگوریتم DC، از بررسی و حساسیت‌سنجی آن نسبت به پارامترهای مساله چشم‌پوشی شده است.

٤-٢- نسبت ابعاد مساله

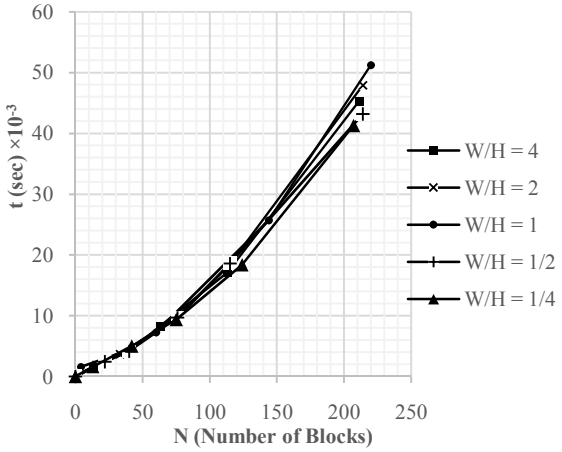
پارامتر مورد بررسی در اینجا، پارامتر W/H (نسبت عرض به ارتفاع بلوک اصلی) است. پارامتر W/H شاخص میزان توزیع بلوک‌ها در محورهای افقی و قائم مساله است (شکل ١١). نتایج زمان اجرای به دست آمده برای مقادیر W/H مختلف و نمودارهای متناظر آنها به ازای Nهای انتخاب‌شده در شکل‌های ١٢ تا ١٥ مشاهده می‌شود.

جدول ٥، نمایانگر درصد افزایش سرعت اجرای الگوریتم‌ها به ازای مقادیر مختلف W/H نسبت به حالت W/H = 1 است. مقادیر منفی به معنای کاهش سرعت و افزایش زمان اجرا است.

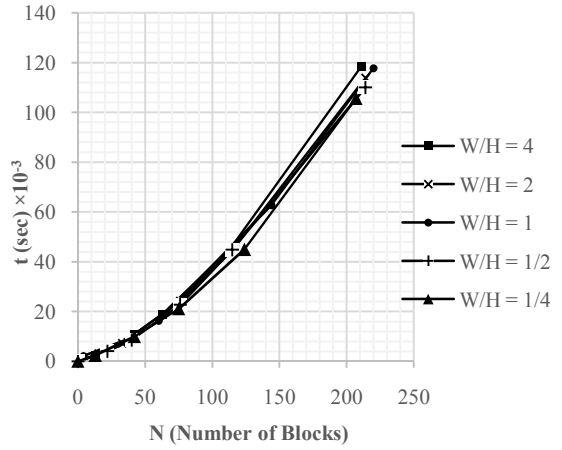
همان‌گونه که در جدول ٥ مشاهده می‌شود، الگوریتم ISU، زمانی که بلوک‌ها به صورت نامتقارن در محورها توزیع شده‌اند، نتایج زمان اجرای بهتری نشان می‌دهد. درحالی‌که الگوریتم DESS، زمانی که بلوک‌ها بیشتر در طول محور قائم توزیع شده‌اند، نتایج زمان اجرای بهتری دارد.



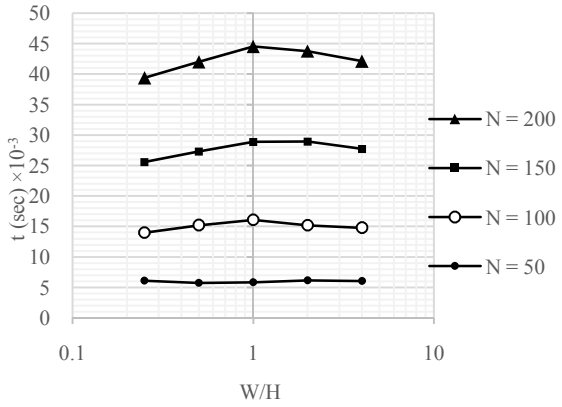
شکل (١١) مدل فیزیکی مساله به‌ازای W/H = 1/2



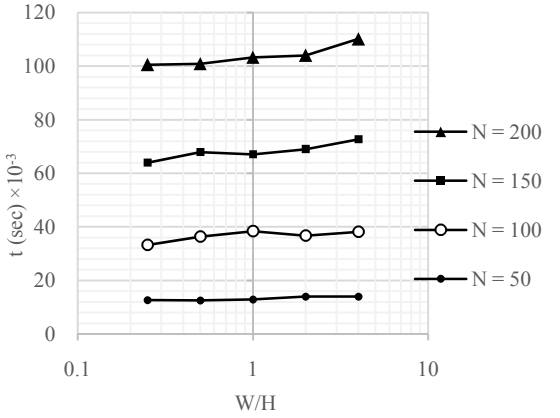
شکل (۱۲) نتایج زمان اجرا برای الگوریتم ISU به ازای تغییرات W/H



شکل (۱۳) نتایج زمان اجرا برای الگوریتم DESS به ازای تغییرات W/H



شکل (۱٤) نتایج زمان اجرا برای الگوریتم ISU به ازای تغییرات W/H و تعداد بلوک N



شکل (۱۵) نتایج زمان اجرا برای الگوریتم DESS به ازای تغییرات W/H و تعداد بلوک N

جدول (۵) درصد افزایش/کاهش سرعت اجرای الگوریتم‌های ISU و DESS نسبت به حالت W/H = 1 به ازای مقادیر مختلف W/H

W/H	ISU	DESS
٤/١	٪١١	٪٥
٢/١	٪٥	٪٢
٢	٪١	٪-١
٤	٪٥	٪-٦

٤-٣- تفرق اندازه‌ی بلوک‌ها

در گام بعدی، اثر تغییرات تفرق اندازه‌ی بلوک‌ها، σ، بر زمان اجرای الگوریتم‌ها بررسی شده است. پارامتر σ به صورت انحراف معیار مساحت بلوک‌ها تعریف شده است.

$$\sigma = \left(\frac{1}{N} \sum_{i=1}^{N} \left(A_i - \overline{A} \right)^2 \right)^{1/2}$$

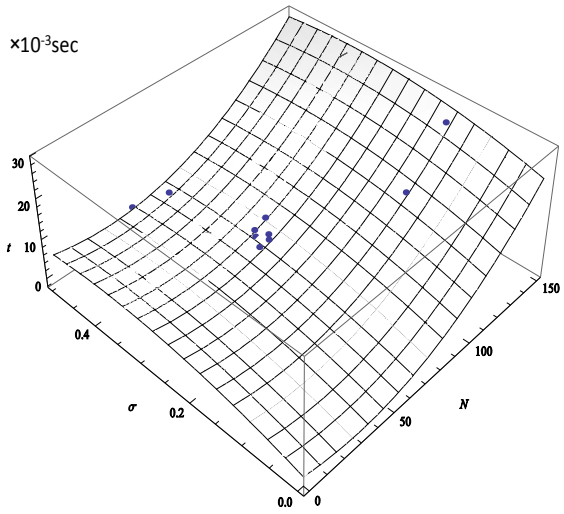
به منظور مقایسه، مدل‌هایی با مقادیر متفاوت N و σ، با تعیین مقادیر مختلف برای s_d ایجاد شده‌است. سپس هر دو الگوریتم برای مدل‌ها اجرا شده‌اند. از آن رو که هر دو پارامتر N و σ بر زمان اجرا موثرند، مدلی برای نحوه‌ی تاثیر این پارامترها در نظر گرفته شده است:

$$t = A \cdot N^2 + B \cdot N + C \cdot \sigma^2 + D \cdot \sigma$$

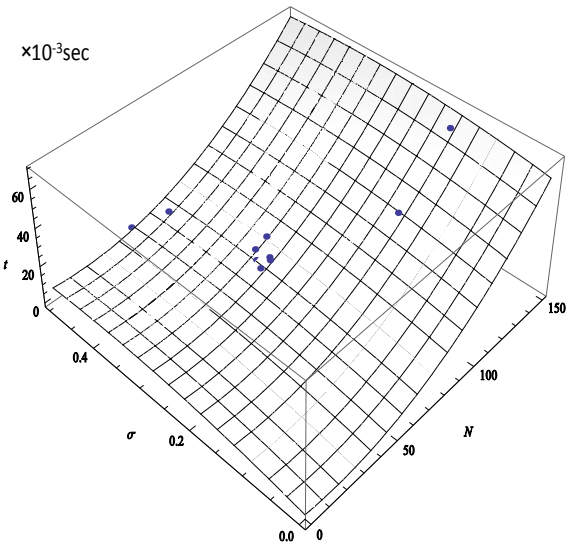
در این مدل مقادیر A، B، C و D به کمک درون‌یابی به وسیله‌ی نرم‌افزار Mathematica و تابع FindFit به دست

آمده‌اند. شکل‌های ١٦ الی ١٩ بـه طـور هـم‌زمـان تغییـرات حقیقی زمان اجرای الگوریتم‌ها را نسبت به تغییـرات تفـرق اندازه‌ی بلوک‌ها نشان می‌دهند.

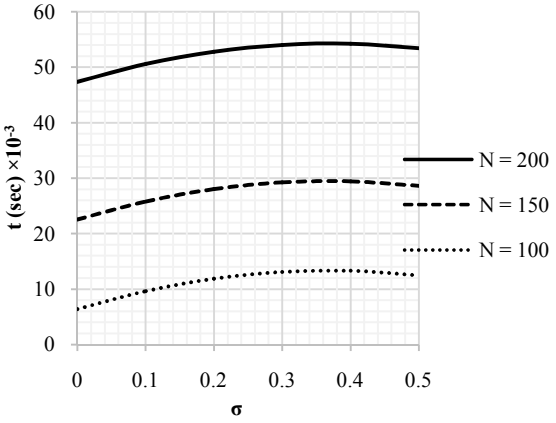
برای نشان دادن تاثیر σ، گراف‌هایی رسم شـده‌انـد کـه تغییرات زمان اجرا را برای مقادیر ثابت N نمـایش داده‌انـد. شـکل‌هـای ١٨ و ١٩، بـا بـرش مـدل بـرای مقـادیر انتخاب‌شده‌ی N به دست آمده‌اند.



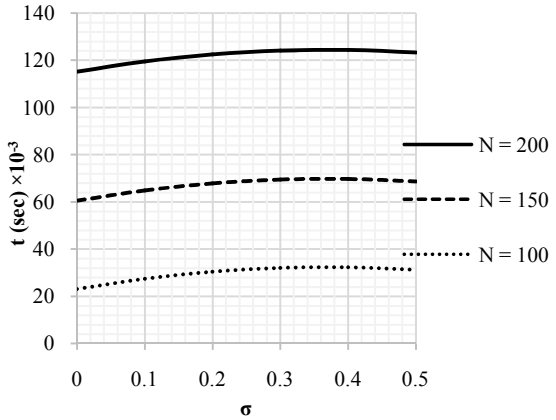
شکل (١٦) نتایج زمان اجرا برای الگوریتم ISU به ازای تغییرات N و σ



شکل (١٧) نتایج زمان اجرا برای الگوریتم DESS به ازای تغییرات N و σ



شکل (١٨) نتایج زمان اجرا برای الگوریتم ISU به ازای تغییرات σ برای مقادیر انتخاب‌شده‌ی N



شکل (١٩) نتایج زمان اجرا برای الگوریتم DESS به ازای تغییرات σ برای مقادیر انتخاب‌شده‌ی N

این نمودارها نشان می‌دهند که به طور کلی با افزایش مقدار σ، زمان اجرا به‌اندازه‌ای جزئی افزایش می‌یابد، و با افزایش تعداد بلوک‌ها می‌توان از اثر تفرق اندازه‌ی بلوک‌ها بر زمـان اجرای الگوریتم صرف نظر کرد.

٤-٤- زاویه‌ی ناپیوستگی‌ها

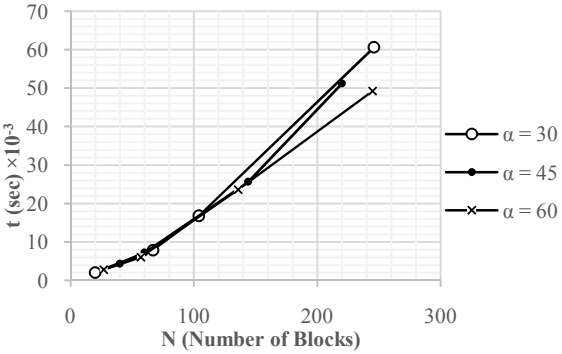
پارامتر دیگر مورد بررسی در اینجا، تاثیر زاویه‌ی ناپیوستگی بر زمان اجرای الگوریتم‌هاست. تغییرات α، بلـوک‌هـایی بـا نسبت ابعاد متفاوت می‌سازد و مقادیر بالای آلفا (کوچک‌تـر

از ۹۰ درجه)، موجب به وجود آمدن شیب‌های لغزش تندتر و در نتیجه کاهش پایداری مدل فیزیکی و افزایش جابه‌جایی بلوک‌ها می‌شود. شکل‌های ۲۰ الی ۲۳ اثر تغییرات زاویه‌ی α را بر زمان اجرای الگوریتم‌ها نشان می‌دهند. شکل‌های ۲۲ و ۲۳، به‌ترتیب نمودارهای متناظر با شکل‌های ۲۰ و ۲۱ هستند که اثر تغییرات زمان اجرا را به ازای مقادیر انتخاب‌شده‌ی N نشان می‌دهند. نتایج نشان می‌دهند که هر دو الگوریتم ISU و DESS با افزایش زاویه‌ی ناپیوستگی‌ها زمان اجرایشان کاهش می‌یابد.

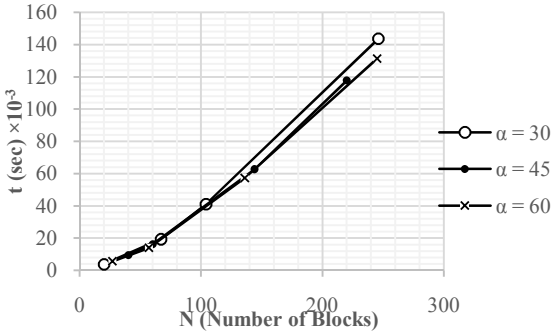
انتظار می‌رفت که با افزایش سرعت جابه‌جایی بلوک‌ها، و در نتیجه، افزایش تعداد تعویض‌ها، زمان اجرای الگوریتم‌ها افزایش یابد. لیکن به دلیل ناپایداری زیاد مدل، بلوک‌ها از یکدیگر جدا شده و تعدادی از تماس‌ها حذف می‌شوند و این امر موجب کاهش زمان اجرا می‌شود.

٤-٥- زاویه‌ی اصطکاک داخلی درزه‌ها

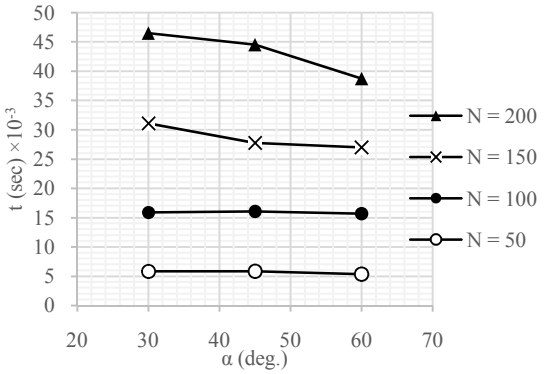
در پایان، اثر زاویه اصطکاک داخلی درزه‌ها، بر زمان اجرا بررسی شده است. بدیهی است که با انتخاب مقادیر بیشتر برای زاویه‌ی اصطکاک داخلی درزه‌ها (ϕ)، مقاومت برشی افزایش می‌یابد. پس انتظار می‌رود که تعداد تبادل‌ها در فاز مرتب‌سازی کاهش یافته، در پایان زمان اجرای الگوریتم نیز کاهش یابد.



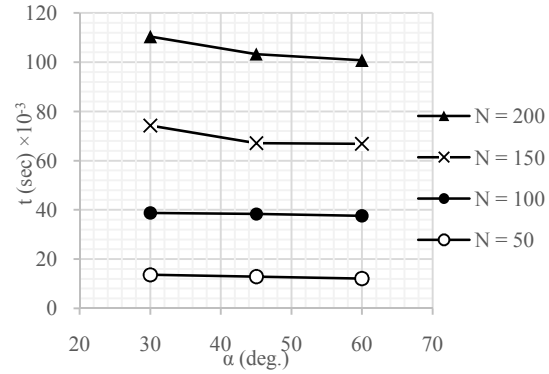
شکل (۲۰) نتایج زمان اجرا برای الگوریتم ISU به ازای تغییرات α



شکل (۲۱) نتایج زمان اجرا برای الگوریتم DESS به ازای تغییرات α



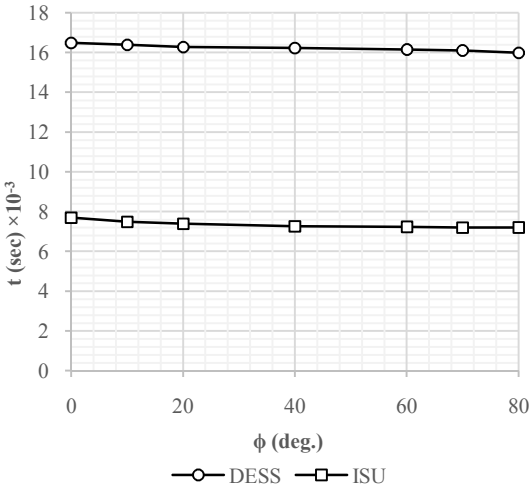
شکل (۲۲) نتایج زمان اجرا برای الگوریتم ISU به ازای تغییرات α و مقادیر انتخاب‌شده‌ی N



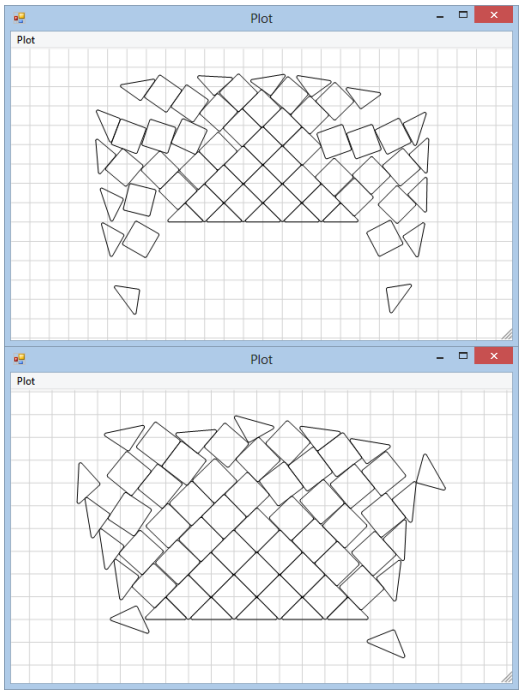
شکل (۲۳) نتایج زمان اجرا برای الگوریتم DESS به ازای تغییرات α و مقادیر انتخاب‌شده‌ی N

شکل ٢٤، اثر تغییرات زاویه‌ی اصطکاک داخلی را بر زمان اجرای الگوریتم‌ها نشان می‌دهد. مشاهده می‌شود که طبق پیش‌بینی انجام شده، با افزایش پایداری نمونه، سرعت

حرکت بلوک‌ها، و به تبع آن، تعداد تعویض‌ها کاهش یافته‌است. این کاهش تعداد تعویض‌ها، اثر خود را بر کاهش زمان اجرای الگوریتم‌ها نشان داده‌است. اما این تاثیر بسیار ناچیز و قابل چشم‌پوشی کردن است.



شکل (۲٤) اثر تغییرات زاویه‌ی اصطکاک داخلی (ϕ) بر زمان اجرا

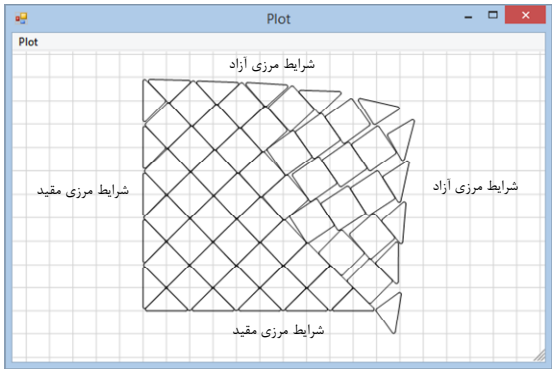


شکل (۲۵) بررسی اثر زاویه اصطکاک بر پایداری مدل به وسیله‌ی نرم‌افزار DA2: بالا) مدل تغییر شکل یافته برای مقدار زاویه اصطکاک داخلی ۲۰ درجه، پایین) و برای مقدار زاویه اصطکاک داخلی ٦۰ درجه

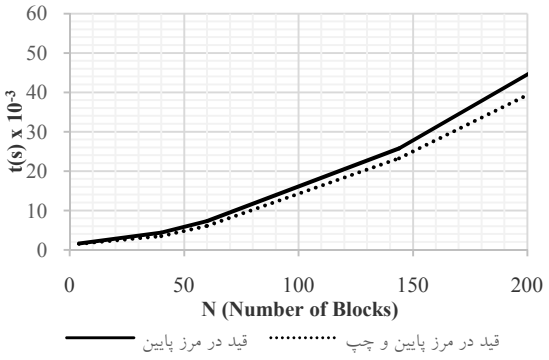
شکل ۲۵، گراف‌های تولیدشده به وسیله‌ی نرم‌افزار DA2 را برای مقادیر مختلف زاویه‌ی اصطکاک داخلی نشان می‌دهد. مشاهده می‌شود که با افزایش زاویه‌ی اصطکاک داخلی، پایداری مدل نیز افزایش یافته‌است.

٤-٦- شرایط تقید مساله

در این بخش به اثر شرایط تقید و مرزی مساله بر زمان اجرای الگوریتم‌ها پرداخته شده‌است. به این منظور در مساله پیشین، یکی از دیواره‌های جانبی را مقید و نتایج زمان اجرای الگوریتم‌ها را مقایسه کرده‌ایم (شکل ۲٦).

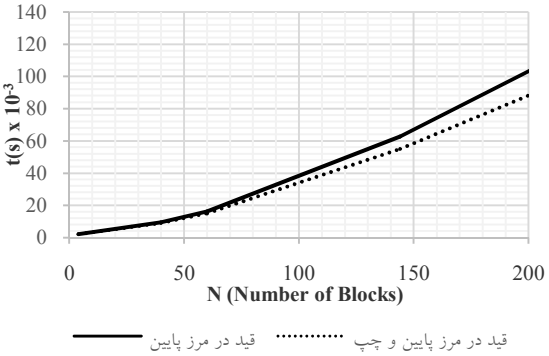


شکل (۲٦) خروجی گرافیکی برنامه DA2 پس از تغییر شرایط تقید (ϕ= 20 و α = 45 ،H = 10 m ،W = 10 m)



شکل (۲۷) اثر تقید بر زمانی اجرای الگوریتم ISU

شکل‌های ۲۷ و ۲۸ به‌ترتیب تغییرات زمان اجرای الگوریتم‌های ISU و DESS را به‌ازای Nهای مختلف و برای دو حالت تقید نشان می‌دهند.



شکل (۲۸) اثر تقید بر زمان اجرای الگوریتم DESS

در حالت اول تقید تنها به تکیه‌گاه پایین مدل اعمال شده‌است (شکل ۹) و در حالت دوم یکی از دیواره‌های جانبی نیز مقید شده‌است (شکل ۲۶). نتایج نشان می‌دهند که برای هر دو الگوریتم با افزایش شرایط تقید مقادیر جابه‌جایی بلوک‌ها کوچکتر شده، درنتیجه تغییرات فهرست تماس‌ها کاهش و سرعت اجرای الگوریتم مقداری افزایش یافته است.

۵- ارتقاء عملکردی روش ISU

از آن‌جایی که شبیه‌سازی مسائلی این‌گونه، حتی به کمک الگوریتم‌های سریع نیز بسیار زمان‌بر است، با توجه به توانایی رایانه‌های امروزی در پردازش داده‌ها به صورت موازی، اصلاح روش‌ها برای اعمال آنها به این‌گونه پردازش‌ها امری ضروری است. فازهای مرتب‌سازی و به‌روزآوری الگوریتم ISU از نظر زمانی که صرف می‌کنند به هم نزدیک هستند و این امر پیاده‌سازی این دو گام را به صورت موازی توجیه‌پذیر می‌کند. با توجه به جدول ٤ و هم‌پوشانی زمان پردازش دو فاز می‌توان پیش‌بینی کرد که موازی‌سازی این روش بیشینه تا ۳۰٪ زمان اجرای الگوریتم را کاهش می‌دهد.

روش مناسب برای انجام این امر در رایانه‌های ارزان امروزی برای کاربران نهایی که از مدل حافظه‌ی مشترک[1] بهره می‌برند، استفاده از پردازش چند رشته‌ای[2] است. ساختار این رایانه‌ها ویژگی‌های مهمی دارد. همه‌ی پردازش‌ها حافظه‌ی فیزیکی مشترکی را به کار می‌برند. پردازنده‌ها در این ساختار با ارسال پیغام به یکدیگر ارتباط ندارند، بلکه اطلاعات را میان یکدیگر به اشتراک می‌گذارند. یک نخ در این میان یک پردازش مستقل از دستورات برنامه است. مدیریت و زمان‌بندی اجرای این نخ‌ها را سیستم‌عامل انجام می‌دهد. در رایانه‌های دارای پردازنده‌های چندهسته‌ای، نخ‌ها ممکن است بر روی پردازنده‌های متفاوتی اجرا شوند [۹].

آنچه در ادامه ارائه می‌شود، راه‌کارهایی برای ارتقاء پیاده‌سازی این روش به صورت موازی است. این راه‌کارها در دو بخش به‌روزآوری تاخیری و موازی‌سازی به‌روزآوری بررسی شده‌اند.

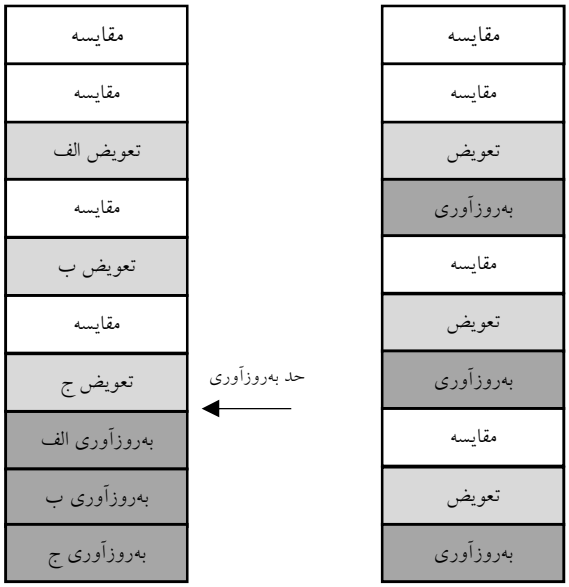
۵-۱- به‌روزآوری تاخیری

هر گام پردازش برای تعیین تماس‌ها در الگوریتم ISU شامل فازهای مرتب‌سازی و به‌روزآوری است. مرتب‌سازی به‌نوبه‌ی خود شامل گام‌های مقایسه و تعویض است. هر تعویض در فاز مرتب‌سازی، معادل با یک گام به‌روزآوری است، بنابراین با قرار دادن هر گام به‌روزآوری بعد از هر تعویض می‌توان این دو فاز را ادغام کرد، چنانکه در الگوریتم ISU صورت گرفته است (شکل ۲۹- راست).

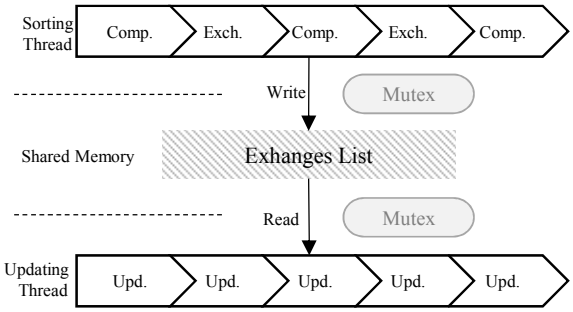
عمل به‌روزآوری نسبت به اعمال مقایسه و تعویض زمان‌برتر است و از سوی دیگر برای انجام عمل به روزآوری نیاز به بازپویش داده‌ها نیست و به روزآوری تنها به تعویض‌ها وابسته است، پس می‌توان عمل به روزآوری را با ثبت و نگاه‌داری تعویض‌ها از مرتب‌سازی جدا کرد.

1- Shared Memory

2- Multi-thread Processing



شکل ۲۹- راست) تعویض و به‌روزاوری متوالی، در مقایسه با رویکرد چپ) به‌روزآوری تاخیری



شکل (۳۰) استفاده از مدل حافظه‌ی مشترک برای پیاده‌سازی موازی مرتب‌سازی و به‌روزآوری و استفاده از Mutex برای دسترسی

برای مسائلی که جابه‌جایی بلوک‌ها در گام‌های زمانی بسیار ناچیز است (مشابه مسائل مکانیک سنگ)، تعداد تعویض‌ها نسبت به تعداد بلوک‌ها بسیار کم است، پس حافظه‌ی مورد استفاده برای نگاه‌داری و ثبت تعویض‌ها ناچیز خواهد بود. همچنین می‌توان عمل به‌روزآوری را پس از ثبت تعداد مشخصی تعویض انجام داد (به‌روزآوری تاخیری، شکل ۲۹- چپ).

۵-۲- موازی‌سازی به‌روزآوری

پس از آنکه به کمک ثبت تعویض‌ها توانستیم گام‌های به‌روزآوری را جدا کنیم، موازی‌سازی به‌سادگی قابل انجام خواهد بود. برای این منظور دو رشته جداگانه برای انجام مرتب‌سازی و به‌روزآوری نیاز خواهیم داشت. اطلاعات مشترک بین این دو پردازش، تعویض‌ها هستند؛ به گونه‌ای که رشته مرتب‌سازی تعویض‌ها را ثبت می‌کند و رشته به‌روزآوری آنها را می‌خواند.

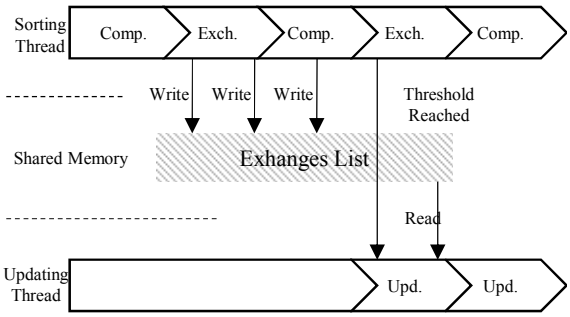
باوجود آنکه پیاده‌سازی الگوریتم به صورت موازی دارای سربار زمانی مدیریت حافظه‌ی مشترک و همچنین سربار حافظه‌ی مورد نیاز برای نگاه‌داری اطلاعات به اشتراک‌گذاشته است، در پایان کارایی الگوریتم به شکل قابل‌توجهی، به‌خصوص در رایانه‌هایی با پردازنده‌های چندهسته‌ای افزایش خواهد یافت. فرایند این روش را در شکل ۳۰ می‌توان مشاهده کرد.

در مدل‌هایی که دارای حافظه‌ی مشترک هستند برای کنترل دسترسی هرکدام از رشته‌ها به داده‌های مشترک و جلوگیری از خرابی آنها ناگزیر از استفاده از Mutex هستیم [۹]. به کمک Mutex می‌توانیم دسترسی خواندن و نوشتن را به صورت انحصاری کنترل کنیم. به گونه‌ای که هر بار دسترسی به حافظه‌ی مشترک یا برای خواندن آزاد است یا برای نوشتن.

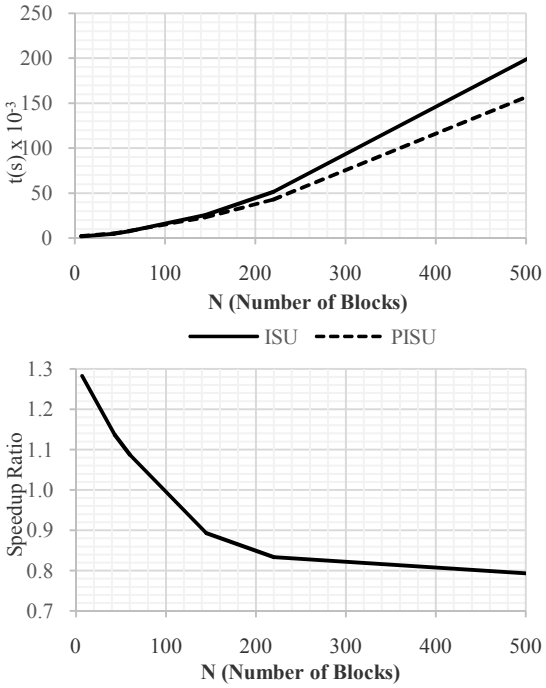
استفاده از Mutex به‌خودی خود دارای سربار زمانی است و به کمک به روزآوری تاخیری می‌توان از Mutex اجتناب کرد. این عمل بدین‌صورت انجام می‌گیرد که رشته مرتب‌سازی پس از ثبت تعداد مشخصی تعویض با ارسال سیگنالی به رشته به‌روزآوری آن را آغاز می‌کند و به روزآوری تا انتهای داده‌های ثبت‌شده تا پیش از سیگنال ادامه می‌یابد (تریگرینگ، شکل ۳۱).

۵-۳- نتایج بهینه‌سازی

به‌منظور مقایسه‌ی الگوریتم بهینه‌شده PISU و الگوریتم ISU، الگوریتم بهینه‌شده، پیاده‌سازی و سپس برای مساله‌ی شکل ۹ اجرا شده‌است.



شکل (۳۱) استفاده از مدل حافظه‌ی مشترک برای پیاده‌سازی موازی مرتب‌سازی و به‌روزآوری و استفاده از تریگرینگ برای دسترسی

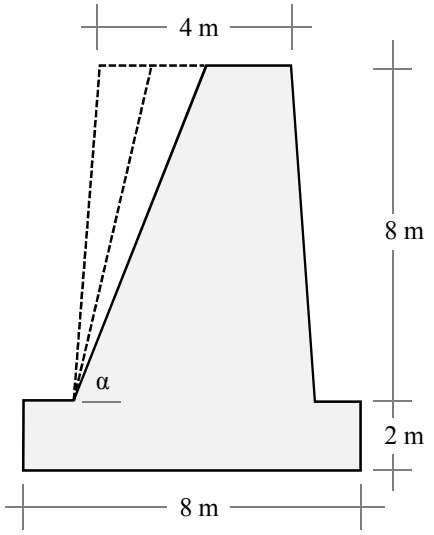


شکل (۳۲) بالا) زمان اجرای الگوریتم‌های ISU و PISU و پایین) نسبت زمان اجرای الگوریتم PISU به زمان اجرای الگوریتم ISU به ازای تغییرات تعداد بلوک‌ها

استخراج داده‌های زمانی برای الگوریتم PISU نشان می‌دهد که هر چه تعداد بلوک‌ها بیشتر باشد، سهم بهینه‌سازی الگوریتم افزایش می‌یابد. به ازای مقادیر کوچک N (کوچکتر از ۱۰۰)، سربار محاسباتی موردنیاز برای موازی‌سازی، سرعت اجرای الگوریتم را کاهش داده‌است، اما به دلیل ناچیز بودن زمان تحلیل برای تعداد بلوک‌های پایین، می‌توان از اثر منفی آن چشم‌پوشی کرد. این نتایج به صورت نسبت زمان اجرای الگوریتم بهینه‌شده به الگوریتم اصلی در شکل ۳۲ رسم شده‌اند.

همان‌گونه که در شکل ۳۲ ملاحظه می‌شود، با افزایش تعداد بلوک‌ها نسبت زمان اجرای الگوریتم PISU به الگوریتم ISU کاهش یافته و به مرز ۰/۸ رسیده است که به معنای ۲۰ درصد افزایش سرعت اجرای الگوریتم است.

در ادامه، برای مقایسه‌ی بهتر الگوریتم ISU و الگوریتم PISU، مساله‌ی دیگری حل شده‌است. مساله‌ی مورد بررسی در این بخش، یک شیروانی مطابق شکل ۳۳ است. برای این شیروانی سعی شده‌است با تغییرات شیب یکی از دیواره‌های آن، پایداری سازه بررسی شود.



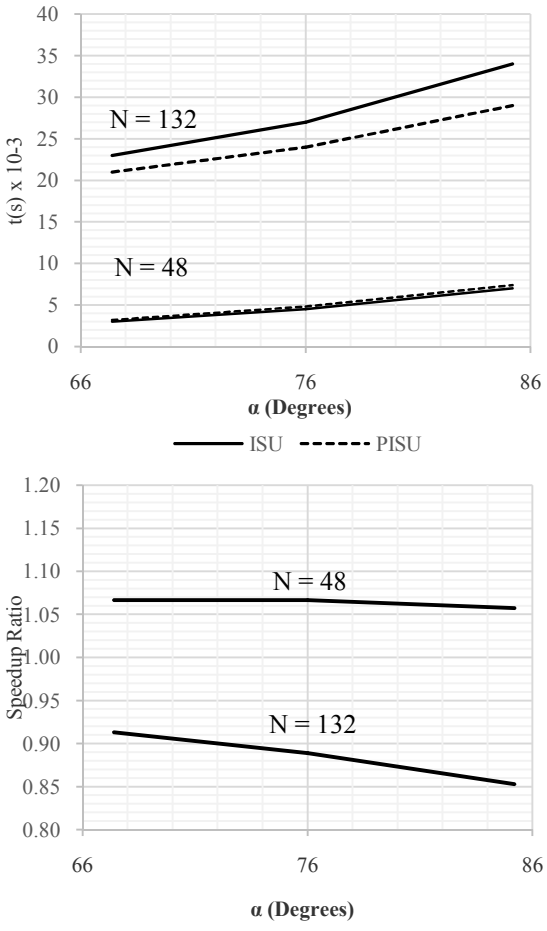
شکل (۳۳) مساله شیروانی برای الگوریتم‌های ISU و PISU

نتایج شبیه‌سازی برای حل مساله‌ی شیروانی (شکل ۳٤)، نشان می‌دهد که با کاهش زاویه‌ی شیروانی و افزایش پایداری مدل فیزیکی، کاهش جابه‌جایی بلوک‌ها و درنتیجه، کاهش تغییرات در فهرست تماس‌ها، زمان اجرای الگوریتم نیز کاهش می‌یابد.

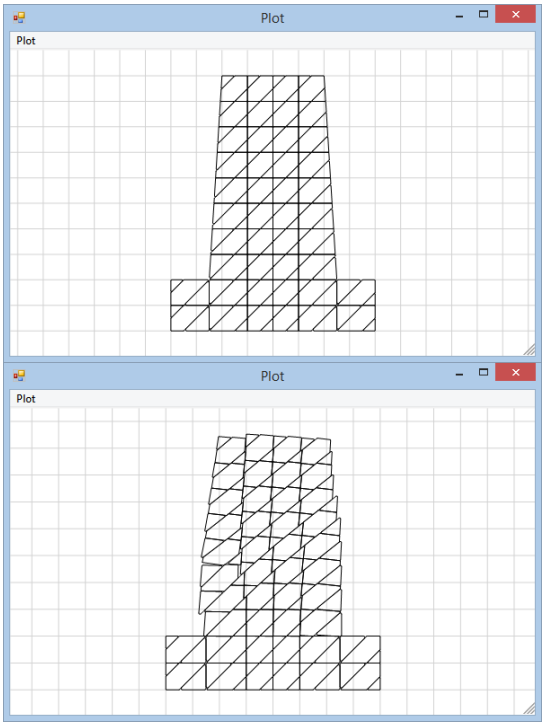
همچنین می‌توان دریافت که با افزایش ناپایداری،

افزایش جابه‌جایی بلوک‌ها، افزایش تغییرات در فهرست تماس‌ها و درنتیجه، افزایش سهم گام‌های به روزآوری در اجرای الگوریتم، کارایی الگوریتم PISU در بهینه‌سازی الگوریتم ISU افزایش یافته‌است.

در مورد تعداد بلوک‌های کم، زمان اجرای الگوریتم بهینه‌شده بیشتر از زمان اجرای الگوریتم ISU است. در این تعداد بلوک کم، افزایش ناپایداری حاصل از افزایش شیب شیروانی و افزایش به‌روزآوری‌ها، از اثر سوء سربار موازی‌سازی الگوریتم کاسته شده و زمان الگوریتم بهینه‌شده به الگوریتم ISU نزدیک شده‌است.



شکل (۳٤) بالا) زمان اجرای الگوریتم‌های ISU و PISU و پایین) نسبت زمان اجرای الگوریتم PISU به زمان اجرای الگوریتم ISU به ازای تغییرات شیب دیواره



شکل (۳۵) خروجی گرافیکی برنامه‌ی DA2 برای مساله‌ی حل‌شده: بالا) مدل قبل از شبیه‌سازی، و پایین) مدل پس از ۳ ثانیه شبیه‌سازی

بنا بر مطالب پیش‌گفته، الگوریتم PISU در محدوده‌ی مسائل مکانیک سنگ، الگوریتمی کارا و بهینه در مقایسه با الگوریتم ISU ارزیابی می‌گردد. شکل ۳۵، خروجی گرافیکی نرم‌افزار DA2 است که مدل را در ناپایدارترین حالت (بیشترین زاویه‌ی دیواره)، قبل و بعد از شبیه‌سازی، نشان می‌دهد.

٦- نتیجه‌گیری

در میان الگوریتم‌های کشف تماس، الگوریتم‌های ISU و DESS بسیار شناخته شده‌اند. الگوریتم‌های کشف تماس متعددند، اما الگوریتم‌هایی که در اینجا به آنها پرداخته شده است، الگوریتم‌هایی است که نسبت به شکل ذره‌ها حساس نیستند و پس برای تحلیل عددی مسائل مکانیک سنگ مناسب است. مسائل مکانیک سنگ بیشتر مسائلی هستند که در آنها مدل مساله دارای تراکم بالای بلوک‌ها، همسایگی

تماس کوچک، تفرق زیاد اندازه‌ی بلوک‌ها و جابه‌جایی‌های بسیار کم است.

طبق نتایج زمان اجرای به دست آمده، الگوریتم ISU، برای مسائل مکانیک سنگ، کمینه دو برابر سریع‌تر از الگوریتم DESS عمل می‌کند و طبق حساسیت‌سنجی‌های به عمل آمده، الگوریتم ISU حساسیت کمتری نسبت به تغییرات پارامترهای مساله نشان می‌دهد.

الگوریتم‌های ISU و DESS دارای الگوهای مشابهی است. هر دو الگوریتم دارای دو گام اصلی مرتب‌سازی و به‌روزآوری‌اند. گام‌های مرتب‌سازی و به‌روزآوری الگوریتم ISU از نظر زمانی که صرف می‌کنند به هم نزدیک هستند و این امر پیاده‌سازی این دو گام را به صورت موازی توجیه‌پذیر می‌کند. همچنین الگوریتم ISU، باوجود آنکه در پیاده‌سازی سنتی‌اش مراحل مرتب‌سازی و به‌روزآوری را ادغام کرده‌است، اما با اصلاحی کوچک و مقدار کمی سربار حافظه و زمان به‌راحتی می‌توان این مراحل را از هم جدا کرد. سپس برای انجام این گام‌ها به صورت موازی اقدام نمود. این موازی‌سازی گام‌های مرتب‌سازی و به‌روزآوری در افزایش کارایی بسیار مؤثر است.

مطابق نتایج ارائه شده، نشان داده شده‌است که روش بهینه‌سازی پیشنهاد شده در این مقاله توانسته‌است سرعت اجرای الگوریتم ISU را برای مساله نمونه‌ی مورد بررسی، تا ۲۰ درصد افزایش دهد.

۷- مراجع

[1] Han, K., Feng, Y.T. and Owen D.R.J., (2007), "Performance Comparison of Tree-Based and Cell-Based Contact Detection Algorithms," Engineering Computations, Internation Journal for Computer-Aided Engineering and Software, Vol. 24, No. 2, pp. 165-181.

[2] Perkins, E., Williams, J. R., (2001), "A fast contact detection algorithm insensitive to object sizes."

[3] Muth, B., Mueller, M.K., Eberhard, P. and S. Luding, (1987), "Contacts between Many Bodies."

[4] Schinner, A., (1999), "Fast algorithms for the simulation of polygonal particles," Granular Matter 2—Springer-Verlag.

[5] Schinner, A., (1995), "Numerische Simulationen fuer granulare Medien," Master's thesis, University of Regensburg.

[6] Baraff, D., Rigid Body Simulation, (1993), "An Introduction to Physically Based Modeling", ACM Siggraph, pages H1-H68.

[7] Sedgewick, R., (2011) "Algorithms," Addison-Wesley, 4th Edition, pp. 250-252.

[8] Miller, R., Boxer, L., (2012), "Algorithms: Sequential & Parallel—A Unified Approach," Cengage Learning, 3rd Edition.

[9] Kleinman, S., Shah, D., Smaalders, B., (1996), "Programming with Threads," SunSoft Press.

Evaluation and Improvement of Contact Detection Algorithms for Using in DEM in Rock Mechanics

H.R. Paseh[1], M. Yazdani[2*], M. Sharifzadeh[3]

1- Ph.D. Student of Civil Engineering, Soil and Foundation Mechanics, Tarbiat Modares University
2- Assistant Professor of Civil Engineering, Soil and Foundation Mechanics, Tarbiat Modares University
3- Associate Professor of Mining Engineering, Rock Mechanics, Amirkabir University of Technology

myazdani@modares.ac.ir

Abstract:
Discrete Element Method (DEM) is a numerical method for computing the motion and effect of a large number of small particles. It is a very common method to solve rock mechanics problems, since it can solve problems containing particles in contact with complicated geometries efficiently. Contact detection is the most time consuming (so the most significant) part of DEM-based problem solving methods.

In this article, authors, with the goal of implementing a numerical hydro-mechanical software to analyze and solve DEM rock mechanics problems (DA2), studied and investigated algorithms able to solve contact detection problem.

The most algorithms designed to find contacts (contact detection algorithms), lie in two classes: 1) algorithms based on bounding boxes, and 2) algorithms based on hashing.

The bounding box idea helps to simplify the contact detection problem and to prevent dealing with particle shapes by enveloping the whole particle in a shape (generally a rectangle or an ellipse) which is easy to check for finding overlaps. Since overlaps of bounding boxes may not directly result in contacts between particles, further checks are needed. In the former class, there are two well-known published algorithms, both based on sorting bounding boxes' extents, able to find contacts between generally shaped particles in a fast and efficient way: incremental sorting and updating (ISU) algorithm, and double-ended spatial sorting (DESS) algorithm. Hashing algorithms are generally appropriate for particles with uniform sizes. Since rock mechanics problems mostly contain models constituted of blocks with non-uniform sizes, hashing algorithms are not utilized for solving them.

In this article, ISU and DESS algorithms along with direct checking (DC) method are compared for their running time results to find the most appropriate (i.e. the fastest) algorithm to find contacts between rock blocks. For this purpose, algorithms were implemented by DA2 software, then, ran in the same environment and for same commonplace geomechanical problems with varying model parameters, like number of blocks, block size variation, angle of discontinuities and friction angle, and compared for their running time results. Results shows that ISU algorithm compared to DESS algorithm gives better/lower running time (ISU is at least twice as fast as DESS), i.e. more performance, and shows less sensitivity to model parameters. Also, ISU algorithm consumes less memory and it is simpler to implement.

In the end, for further improvement of performance of ISU algorithm, delayed updating and parallelization solutions are offered. Delayed updating is a common way to optimize algorithms containing two phases of processing and updating. In order to apply delayed updating and parallelization to ISU algorithm, a solution is presented to separate sorting and updating steps. Then, parallelization is applied. Results show that using these techniques can increase the performance of ISU algorithm by 20%.

Keywords: Rock Mechanics, Numerical Analysis, Discrete Element Method, DA2 Software, Contact Detection Algorithm