דו"ח סופי פרויקט גמר

1 פתיחה

1.1 מכללה: מכללת אפקה.

1.2 מחלקה: מדעי המחשב.

1.3 שם הפרויקט: Smart time table

1.4 מגישות:

a. לירז אוחיון 315872895

b. ענת קמינסקי 303050512

1.5 מנחה: ד"ר דינה גורן-בר.

1.6 תאריך הגשה: 14/04/23

2 תודות

ברצוננו להודות לד"ר דינה גורן-בר על האמון, העצמאות והלמידה במהלך ביצוע הפרויקט הסופי.

תודה לעומר נחשון על התמיכה והסיוע לאורך הסמסטר.

תודה למכללת אפקה שאפשרה לידע שנצבר במהלך הלימודים לספק לנו את הכלים לביצוע פרויקט זה.

3 תוכן עניינים

4 תקציר

תזמון מערכת שעות של סטודנט היא בעיה קומבינטורית מורכבת. תהליך התזמון דורש זמן רב וחשוף לטעויות אנוש היכולות להוביל לאי סדר.

סטודנט הנאלץ לבנות מערכת שעות עצמאית, נדרש ממנו לעשות בדיקה אישית עבור צרכיו טרם בניית המערכת האקדמית. דבר זה לוקח הרבה זמן, עלול להוביל לטעויות אנוש, כתוצאה מכך ישנה השפעה על מצבו האישי והאקדמי.

בפרויקט זה מימשנו אלגוריתם PSO בשילוב אילוצים אישיים של סטודנט בכדי לקבל מערכת שעות אופטימלית המותאמת אישית לצרכיו של הסטודנט.

Scheduling students' schedules is a complex combinatorial problem. The scheduling process takes a lot of time and is prone to human error that can lead to confusion.

Students who are forced to create independent time systems need to conduct a personal examination of their own needs before establishing an academic system. It takes a lot of time and can lead to human error, which has an impact on his personal and academic situation.

In this project, we implemented the PSO algorithm combined with the individual constraints of the students to obtain an optimal system of class hours tailored to the needs of the students.

5 מבוא

5.1 המוטיבציה:

כסטודנטיות בשנה ג', נאלצנו לבנות מערכת שעות ידנית דבר הדורש מאיתנו הרבה זמן ומחשבה.

היינו צריכים להתחשב בתנאי הקדם, שעות הקורסים, לוח הבחינות, כמובן גם באילוצים האישיים של כל אחד ואחת מאיתנו.

לכן בחרנו לייצר – אלגוריתם שיודע לבנות מערכת שעות אופטימלית בהינתן אילוצים אישיים של סטודנט/ית.

5.2 הגדרת הבעיה:

למוסדות אקדמאים, לוקח הרבה מאוד זמן לתזמן ידנית את כלל השיעורים שסטודנט צריך לקחת בהתאם לזמינות המרצים והכיתות. לפיכך, כוונת ההצעה לאוטומציה של תהליך זה היא לענות על צרכים אמיתיים, במטרה העיקרית לצמצם את הזמן הנדרש להשלמתו ביעילות.

עפ"י פונקציות מתמטיות, נחפש את הערכים של מודלים כאלו תוך שימוש בטכניקות אופטימיזציה.

באופן כללי, פתרון אתגרים כאלה וקבלת פתרונות אופטימליים מדויקים הם בלתי פתירים מבחינה חישובית. לכן, בניית מערכת שעות של מוסד אקדמי היא דוגמה לבעיה לא פולינומית (NP) קשה. אלו בעיות ללא פתרונות יעילים במיוחד

5.3 מטרות ויעדים :

המערכת תעסוק בבעיית אופטימיזציה של בניית מערכת שעות בהינתן אילוצי מערכת ואילוצים אישיים של סטודנט.

מערכת השעות שתתקבל תיעשה ע"י פונקציית סכימה של סך ההפרות של כל אילוץ שהתקבל פר שיבוץ נוכחי, כאשר בכל איטרציה יבוצע שינוי בשיבוץ עד אשר נקבל סך ערך הפרות מינימלי עם מערכת שעות אופטימלית ללא התנגשויות.

6 סקירה ספרותית

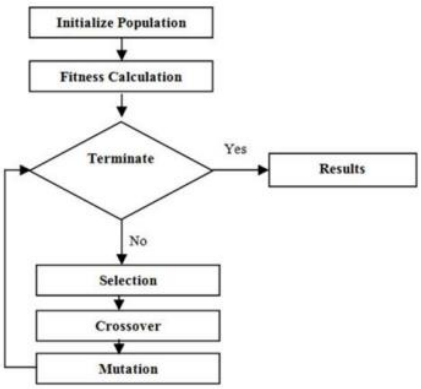
נסקר מספר אלגוריתמי אופטימיזציה שניתן ליישם כפתרונות אפשריים לבעיית בניית מערכת שעות של הקורסים של סטודנטים במוסד אקדמי המטרה העיקרית היא לספק פתרונות שונים לסוגיית לוח הזמנים. אחת הגישות הפופולריות ביותר **[1]** היא רכישת פתרונות המבוססים על אלגוריתמים גנטיים (GAs) בשל מידת ההקבלה החישובית הגבוהה שלהם וביצועים חישוביים משופרים.

GA היא אסטרטגיית אופטימיזציה של פרמטרים החוזרת על פני אוכלוסייה, מחפשת את ההתאמה הטובה ביותר לבעיה עד שמשיגה פתרון אופטימלי. בפתרון בניית מערכת השעות GAs היא אסטרטגיה שניתן להשתמש בה כדי לפתור את התנאים הללו ביעילות. היא מבוססת על עקרונות הגנטיקה והברירה הטבעית, כשהרעיון המרכזי הוא בחירת הפרטים המתאימים ביותר באוכלוסייה, אשר לאחר מכן מתחברים עם אחרים או עוברים מוטציה לצורות חדשות כדי ליצור קבוצות חדשות. GAs מיושמים במיוחד לבעיות אופטימיזציה מורכבות, שהן אתגרים בעלי פרמטרים או מאפיינים שונים שצריך לשלב בחיפוש אחר הפתרון הטוב ביותר, ובו בזמן, לא ניתן לייצוג מתמטי.

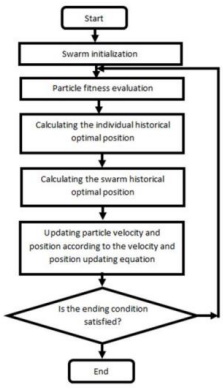
יתרון אחד של שימוש בGA הוא בכך שהוא יכול למצוא פתרונות לבעיות שקשה או בלתי אפשרי לפתור בשיטות מסורתיות. יתרון נוסף הוא שניתן להשתמש באלגוריתמים גנטיים כדי לפתור בעיות שיש להן מטרות או אילוצים מרובים.

החסרונות של GA הם: איטיות, יכולים להיות יקרים ליישום, קשים להבנה, קשים לניפוי באגים ויכול להיות קושי לייעל אותם.

איור 1ממחיש את תרשים הזרימה של אלגוריתם GA, המהאוכלוסיה הראשונית ועד כאשר מתקבלת תוצאה מתאימה.



איור 1.תרשים זרימה של GA [41].



איור 2. תרשים זרימה של אלגוריתם PSO.

אלגוריתם נוסף שניתן בעזרתו ליישם פתרון אפשרי לבעיית בניית מערכת שעות, הינו אלגוריתם גנטי אדפטיבי AGA (מסתגל). **[2]**

אלגוריתם זה הוא איטרטיבי, ומאותחל באמצעות יצירה של פתרונות אקראיים, ולאחר מכן באיטרציות הבאות פתרונות חדשים נוצרים באמצעות שילובים של הפתרונות מהאיטרציה הקודמת.

למעשה כל איטרציה היא דור של פתרונות, שמתוכו "נולד" דור הפתרונות הבא. בכל דור, כל פתרון נבחן על ידי פונקציית כשירות (fitness) שנועדה לקבוע באיזו מידה הפתרון עונה על האילוצים שהוגדרות.

המאמר מגדיר 2 סוגים של אילוצים בבניית המערכת: אילוצים קשים ואילוצים רכים. כל פיתרון נדרש להימנע לחלוטין מהפרה של אילוצים קשים. הפרה של אילוצים רכים מותרת, אבל ככל שיש יותר מהם, איכות הפתרון, שנמדדת באמצעות פונקציית הכשירות, יורדת.

פונקציית הכשירות מוגדרת כך:

$$f = \frac{1}{soft\ constraints\ violation + 1}$$

הפונקציה מחזירה מספר בין 0 ל – 1, כאשר ככל שהערך שלה מתקרב ל – 1 איכות הפיתרון טובה יותר.

לשם יצירת הפתרונות בכל דור (איטרציה), האלגוריתם משתמש בשני סוגים של פעולות על הפתרונות הקיימים:

- הכלאה (Crossover): יצירה של פתרון חדש באמצעות קומבינציה של פתרונות קיימים
- מוטציה: הכנסת מרכיב של שינוי רנדומלי לפתרון. בתחילה, השימוש במוטציה מוגבל לחלק קטן מהפתרונות, ובדור הבא נבחנת הכשירות של הפתרונות שעברו מוטציה. אם הכשירות שלהם עלתה, התדירות של המוטציה המסוימת שהם עברו גדלה ומתרחבת לאחוז גדול יותר של הפתרונות בדור הבא, ואם הכשירות ירדה בעקבות המוטציה התדירות שלה יורדת.

בכל דור בודקים האם האלגוריתם השיג התכנסות, כאשר התכנסות מוגדרת כמצב שבו הכשירות הכי גבוהה בכל דור זהה.

תרשים זרימה המתאר את האלגוריתם:

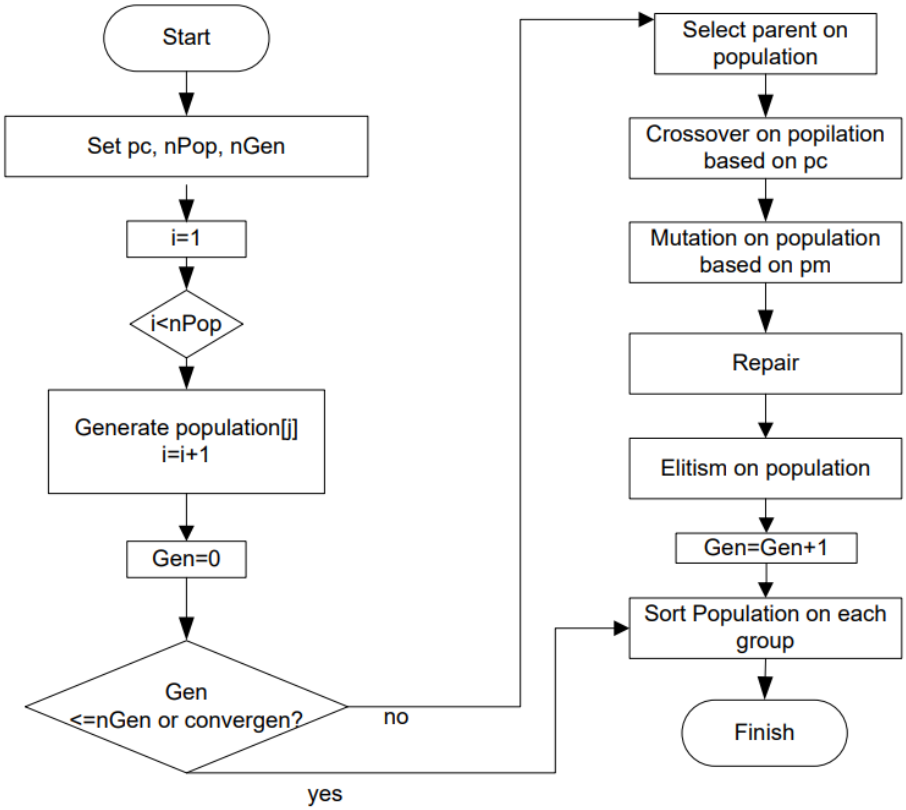


Figure 1. The genetic algorithm flowchart

בהמשך, המאמר מציג תוצאות ניסוי שבו נבנתה מערכת שעות באמצעות אלגוריתם גנטי אדפטיבי, ומראה שהאלגוריתם מצליח להשיג תוצאות יותר טובות מאלגוריתם גנטי מקורי: כשירות מקסימלית של 0.054, לעומת 0.045 עבור אלגוריתם גנטי מקורי.

בהשוואה לGA הכללי, היתרונות של AGA שהוא מהיר יותר, בעל חוסן גבוה יותר ועם ביצועים טובים יותר. חיסרון שהוא אולי יכול להיות יותר יקר ליישום.

בנייה של מערכת שעות היא בעיה של הקצאת זמנים שעונה על אילוצים בצורה מספקת אבל לא אופטימלית. ישנו אלגוריתם יעיל **[3]** PSO (Particle Swarm Optimization) שבא לפתור בעיות בדידות של בניית מערכת שעות.

PSO הינו אלגוריתם שמבוסס על התנהגות של נחילים בטבע (ציפורים, דגים וכו'). באלגוריתם זה ישנו מרחב פתרונות פוטנציאלים שנקראים חלקיקים. PSO מאותחל באמצעות קבוצה של פתרונות אקראיים, ואז הוא מחפש פתרון אופטימלי באמצעות שינוי הפתרונות תוך שימוש במידע מפתרונות אחרים ממרחב הפתרונות הפוטנציאלים. האלגוריתם מדמה תנועה של פרטים בתוך נחיל, בכך שכל פתרון מקבל "מהירות" ו"מיקום", ושני הפרמטרים מעודכנים בכל איטרציה כתלות במידע שמתקבל מפתרונות אחרים.

כל הפתרונות הפוטנציאלים עוברים הערכה של התאמתם לפתרון הבעיה ע"י פונקציית תאימות (fitness function).

השיטה המוצעת לשימוש בPSO עבור בניית מערכת שעות משנה בכל איטרציה את השיבוץ במערכת כך שבסופו של התהליך מתקבל שיבוץ אופטימלי.

המאמר מציג תוצאה ניסיונית של יישום האלגוריתם עבור בניית מערכת שעות של מבחנים. האיכות של כל פתרון נמדדת על ידי פונקציה שסוכמת את כל ההפרות של האילוצים שקיימים עבור כל סטודנט. כל אילוץ מקבל משקל, שמגדיר את ה"עונש" שניתן עבור הפרה שלו.

התוצאות מראות ש – PSO הוא אלגוריתם יעיל עבור בעיות של בניית מערכת שעות, כאשר נבנו באמצעותו מערכות שעות שונות ללא התנגשויות.

היתרונות העיקריים של אלגוריתם ה-PSO: רעיון פשוט, יישום קל, חוסן בשליטה בפרמטרים ויעילות חישובית בהשוואה לאלגוריתם מתמטי וטכניקות אופטימיזציה היוריסטיות אחרות.

החסרונות של אלגוריתם PSO הוא שקל ליפול לאופטימום מקומי במרחב בעל מימד גבוה ויש לו קצב התכנסות נמוך בתהליך האיטרטיבי.

בגרף המצורף ניתן לראות את ירידת ערך ה"עונש" עבור פתרונות פוטנציאליים שונים עם התקדמות האלגוריתם:

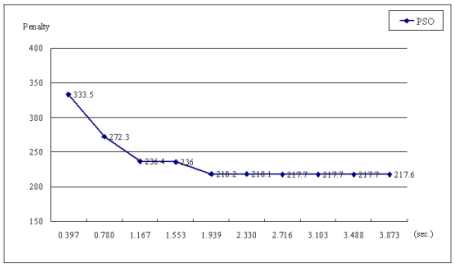


Figure 2. A student takes 11 exams at most.

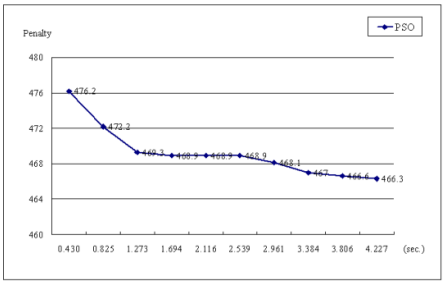


Figure 4. A student takes 15 exams at most.

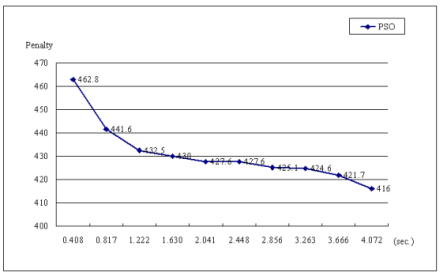


Figure 3. A student takes 13 exams at most.

הפניה למקורות מידע:

2. A Review of Optimization Algorithms for University Timetable Scheduling

https://www.etasr.com/index.php/ETASR/article/view/3832

3. Adaptive Genetic Algorithm for High School Time-Table

https://www.proquest.com/docview/2570431331?pq-origsite=primo

4. Timetable Scheduling Using Particle Swarm Optimization

https://www.researchgate.net/publication/224647262_Timetable_Scheduling_Using_Particle_Swarm_Optimization

7 סקר שוק

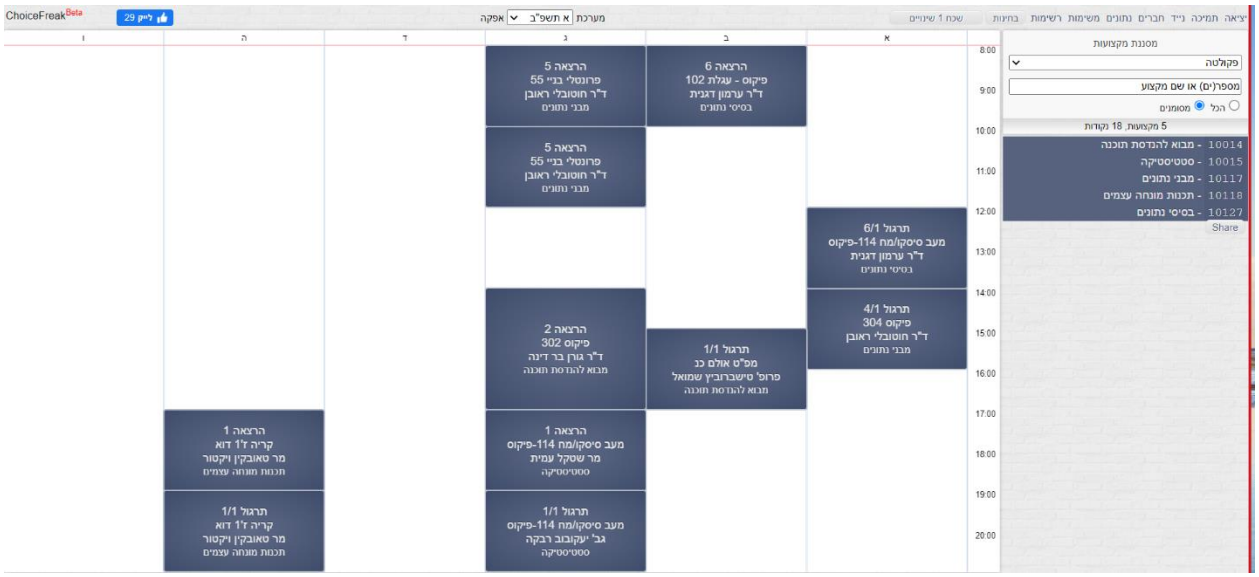
שם	choice freak	trent university	Smart Time Table
האם נדרשת התערבות של המשתמש?	כן. • בדיקה עצמאית של תנאי הקדם. • בחירה ידנית של שעות הקורסים מבין האופציות.	כן. • בדיקה עצמאית של תנאי הקדם. • בחירה ידנית של שעות הקורסים מבין האופציות.	לא.
האם יודע לבנות לבד את המערכת שעות?	לא.	כן. ללא התחשבות באילוצים אישיים.	כן.
האם יש התחשבות באילוצים האישיים?	לא.	לא.	כן.

<u>היתרון התחרותי שלנו:</u>

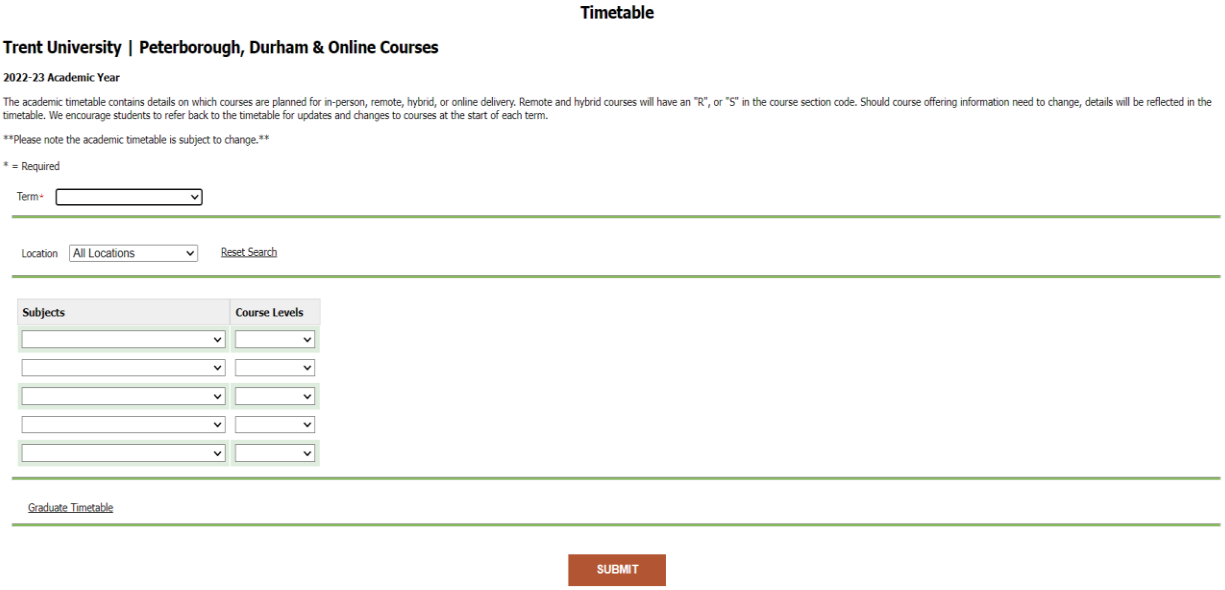
האלגוריתם שלנו יידע לבנות בצורה עצמאית מערכת שעות בשילוב התחשבות אילוצים אישיים של סטודנט.

8 חלופות

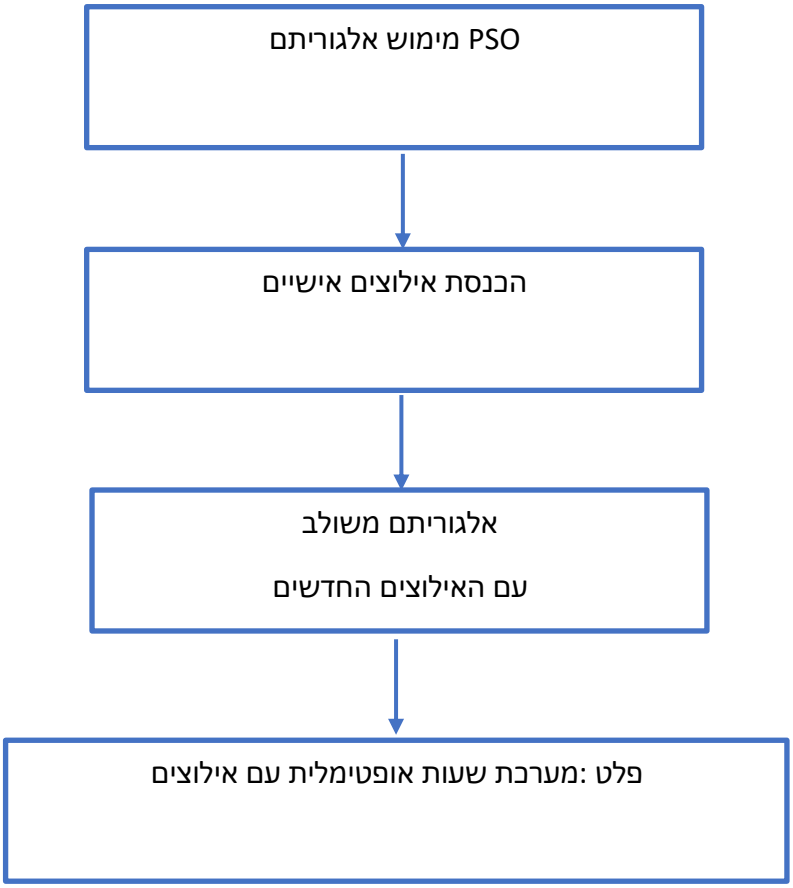
choice freak"" - מראה את האופציות של הקבוצות השונות הקיימות בקורס כלשהו שסטודנט בוחר. אמנם זה מקל על הסטודנט, אך הסטודנט בעצמו נדרש לבדוק התאמה של מספר "קומבינציות" עד אשר מוצא משהו שמתאים לו ולאילוציו. האלגוריתם אינו יודע לשבץ עצמאית ודורש מגע יד אדם על מנת לבצע זאת.



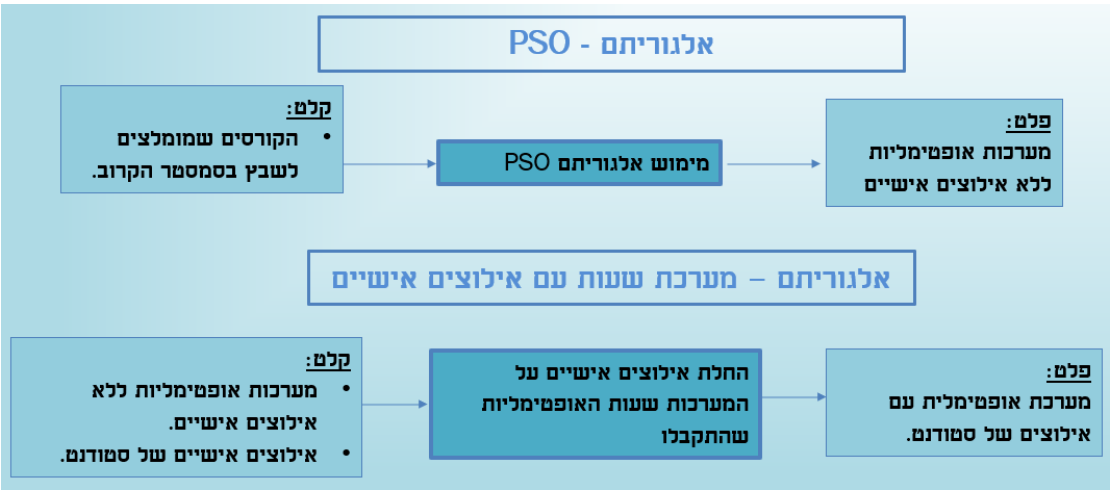
trent university"" - מאפשרת לסטודנט להזין ידנית קורסים רצויים עבור מערכת השעות שלו, אך בתוצאה הסופית האלגוריתם אינו יודע לבנות לבד מערכת שלמה מותאמת אישית על פי אילוצים. לא יעיל והרבה "עבודה שחורה".



9 ארכיטקטורה



<u>תיאור המודל:</u>

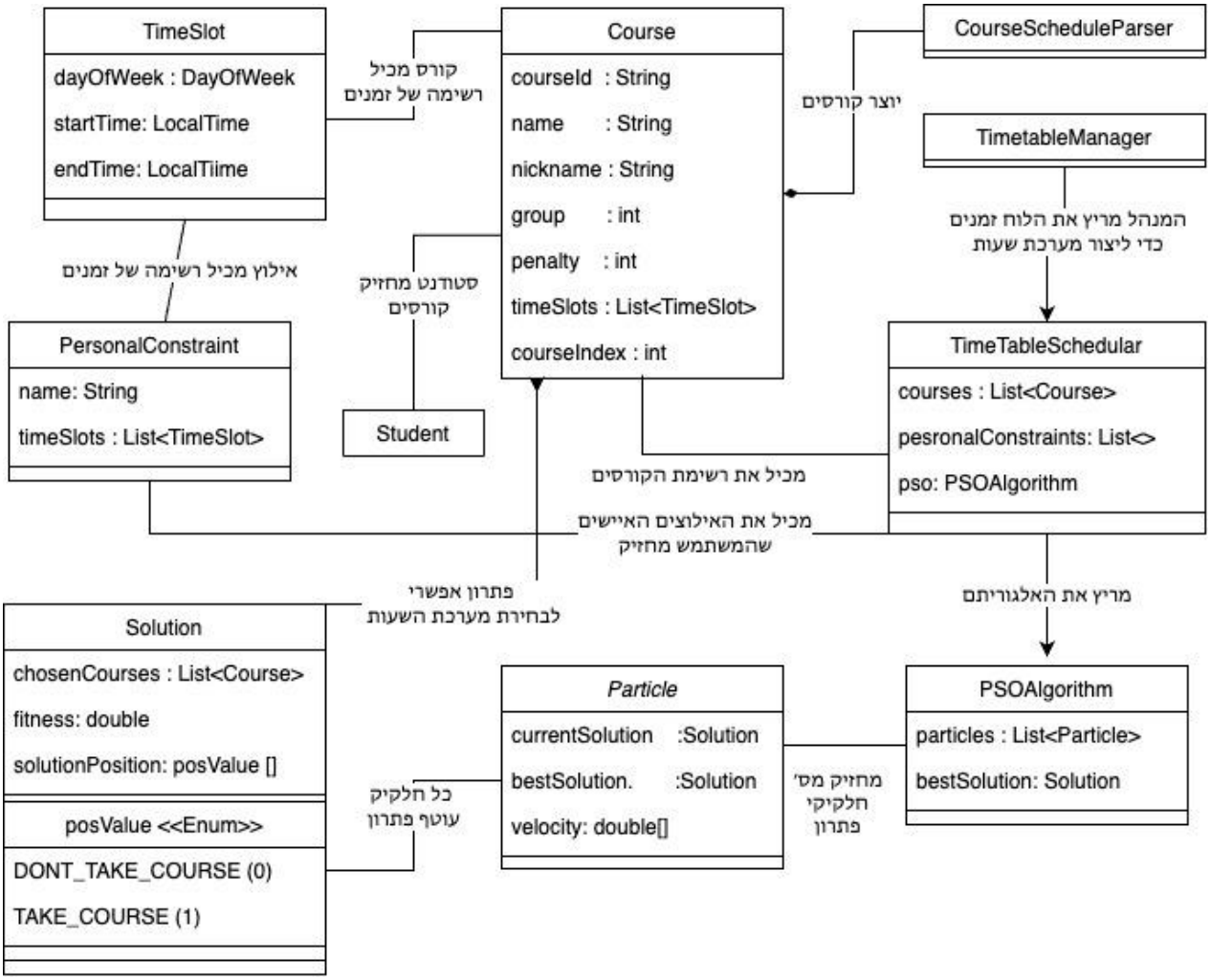


10 תכן מפורט

10.1 כללי

- מימוש OOP בשפת JAVA.
- נתונים הנוגעים ל"קורס" יילקחו מתוך מסלול מדעי המחשב כפי שהוא מפורסם באפקה.
- טעינת הנתונים בפורמט json, ספציפית מקורס course.json
- אילוצי הסטודנט יוזנו ע"י המשתמש.

10.2 תרשים מחלקות



10.3 פירוט מחלקות

10.3.1 מחלקת קורס - **TimeSlot**

מחלקת TimeSlot, מייצגת משבצת זמן בהקשר של לוח זמנים או קורסים. למחלקה יש שלושה מאפיינים: יום בשבוע, שעת התחלה ושעת סיום.

בנוסף, המחלקה כוללת פעולות שונות:

- בנאי שמקבל יום בשבוע, שעת התחלה ושעת סיום.
- פעולות לקבלת והגדרת יום השבוע, שעת התחלה ושעת סיום.
- פעולת toString המחזירה תיאור של המשבצת הזמן כמחרוזת.
- פעולה בשם overlaps שמקבלת משבצת זמן אחרת ובודקת אם יש חפיפה בין שתי המשבצות.

ניתן להשתמש במחלקת TimeSlot כדי לייצג משבצות זמן ביישומים כמו מערכות ניהול לוחות זמנים, מערכות רישום לקורסים או מערכות ניהול משאבים.

10.3.2 מחלקת קורס - **Course**

Course , מייצגת קורס. יש לה מאפיינים כמו שם הקורס, מזהה הקורס, קבוצה, משבצות זמן, עונש וכינוי. המחלקה גם מחזיקה רשימה סטטית של כל המופעים של הקורסים בשם allCourses.

המחלקה מספקת שיטות להוספת משבצות זמן לקורס, בדיקה אם קורס מתנגש עם קורס אחר או אילוצים אישיים והחזרת פרטים שונים אודות הקורס.

בנוסף, למחלקה יש מספר שיטות סטטיות לחישוב מספר עימותי הלוח זמנים של כל הקורסים, העונש הכולל של כל הקורסים ושיטות לקבלת כל הקורסים או קורס לפי האינדקס שלו.

ניתן להשתמש במחלקת Course כדי לעצב קורסים ולוחות זמנים שלהם ביישומים כמו מערכות ניהול לוחות זמנים, מערכות רישום לקורסים או אלגוריתמים למילוי מחלקות.

10.3.3 מחלקת קורס - **Student**

מחלקת Student, מייצגת סטודנט. למחלקה יש שני מאפיינים: שם הסטודנט ורשימת הקורסים שניתן ללמוד בסמסטר הבא.

בנוסף, המחלקה כוללת פעולות שונות:

- בנאי שמקבל שם של סטודנט.
- פעולות לקבלת והגדרת שם הסטודנט.
- פעולה להוספת קורס לרשימת הקורסים של הסטודנט.
- פעולה לקבלת רשימת הקורסים של הסטודנט.
- פעולת toString המחזירה נתונים אודות הסטודנט ורשימת הקורסים שלו.
- ניתן להשתמש במחלקת Student כדי לייצג סטודנטים ביישומים כמו מערכות ניהול רישום לקורסים, מערכות ציונים או מערכות ניהול לוחות זמנים.

10.3.4 מחלקת קורס - **CourseScheduleParser**

מחלקת CourseScheduleParser, מיועדת לפרסר קובץ שמכיל מידע אודות קורסים וליצירת רשימת מופעים מסוג Course המיוצגים במערכת. המחלקה מכילה פונקציה בשם ParseCourse שמקבלת נתיב לקובץ הקורסים.

בתוך פונקציה ParseCourse:

- מייבאת את הספרייה JSONParser לפרוסת קבצי JSON.
- יוצרת רשימת Course חדשה לשמירת הקורסים המפוענחים.
- פותחת קובץ הקורסים לקריאה ומפענחת אותו לפי תבנית ה-JSON.
- מעבר על כל הקורסים והקבוצות בתוך הקובץ ויצירת אובייקטים מסוג Course עבורם.
- הוספת משבצות זמן לכל אובייקט Course.
- הוספת אובייקט ה-Course לרשימת הקורסים המפוענחים.
- זריקת חריגות במקרים של שגיאות קריאת קובץ או פרוסת ה-JSON.

בסיומה של פונקציה ParseCourse, היא מחזירה את הרשימה של אובייקטים מסוג Course שנוצרו מהקובץ. ניתן להשתמש במחלקת CourseScheduleParser כדי לטעון קובץ קורסים וליצור אובייקטים מסוג Course מתוך המידע בקובץ.

10.3.5 מחלקת **PersonalConstraint**

מחלקת PersonalConstraint מיועדת לייצוג מגבלות אישיות של המשתמש בקשר ללוח זמנים. כל מגבלה אישית מכילה שם ורשימת אובייקטים מסוג TimeSlot המייצגים את המשבצות הזמן בהם קיימת המגבלה האישית.

המחלקה מכילה:

- בנאי שמקבל את שם המגבלה ורשימת משבצות הזמן.
- פונקציה getTimeSlots שמחזירה את רשימת המשבצות הזמן של המגבלה האישית.
- פונקציה overlapsWith שמקבלת שעה ויום ובודקת אם קיימת מגבלה אישית באותו זמן.
- פונקציה toString שמחזירה מחרוזת המייצגת את המגבלה האישית ואת רשימת המשבצות הזמן שלה.
- פונקציה getName שמחזירה את שם המגבלה האישית.

מחלקת PersonalConstraint משמשת לייצוג וניהול מגבלות אישיות של המשתמש בתכנון לוח זמנים, כגון זמנים בהם המשתמש אינו יכול לקיים מפגשים או קורסים.

10.3.6 מחלקת **Solution**

מחלקת Solution בשפת התכנות ג'אווה מייצגת פתרון עבור בעיה של תכנון לוח זמנים. המחלקה מכילה רשימה של קורסים שנבחרו, פונקציה חישוב המדד (fitness), ומערך של ערכים בינאריים המייצגים את נוכחות הקורסים בפתרון. מטרת הבעיה היא למצוא פתרון שמביא לערך מינמאלי של מדדfitness)) כך נמנעים מהתנגשויות בין הקורסים ומקיימים כמה שיותר קורסים בהתחשבות ב"בעונש" (penalty).

1. מחלקה פנימית **posValue** מייצגת את האפשרויות הבינאריות בפתרון: לא לקחת את הקורס (0) או לקחת אותו (1). המחלקה הזו מכילה שני ערכים אפשריים עבור הפתרון ופונקציה לקבלת הערך שלהם.

<u>מחלקת Solution כוללת את הפונקציות הבאות:</u>

- בנאים שיצירת פתרון רנדומלי בהתחשב בתנאים הנדרשים.
- בנאי העתקה ליצירת פתרון חדש מבוסס על פתרון קיים.
- פונקציות לקבלת ושינוי ערך המיקום במערך הפתרון.
- פונקציה לבדיקה אם ניתן להוסיף קורס מבלי לגרום להתנגשות.
- פונקציה לחישוב המדד (fitness) של הפתרון.
- פונקציה להשוואת שני פתרונות לפי המדד שלהם.
- פונקציה לבדיקת שוויון בין שני פתרונות לפי המיקום שלהם.
- פונקציה להדפסת הפתרון לפי מבנה לוח זמנים קיים.

- פונקציה להדפסת הפתרון עם מגבלות אישיות בנוסף לקורסים.

בקצרה, מחלקת Solution מייצגת פתרון של לוח זמנים בהתחשב בתנאים הנדרשים ומטרתה היא למצוא פתרון בעל הערך המינימאלי של מדד ה fitness עם ההתנגשויות מינמאליות בין הקורסים ועם כמה שיותר קורסים. לה יש מחלקה פנימית posValue שמייצגת את האפשרויות הבינאריות בפתרון.

10.3.7 מחלקת **Particle**

זוהי מחלקת Particle המייצגת חלקיק באלגוריתם PSO (Particle Swarm Optimization). המחלקה מכילה את התכונות הבאות:

- currentSolution - הפתרון הנוכחי של החלקיק.
- bestSolution - הפתרון האישי הטוב ביותר שהחלקיק מצא עד כה.
- globalBestSolution - הפתרון הכולל הטוב ביותר שהחלקיק ראה עד כה.
- velocity - מערך שמכיל את המהירות של החלקיק בכל צעד.
- random - מופע של מחלקת Random ליצירת מספרים אקראיים.

המחלקה מכילה גם פונקציות כמו:

- updateVelocity - מעדכנת את המהירות של החלקיק בהתאם לפתרון האישי הטוב ביותר והפתרון הכולל הטוב ביותר.
- updatePosition - מעדכנת את המיקום של החלקיק, כלומר את הפתרון הנוכחי, על פי המהירות העדכנית.
- updateGlobalBestSolution - מעדכנת את הפתרון הכולל הטוב ביותר של החלקיק, אם הפתרון החדש טוב יותר.
- getBestSolution - מחזירה את הפתרון האישי הטוב ביותר של החלקיק.
- printPosition ו-printPositionValues - מדפיסות את המיקום הנוכחי של החלקיק ואת הערך הכולל שלו.

בנוסף, המחלקה מממשת את הממשק Comparable על מנת לאפשר השוואה בין חלקיקים לפי הערך הכולל של פתרונותיהם.

10.3.8 מחלקת **PSO Algorithm**

זוהי מחלקת PSOAlgorithm המייצגת את אלגוריתם PSO (Particle Swarm Optimization) עבור בעיה של מערכת שעות לקורסים. המחלקה מכילה את התכונות הבאות:

- קבועים לאלגוריתם, כולל מספר האיטרציות המרבי, גודל האוכלוסין, גורמים למידה, מהירויות מרביות ומינימליות וכו'.
- particles - רשימה של חלקיקים.
- bestSolution - הפתרון הטוב ביותר שהאלגוריתם מצא.

המחלקה PSO Algorithm מכילה גם פונקציות כמו:

- בנאי - מאתחל את כלל הפתרונות האקראיים.
- run - מבצע את האלגוריתם, מעדכן את המהירות והמיקום של כל חלקיק ומעדכן את הפתרון הטוב ביותר ברמה הכוללת.
- getGlobalBestSolution - מחזירה את הפתרון הטוב ביותר ברמה הכוללת מבין כל החלקיקים.
- updateGlobalBestSolution - מעדכן את הפתרון הטוב ביותר ברמה הכוללת של כל החלקיקים.
- getBestSolution - מחזירה את הפתרון הטוב ביותר שהאלגוריתם מצא.
- getAllSolutions - מחזירה רשימה של כל הפתרונות הטובים ביותר של כל החלקיקים באוכלוסין.

מטרת האלגוריתם היא למצוא את הפתרון הטוב ביותר לבעיה הנתונה.

10.3.9 מחלקת **TimetableScheduler**

זוהי מחלקת TimetableScheduler המייצגת מתכנן לוח זמנים לסטודנטים. המחלקה מכילה את התכונות הבאות:

- student - סטודנט שעבורו נבנה לוח הזמנים.
- courses - רשימה של קורסים שיכולים להיכנס ללוח הזמנים.
- personalConstraints - רשימה של מגבלות אישיות של הסטודנט.
- pso - אלגוריתם PSO למציאת הפתרון הטוב ביותר ללוח הזמנים.

המחלקה מכילה גם פונקציות כמו:

- בנאי - מאתחל את הסטודנט, קורסים ומגבלות אישיות מקבצים ומריץ את אלגוריתם ה-PSO.

- getBestSchedule - מחזיר את הלוח הזמנים הטוב ביותר שנמצא על ידי האלגוריתם.
- getTopSchedules - מחזיר את לוחות הזמנים הטובים ביותר במספר שנבחר.
- getAllCourses - מחזיר את כל הקורסים הקיימים במערכת.
- getPersonalConstraints - מחזיר את רשימת המגבלות האישיות של הסטודנט.
- addPersonalConstraint - מוסיף מגבלה אישית לרשימת המגבלות.
- removePersonalConstraint - מסיר מגבלה אישית מהרשימה לפי שם.
- getTopSchedulesWithConstraint - מחזיר את לוחות הזמנים הטובים ביותר במספר שנבחר עם התחשבות במגבלות אישיות.

המטרה של המחלקה היא להכין לוח זמנים מותאם אישית לסטודנט על פי הקורסים הזמינים

והמגבלות האישיות שלו. היא משתמשת באלגוריתם PSO כדי למצוא את הפתרון הטוב ביותר ללוח הזמנים, תוך שימוש במגבלות כדי להבטיח שהפתרון מתאים לדרישות הסטודנט. ניתן להוסיף או להסיר מגבלות ולקבל את הלוחות הזמנים הטובים ביותר עבור הסטודנט בהתאם לצרכים שלו.

10.3.10 מחלקת **TimetableManager**

מחלקת TimetableManager היא כניסת המשתמש בתוך מערכת התכנון של לוח הזמנים. המחלקה מגדירה את התפריט הראשי ואת התהליך של בחירת הפעולות השונות שהמערכת יכולה לבצע. היא מאפשרת למשתמש לבחור בין הצגת כל הקורסים, הצגת הלוח הזמנים הטוב ביותר, הצגת רשימה של הלוחות הזמנים הטובים ביותר והוספת או הסרת מגבלות אישיות. לאחר שהמשתמש בוחר פעולה, המחלקה מבצעת את הפעולה הרלוונטית ומציגה את התוצאות. המחלקה מקבלת את המידע מקובץ JSON שמכיל את הקורסים.

מחלקת TimetableManager גם מאפשרת למשתמש לבצע פעולות עם מגבלות אישיות, כולל הצגת רשימת המגבלות האישיות הקיימות להצגת לוחות הזמנים הטובים ביותר תוך קיבוע מגבלות אישיות. כמו כן, המחלקה מספקת פונקציות עזר להוספת והסרת הגבלות אישיות מהרשימה, כאשר המשתמש מזין את המידע הדרוש עבורם.

מחלקת TimetableManager המשתמשת במחלקת TimetableScheduler, שמטפלת בייעוץ לוח הזמנים עם קביעת הקורסים והמגבלות האישיות. מחלקה שמשתמשת בקלט מהמשתמש כדי לבצע פעולות על פי בחירת המשתמש.

לסיכום, מחלקת TimetableManager משמשת כממשק בין המשתמש לתכנון לוח זמנים. היא מאפשרת למשתמש לבחור בין מגוון פעולות כמו להציג קורסים, להציג לוחות זמנים טובים, להוסיף ולהסיר מגבלות אישיות, ולקבל לוחות זמנים מותאמים אישית תוך שימוש במגבלות אישיות.

10.4 פירוט התפריט הראשי

התפריט הראשי למשתמש

```
Welcome to the Schedule Manager!
Please enter your name: Liraz
Hello, Liraz!
Creating Your Schedule :) Please Wait...
Done :)

>>>Please choose an option:
1. Show all courses
2. Show best schedule
3. Show top 4 schedules

4. Add personal constraint
5. Remove personal constraint
6. Show personal constraints

7. Show best schedule WITH CONSTRAINT
8. Show top 4 schedules WITH CONSTRAINT

0. Exit
```

בעת נפרט ונראה דוגמאות לכל אפשרות של התפריט הראשי

10.4.1 להראות את כל הקורסים הקיימים - **Show all Courses**

לדוגמה,

```
################ START - All Courses #######################
Digital Logic Design(10007):
Nichname: 'Dig Logic'
Groups:
        Group 1
                THURSDAY 10:00-14:00
                MONDAY 08:00-10:00
        Group 2
                SUNDAY 15:00-19:00
                THURSDAY 10:00-12:00
        Group 3
                SUNDAY 17:00-20:00
                MONDAY 18:00-21:00

Introduction to Computer Science(10006):
Nichname: 'Intro CS'
Groups:
        Group 1
                MONDAY 17:00-19:00
                THURSDAY 14:00-17:00
                TUESDAY 11:00-12:00
```

Group 2
MONDAY 12:00-14:00
SUNDAY 12:00-16:00
Group 3
SUNDAY 15:00-18:00
MONDAY 09:00-12:00

Differential and Integral Calculus 1(90901):
Nichname: 'Dif&Calc 1'
Groups:
Group 1
WEDNESDAY 17:00-21:00
TUESDAY 18:00-20:00
Group 2
TUESDAY 08:00-10:00
WEDNESDAY 13:00-16:00
THURSDAY 09:00-10:00
Group 3
TUESDAY 17:00-21:00
SUNDAY 17:00-19:00

Linear Algebra(90905):
Nichname: 'Lin Alg'
Groups:
Group 1
THURSDAY 18:00-22:00
WEDNESDAY 08:00-10:00
Group 2
MONDAY 17:00-19:00
SUNDAY 15:00-18:00
THURSDAY 16:00-17:00
Group 3
MONDAY 10:00-14:00
WEDNESDAY 16:00-18:00

Discrete Mathematics(90906):
Nichname: 'Dis Math'
Groups:
Group 1
MONDAY 15:00-19:00
TUESDAY 13:00-15:00
Group 2
TUESDAY 10:00-14:00
WEDNESDAY 14:00-16:00
Group 3
TUESDAY 11:00-13:00
WEDNESDAY 09:00-12:00
MONDAY 10:00-11:00

################ END - All Courses #######################

10.4.2 להראות את הלו"ח זמנים הכי טוב - **Show best schedule**

לדוגמה,

```
Fitness: 1.4143988690240341E-8
Position: [001100010100100]
Schedule:
              |  SUN             |  MON             |  TUE             |  WED             |  THU             |
 8:00 -  9:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
 9:00 - 10:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
10:00 - 11:00 |                  |                  |                  |                  |  #1# Dig Logic   |
11:00 - 12:00 |                  |                  |                  |                  |  #1# Dig Logic   |
12:00 - 13:00 |  #2# Intro CS    |  #2# Intro CS    |                  |                  |  #1# Dig Logic   |
13:00 - 14:00 |  #2# Intro CS    |  #2# Intro CS    |  #1# Dis Math    |                  |  #1# Dig Logic   |
14:00 - 15:00 |  #2# Intro CS    |                  |  #1# Dis Math    |                  |                  |
15:00 - 16:00 |  #2# Intro CS    |  #1# Dis Math    |                  |                  |                  |
16:00 - 17:00 |                  |  #1# Dis Math    |                  |                  |                  |
17:00 - 18:00 | #3# Dif&Calc 1   |  #1# Dis Math    | #3# Dif&Calc 1   |                  |                  |
18:00 - 19:00 | #3# Dif&Calc 1   |  #1# Dis Math    | #3# Dif&Calc 1   |                  |   #1# Lin Alg    |
19:00 - 20:00 |                  |                  | #3# Dif&Calc 1   |                  |   #1# Lin Alg    |
```

10.4.3 להראות את ה-4 הלוחות זמנים הכי טובים - **Show top 4 schedule**

לדוגמה,

```
 ################ START - Top Solutions ####################

 ################ Solution 1 >:
Fitness: 1.4143988690240341E-8
Position: [001100010100100]
Schedule:
              |  SUN             |  MON             |  TUE             |  WED             |  THU             |
 8:00 -  9:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
 9:00 - 10:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
10:00 - 11:00 |                  |                  |                  |                  |  #1# Dig Logic   |
11:00 - 12:00 |                  |                  |                  |                  |  #1# Dig Logic   |
12:00 - 13:00 |  #2# Intro CS    |  #2# Intro CS    |                  |                  |  #1# Dig Logic   |
13:00 - 14:00 |  #2# Intro CS    |  #2# Intro CS    |  #1# Dis Math    |                  |  #1# Dig Logic   |
14:00 - 15:00 |  #2# Intro CS    |                  |  #1# Dis Math    |                  |                  |
15:00 - 16:00 |  #2# Intro CS    |  #1# Dis Math    |                  |                  |                  |
16:00 - 17:00 |                  |  #1# Dis Math    |                  |                  |                  |
17:00 - 18:00 | #3# Dif&Calc 1   |  #1# Dis Math    | #3# Dif&Calc 1   |                  |                  |
18:00 - 19:00 | #3# Dif&Calc 1   |  #1# Dis Math    | #3# Dif&Calc 1   |                  |   #1# Lin Alg    |
19:00 - 20:00 |                  |                  | #3# Dif&Calc 1   |                  |   #1# Lin Alg    |

 ################ Solution 2 >:
Fitness: 2.9369566512716883E-8
Position: [000100010100100]
Schedule:
              |  SUN             |  MON             |  TUE             |  WED             |  THU             |
 8:00 -  9:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
 9:00 - 10:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
10:00 - 11:00 |                  |                  |                  |                  |  #1# Dig Logic   |
11:00 - 12:00 |                  |                  |                  |                  |  #1# Dig Logic   |
12:00 - 13:00 |  #2# Intro CS    |  #2# Intro CS    |                  |                  |  #1# Dig Logic   |
13:00 - 14:00 |  #2# Intro CS    |  #2# Intro CS    |  #1# Dis Math    |                  |  #1# Dig Logic   |
14:00 - 15:00 |  #2# Intro CS    |                  |  #1# Dis Math    |                  |                  |
15:00 - 16:00 |  #2# Intro CS    |  #1# Dis Math    |                  |                  |                  |
16:00 - 17:00 |                  |  #1# Dis Math    |                  |                  |                  |
17:00 - 18:00 |                  |  #1# Dis Math    |                  |                  |                  |
18:00 - 19:00 |                  |  #1# Dis Math    |                  |                  |   #1# Lin Alg    |
19:00 - 20:00 |                  |                  |                  |                  |   #1# Lin Alg    |
```

```
################ Solution 3 >:
Fitness: 2.9369566512716883E-8
Position: [001100000100100]
Schedule:
              |  SUN             |  MON             |  TUE             |  WED             |  THU             |
 8:00 -  9:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
 9:00 - 10:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
10:00 - 11:00 |                  |                  |                  |                  |  #1# Dig Logic   |
11:00 - 12:00 |                  |                  |                  |                  |  #1# Dig Logic   |
12:00 - 13:00 |                  |                  |                  |                  |  #1# Dig Logic   |
13:00 - 14:00 |                  |                  |  #1# Dis Math    |                  |  #1# Dig Logic   |
14:00 - 15:00 |                  |                  |  #1# Dis Math    |                  |                  |
15:00 - 16:00 |                  |  #1# Dis Math    |                  |                  |                  |
16:00 - 17:00 |                  |  #1# Dis Math    |                  |                  |                  |
17:00 - 18:00 | #3# Dif&Calc 1   |  #1# Dis Math    | #3# Dif&Calc 1   |                  |                  |
18:00 - 19:00 | #3# Dif&Calc 1   |  #1# Dis Math    | #3# Dif&Calc 1   |                  |   #1# Lin Alg    |
19:00 - 20:00 |                  |                  | #3# Dif&Calc 1   |                  |   #1# Lin Alg    |

################ Solution 4 >:
Fitness: 2.9369566512716883E-8
Position: [001100010100000]
Schedule:
              |  SUN             |  MON             |  TUE             |  WED             |  THU             |
 8:00 -  9:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
 9:00 - 10:00 |                  |  #1# Dig Logic   |                  |   #1# Lin Alg    |                  |
10:00 - 11:00 |                  |                  |                  |                  |  #1# Dig Logic   |
11:00 - 12:00 |                  |                  |                  |                  |  #1# Dig Logic   |
12:00 - 13:00 |  #2# Intro CS    |  #2# Intro CS    |                  |                  |  #1# Dig Logic   |
13:00 - 14:00 |  #2# Intro CS    |  #2# Intro CS    |                  |                  |  #1# Dig Logic   |
14:00 - 15:00 |  #2# Intro CS    |                  |                  |                  |                  |
15:00 - 16:00 |  #2# Intro CS    |                  |                  |                  |                  |
16:00 - 17:00 |                  |                  |                  |                  |                  |
17:00 - 18:00 | #3# Dif&Calc 1   |                  | #3# Dif&Calc 1   |                  |                  |
18:00 - 19:00 | #3# Dif&Calc 1   |                  | #3# Dif&Calc 1   |                  |   #1# Lin Alg    |
19:00 - 20:00 |                  |                  | #3# Dif&Calc 1   |                  |   #1# Lin Alg    |

################ END - Top Solutions #####################
```

10.4.4 להוסיף אילוץ אישי

```
4
Enter constraint name:
GYM
Enter day of the week (e.g. MONDAY):
TUESDAY
Enter start time (hh:mm):
17:00
Enter end time (hh:mm):
20:00
Personal constraint added successfully.
```

10.4.5 להסיר אילוץ אישי

```
5
Enter constraint name to remove:
GYM
Personal constraint removed successfully.
```

```
6
Personal Constraints:
None
```

10.4.6 להראות את האילוצים האישים

```
6
Personal Constraints:
- GYM:
    Day: TUESDAY
    Start Time: 17:00
    End Time: 20:00
```

10.4.7 להראות את הלו"ח זמנים הכי טוב עם האילוצים האישים

```
################ START - Top Solutions WITH PERSONAL CONSTRAINT #####################

################ Special Solution 1 >:
Fitness: 2.9369566512716883E-8
Position: [000100010100100]
Schedule:
              |  SUN            |  MON            |  TUE            |  WED            |  THU            |  FRI            |  SAT            |
 8:00 -  9:00 |                 |  #1# Dig Logic  |                 |  #1# Lin Alg    |                 |                 |                 |
 9:00 - 10:00 |                 |  #1# Dig Logic  |                 |  #1# Lin Alg    |                 |                 |                 |
10:00 - 11:00 |                 |                 |                 |                 |  #1# Dig Logic  |                 |                 |
11:00 - 12:00 |                 |                 |                 |                 |  #1# Dig Logic  |                 |                 |
12:00 - 13:00 |  #2# Intro CS   |  #2# Intro CS   |                 |                 |  #1# Dig Logic  |                 |                 |
13:00 - 14:00 |  #2# Intro CS   |  #2# Intro CS   |  #1# Dis Math   |                 |  #1# Dig Logic  |                 |                 |
14:00 - 15:00 |  #2# Intro CS   |                 |  #1# Dis Math   |                 |                 |                 |                 |
15:00 - 16:00 |  #2# Intro CS   |  #1# Dis Math   |                 |                 |                 |                 |                 |
16:00 - 17:00 |                 |  #1# Dis Math   |                 |                 |                 |                 |                 |
17:00 - 18:00 |                 |  #1# Dis Math   |       GYM       |                 |                 |                 |                 |
18:00 - 19:00 |                 |  #1# Dis Math   |       GYM       |                 |  #1# Lin Alg    |                 |                 |
19:00 - 20:00 |                 |                 |       GYM       |                 |  #1# Lin Alg    |                 |                 |

################ END - Top Solutions WITH PERSONAL CONSTRAINT #####################
```

10.4.8 להראות את ה-4 לוחות הזמנים הכי טובים עם האילוצים האישיים

```
################ START - Top Solutions WITH PERSONAL CONSTRAINT #####################

################ Special Solution 1 >:
Fitness: 2.9369566512716883E-8
Position: [000100010100100]
Schedule:
              |  SUN            |  MON            |  TUE            |  WED            |  THU            |
 8:00 -  9:00 |                 |  #1# Dig Logic  |                 |  #1# Lin Alg    |                 |
 9:00 - 10:00 |                 |  #1# Dig Logic  |                 |  #1# Lin Alg    |                 |
10:00 - 11:00 |                 |                 |                 |                 |  #1# Dig Logic  |
11:00 - 12:00 |                 |                 |                 |                 |  #1# Dig Logic  |
12:00 - 13:00 |  #2# Intro CS   |  #2# Intro CS   |                 |                 |  #1# Dig Logic  |
13:00 - 14:00 |  #2# Intro CS   |  #2# Intro CS   |  #1# Dis Math   |                 |  #1# Dig Logic  |
14:00 - 15:00 |  #2# Intro CS   |                 |  #1# Dis Math   |                 |                 |
15:00 - 16:00 |  #2# Intro CS   |  #1# Dis Math   |                 |                 |                 |
16:00 - 17:00 |                 |  #1# Dis Math   |                 |                 |                 |
17:00 - 18:00 |                 |  #1# Dis Math   |       GYM       |                 |                 |
18:00 - 19:00 |                 |  #1# Dis Math   |       GYM       |                 |  #1# Lin Alg    |
19:00 - 20:00 |                 |                 |       GYM       |                 |  #1# Lin Alg    |

################ Special Solution 2 >:
Fitness: 1.1557636012225392E-7
Position: [000100000100100]
Schedule:
              |  SUN            |  MON            |  TUE            |  WED            |  THU            |
 8:00 -  9:00 |                 |  #1# Dig Logic  |                 |  #1# Lin Alg    |                 |
 9:00 - 10:00 |                 |  #1# Dig Logic  |                 |  #1# Lin Alg    |                 |
10:00 - 11:00 |                 |                 |                 |                 |  #1# Dig Logic  |
11:00 - 12:00 |                 |                 |                 |                 |  #1# Dig Logic  |
12:00 - 13:00 |                 |                 |                 |                 |  #1# Dig Logic  |
13:00 - 14:00 |                 |                 |  #1# Dis Math   |                 |  #1# Dig Logic  |
14:00 - 15:00 |                 |                 |  #1# Dis Math   |                 |                 |
15:00 - 16:00 |                 |  #1# Dis Math   |                 |                 |                 |
16:00 - 17:00 |                 |  #1# Dis Math   |                 |                 |                 |
17:00 - 18:00 |                 |  #1# Dis Math   |       GYM       |                 |                 |
18:00 - 19:00 |                 |  #1# Dis Math   |       GYM       |                 |  #1# Lin Alg    |
19:00 - 20:00 |                 |                 |       GYM       |                 |  #1# Lin Alg    |
```

```
 ################ Special Solution 3 >:
Fitness: 1.1557636012225392E-7
Position: [000000010100100]
Schedule:
               |  SUN            |  MON            |  TUE            |  WED            |  THU            |
 8:00 -  9:00  |                 |  #1# Dig Logic  |                 |                 |                 |
 9:00 - 10:00  |                 |  #1# Dig Logic  |                 |                 |                 |
10:00 - 11:00  |                 |                 |                 |                 |  #1# Dig Logic  |
11:00 - 12:00  |                 |                 |                 |                 |  #1# Dig Logic  |
12:00 - 13:00  |  #2# Intro CS   |  #2# Intro CS   |                 |                 |  #1# Dig Logic  |
13:00 - 14:00  |  #2# Intro CS   |  #2# Intro CS   |  #1# Dis Math   |                 |  #1# Dig Logic  |
14:00 - 15:00  |  #2# Intro CS   |                 |  #1# Dis Math   |                 |                 |
15:00 - 16:00  |  #2# Intro CS   |  #1# Dis Math   |                 |                 |                 |
16:00 - 17:00  |                 |  #1# Dis Math   |                 |                 |                 |
17:00 - 18:00  |                 |  #1# Dis Math   |       GYM       |                 |                 |
18:00 - 19:00  |                 |  #1# Dis Math   |       GYM       |                 |                 |
19:00 - 20:00  |                 |                 |       GYM       |                 |                 |

 ################ Special Solution 4 >:
Fitness: 1.1557636012225392E-7
Position: [000100010100000]
Schedule:
               |  SUN            |  MON            |  TUE            |  WED            |  THU            |
 8:00 -  9:00  |                 |  #1# Dig Logic  |                 |  #1# Lin Alg    |                 |
 9:00 - 10:00  |                 |  #1# Dig Logic  |                 |  #1# Lin Alg    |                 |
10:00 - 11:00  |                 |                 |                 |                 |  #1# Dig Logic  |
11:00 - 12:00  |                 |                 |                 |                 |  #1# Dig Logic  |
12:00 - 13:00  |  #2# Intro CS   |  #2# Intro CS   |                 |                 |  #1# Dig Logic  |
13:00 - 14:00  |  #2# Intro CS   |  #2# Intro CS   |                 |                 |  #1# Dig Logic  |
14:00 - 15:00  |  #2# Intro CS   |                 |                 |                 |                 |
15:00 - 16:00  |  #2# Intro CS   |                 |                 |                 |                 |
16:00 - 17:00  |                 |                 |                 |                 |                 |
17:00 - 18:00  |                 |                 |       GYM       |                 |                 |
18:00 - 19:00  |                 |                 |       GYM       |                 |  #1# Lin Alg    |
19:00 - 20:00  |                 |                 |       GYM       |                 |  #1# Lin Alg    |

 ################ END - Top Solutions WITH PERSONAL CONSTRAINT #####################
```

11 תיאור התוצר גרסת ALPHA

11.1 אלגוריתמים:

אלגוריתם PSO (Particle Swarm Optimization) הוא אלגוריתם אופטימיזציה גלובלי המבוסס על חיפוש משולב של חלקיקים בתוך מרחב הפתרונות. המחלקות PSOAlgorithm, Particle ו-Solution מייצגות גרסה מותאמת אישית של אלגוריתם PSO עבור בעיית ייעוץ לוח זמנים.

- מחלקת **PSOAlgorithm:** מחלקה זו מכילה את קוד האלגוריתם הראשי של PSO, ומטפלת באתחול החלקיקים, עדכון המהירות והמיקום שלהם, חישוב התאמת הפתרונות. כמו כן, המחלקה גורסת את האלגוריתם לפי מספר אבחנות מוגדר מראש, ומחזירה את הפתרון הטוב ביותר עבור בעיית הייעוץ.
- מחלקת **Particle**: מחלקה זו מייצגת חלקיק באלגוריתם PSO. כל חלקיק מכיל פתרון נוכחי (שהוא מופע של מחלקת Solution) ומהירות, שמשמעותה עדכון הפתרון בהתאם לפתרונות הטובים ביותר של החלקיק ושל הזוגה כולה. המחלקה מכילה פונקציות לעדכון מהירות החלקיק ולפיהם גם את הפתרון הנוכחי.
- מחלקת **Solution:** מחלקה זו מייצגת פתרון אפשרי במסגרת הבעיה, מכילה רשימה של קורסים מתוכננים בלוח הזמנים. בנוסף, המחלקה מכילה פונקציות לחישוב ההתאמה של הפתרון, שהיא מדד האיכות של הפתרון עבור בעיית הייעוץ. פונקציות נוספות כוללות בדיקות עבור התנגשויות בין קורסים והשוואה בין פתרונות על מנת לקבוע אם הם זהים.

11.1.1 מיקום ומהירות

במקרה של אלגוריתם PSO שלנו המיקום (**position**) של פתרון מייצג את ההקצאה הנוכחית של קורסים וקבוצות בלוח הזמנים. במילים אחרות, המיקום מקודד את המבנה של לוח הזמנים עבור כל קורס וקבוצה על פי הגבלות הבעיה. לכל Particle באלגוריתם יש מצב (position) שמייצג פתרון אפשרי במרחב הפתרונות (ראו דוגמה להרצה בסעיף 11.3 למיקום של חלקיק.)

מהירות (**velocity**) היא ייצוג של השינוי של המיקום בזמן. באלגוריתם PSO, המהירות משמשת כגורם המשפיע על השינוי והעדכון של ההקצאה החדשה לקורסים שלוקחים יחסית להקצאה של הקורסים הנוכחית של כל **Particle** במהלך האיטרציות. "המהירות" מתחשבת בהשפעה של הפתרון הטוב ביותר של החלקיק המסוים ובפתרון הטוב ביותר באופן כללי בקבוצת החלקיקים. בכל איטרציה, המהירות משולבת עם גורמים אקראיים כגון קבועים קוגניטיביים קבועים חברתיים כדי להביא לשינויים במיקום הפתרון (ראו דוגמה להרצה בסעיף 11.3 לשינוי מהירות של חלקיק.)

בקצרה, המיקום מייצג את ההקצאה הנוכחית של לקיחת הקורסים בלו"ז הנוכחי עבור כל חלקיק, והמהירות מייצגת את השינוי לקיחת הקורסים עם זמן בהתאם לפתרונות הטובים ביותר שהתגלו עד כה.

לסיכום, האלגוריתם ה-PSO כפי שמומש במחלקות PSOAlgorithm, Particle ו-Solution, מספק שיטה למציאת פתרון אופטימלי לבעיית ייעוץ לוח זמנים. האלגוריתם משתמש בחלקיקים המייצגים פתרונות ומעדכן אותם בהתאם לפתרונות הטובים ביותר של כל חלקיק ושל הזוכה כולה. מחלקת Solution משמשת לייצוג פתרון אפשרי ולחישוב ההתאמה שלו.

11.2 קוד

נתאר את האלגוריתם באמצעות קטעי קוד מהמחלקות השונות:

1. **מחלקת PSOAlgorithm:**

a. יצירת חלקיקים:

```
public PSOAlgorithm(List<Course> courses) {
    particles = new ArrayList<Particle>();
    Particle particle;
    // Initialize the population with random solutions
    for (int i = 0; i < POPULATION_SIZE; i++) {

        particle = new Particle();
        particles.add(particle);

        //System.out.println("################ SOLUTION " + i + "################");
        //System.out.println(particle.getBestSolution());
    }
    //System.out.println("##########################");
}
```

b. עדכון חלקיקים ומציאת הפתרון הטוב ביותר:

```
public void run() {
    int iteration = 0;

    while (iteration < MAX_ITERATIONS) {
        // Update the velocity and position of each particle

        //int i = 1;
        for (Particle particle : particles) {
            //System.out.print(i++ + "\n\t");
            //particle.printPositionValues();

            particle.updateVelocity(getGlobalBestSolution());
            particle.updatePosition();

            //System.out.print("\t");
            //particle.printPositionValues();
        }
        // Update the global best solution
        updateGlobalBestSolution();
        iteration++;
    }
    // Assign the best solution to the schedule
    this.bestSolution = getGlobalBestSolution();
}
```

2. **מחלקת Particle:**

a. יצירת פתרון ראשוני:

```
public Particle() {
    currentSolution = new Solution();
    bestSolution = new Solution(currentSolution);
    globalBestSolution = new Solution(currentSolution);
    velocity = new double[Course.getAllCourses().size()];
    random = new Random();
}
```

b. חישוב מהירות חדשה "הקצאה חדשה של קורסים יחסית לפתרון הנוכחי":

```
public void updateVelocity(Solution globalBestSolution) {
    for (int i = 0; i < velocity.length; i++) {

        int currentGeneValue = currentSolution.getGenePosValue(i).getValue();
        int bestGeneValue = bestSolution.getGenePosValue(i).getValue();
        int globalBestGeneValue = globalBestSolution.getGenePosValue(i).getValue();

        // Calculate the cognitive and social components of the velocity
        double cognitiveComponent = PSOAlgorithm.C1 * random.nextDouble() * (bestGeneValue - currentGeneValue);
        double socialComponent = PSOAlgorithm.C2 * random.nextDouble() * (globalBestGeneValue - currentGeneValue);
        velocity[i] += cognitiveComponent + socialComponent;
        // Ensure that the velocity does not exceed the maximum or minimum values
        if (velocity[i] > PSOAlgorithm.MAX_VELOCITY) {
            velocity[i] = PSOAlgorithm.MAX_VELOCITY;
        } else if (velocity[i] < PSOAlgorithm.MIN_VELOCITY) {
            velocity[i] = PSOAlgorithm.MIN_VELOCITY;
        }
    }
```

c. עדכון המיקום של החלקיק "עדכון הקורסים שלוקחים" על פי המהירות החדשה:

```
public void updatePosition() {
    for (int i = 0; i < Course.getAllCourses().size(); i++) {
        // Update the gene value based on the current velocity
        Solution.posValue currentGeneValue = currentSolution.getGenePosValue(i);
        Solution.posValue newValue = currentGeneValue;

        if (velocity[i]!=0) {
            if (velocity[i] > PSOAlgorithm.THRESHOLD_NEW_COURSE &&
                    currentSolution.canAddCourse(i)) {
                newValue = Solution.posValue.TAKE_COURSE;
            } else {
                newValue = Solution.posValue.DONT_TAKE_COURSE;
            }
        }
        // Update the gene value in the current solution
        currentSolution.setGenePosValue(i, newValue);
    }
    // Update the personal best solution if the new solution is better
    if (currentSolution.getFitness() < bestSolution.getFitness()) {
        bestSolution = new Solution(currentSolution);
    }
}
```

3. מחלקת Solution:

a. יצירת פתרון ראשוני:

```java
public Solution() {
    List<Course> allCourses = Course.getAllCourses();

    this.chosenCourses = new ArrayList<Course>();


    this.solutionPosition = new posValue[allCourses.size()];
    for (int i=0; i< this.solutionPosition.length; i++)
        this.solutionPosition[i] = posValue.DONT_TAKE_COURSE;

    // Shuffle the courses randomly to create a random solution
    Collections.shuffle(allCourses, new Random());

    // Add courses to the solution one by one, ensuring no collisions
    for (Course course : allCourses) {
        boolean overlaps = false;

        for (Course addedCourse : chosenCourses) {
            if (addedCourse.overlapsWith(course)) {
                overlaps = true;
                break;
            }
        }


        if (!overlaps) {
            this.chosenCourses.add(course);
            this.solutionPosition[course.getIndex()] = posValue.TAKE_COURSE;
        }
    }

    // Calculate the fitness of the solution
    fitness = calculateFitness();
}
```

b. חישוב fitness:

```java
// Calculate the penalty for Course clashes
private double calculateFitness() {
    double penalty = 0.0;
    double numClashes = Course.calculateNumClashes();
    int numChosenCourses = chosenCourses.size();
    int totalNumCourses = Course.getAllCourses().size();

    // Calculate the penalty for not considering each Course
    for(int i=0; i<this.solutionPosition.length;i++) {
        if(this.solutionPosition[i].value ==posValue.DONT_TAKE_COURSE.value)
            penalty += Course.getCourseByIndex(i).getPenalty();
    }

    double totalPenalty = Course.totalPenalty();
    // Calculate the fitness based on the penalty, number of clashes, and number of chosen courses
    double fitness = 1.0 / (totalPenalty - penalty + 1.0);
    fitness *= 1.0 / (numClashes + 1.0);
    fitness /= (double)numChosenCourses / (double)totalNumCourses;

    return fitness;
}
```

לסיכום, האלגוריתם מתחיל מבוסס על מחלקות: PSOAlgorithm, Particle, ו-Solution. הפועל על פי השלבים הבאים:

1. במחלקת PSOAlgorithm, יוצרים חלקיקים ומאתחלים אותם עם פתרונות אקראיים.
2. במהלך כל איטרציה, מעדכנים את החלקיקים על פי המהירות והמיקום שלהם, ומחפשים את הפתרון הטוב ביותר.
3. במחלקת Particle, מחשבים את המהירות החדשה עבור כל חלקיק בהתאם לפתרון הטוב ביותר של החלקיק והפתרון הטוב ביותר באופן כללי.
4. מעדכנים את המיקום של החלקיק על פי המהירות חדשה ובודקים את ההתאמה של הפתרון החדש.
5. במחלקת Solution, מחשבים את ההתאמה של כל פתרון על פי מספר הפגישות המתנגשות בלוח הזמנים.
6. לבסוף, לאחר כל האיטרציות, האלגוריתם מחזיר את הפתרון הטוב ביותר שנמצא במהלך הריצה.

11.3 הדגמה

הדגמה הנוכחית נגדיר שיש רק**2** חלקיקים עם **5** סיבובים

```
public class PSOAlgorithm {

    public static final int MAX_ITERATIONS = 5; // Maximum number of iterations
    public static final int POPULATION_SIZE = 2; // Population size
```

דוגמה להרצה

```
Welcome to the Schedule Manager!
Please enter your name: Liraz
Hello, Liraz!
Creating Your Schedule :) Please Wait...

Round #0:
1
        Current position: 100000010100001 (Fitness: 2.232087054966483E-6)
        Next    position: 100000010001000 (Fitness: 2.3808730185184304E-6)
2
        Current position: 100100010001010 (Fitness: 1.4285428577142744E-6)
        Next    position: 100100010001010 (Fitness: 7.142714288571372E-7)

Round #1:
1
        Current position: 100000010001000 (Fitness: 2.3808730185184304E-6)
        Next    position: 100100010000010 (Fitness: 1.1160435274832414E-6)
2
        Current position: 100100010001010 (Fitness: 7.142714288571372E-7)
        Next    position: 100100010001010 (Fitness: 4.7618095257142476E-7)

Round #2:
1
        Current position: 100100010000010 (Fitness: 1.1160435274832414E-6)
        Next    position: 100000010100000 (Fitness: 1.3227072325102391E-6)
2
        Current position: 100100010001010 (Fitness: 4.7618095257142476E-7)
        Next    position: 100100010001010 (Fitness: 3.571357144285686E-7)

Round #3:
1
        Current position: 100000010100000 (Fitness: 1.3227072325102391E-6)
        Next    position: 100100010001010 (Fitness: 5.494395606593363E-7)
2
        Current position: 100100010001010 (Fitness: 3.571357144285686E-7)
        Next    position: 100100010001010 (Fitness: 2.857085715428548E-7)

Round #4:
1
        Current position: 100100010001010 (Fitness: 5.494395606593363E-7)
        Next    position: 100000010000000 (Fitness: 1.4880208370533852E-6)
2
        Current position: 100100010001010 (Fitness: 2.857085715428548E-7)
        Next    position: 100100010001010 (Fitness: 2.3809047628571238E-7)
Done :)
```

1. התחלנו עם שני מיקומים לחלקיקים - בירוק המיקום ההתחלתי של של שני החלקיקים והערך של ה fitness שלהם מיוצג הדפסה של double fitness(שימו לב ש **E**-7 אומר שצריך להכפיל את המס' ב 10 בחזקת 7-)

2. מתחילים לעשות סיבובים לשינוי מיקום החלקיקים, כלומר מנסים לשנות את בחירת הקורסים של אותו חלקיק על ידי השוואה למיקום של הפתרון הטוב ביותר בעל ה fitness הקטן ביותר.
   a. ניתן להבחין שהחלקיקים מוצאים ומשפרים את המיקום שלהם למקומות בעלי fitness קטם יות.
   b. שימו לב שכל חלקיק זוכר את המיקום בו היה לו את ה fitness הנמוך ביותר, אז לא צריך לדאוג מזה שהחלקיקים לא מסיימים הסיבובים שלהם במיקום לא מינימלי

3. לבסוף במקרה שלנו, הפתרון האופטימלי מתקבל כאשר ה **fitness** של חלקיק מס' 2 בתום ה **Round** ה - 5 [סופרים מ -0 את ה roundים]:

```
Round #4:
1
        Current position: 100100010001010 (Fitness: 5.494395606593363E-7)
        Next    position: 100000010000000 (Fitness: 1.4880208370533852E-6)
2
        Current position: 100100010001010 (Fitness: 2.857085715428548E-7)
        Next    position: 100100010001010 (Fitness: 2.3809047628571238E-7)
Done :)
```

המערכת שעות שקיבלנו היא 🙂 :

```
Fitness: 2.3809047628571238E-7
Position: [100100010001010]
Schedule:
```

	SUN	MON	TUE	WED	THU
8:00 - 9:00				#1# Lin Alg	
9:00 - 10:00				#1# Lin Alg	
10:00 - 11:00			#2# Dis Math		
11:00 - 12:00			#2# Dis Math		
12:00 - 13:00	#2# Intro CS	#2# Intro CS	#2# Dis Math		
13:00 - 14:00	#2# Intro CS	#2# Intro CS	#2# Dis Math		
14:00 - 15:00	#2# Intro CS			#2# Dis Math	
15:00 - 16:00	#2# Intro CS			#2# Dis Math	
16:00 - 17:00					
17:00 - 18:00	#3# Dig Logic			#1# Dif&Calc 1	
18:00 - 19:00	#3# Dig Logic	#3# Dig Logic	#1# Dif&Calc 1	#1# Dif&Calc 1	#1# Lin Alg
19:00 - 20:00	#3# Dig Logic	#3# Dig Logic	#1# Dif&Calc 1	#1# Dif&Calc 1	#1# Lin Alg

12 הערכה

12.1 DATA SET

תוכנית לימודים מסלול יום מדעי המחשב -תשפ"ג

12.1.1 נייצג את הקורסים בפורמט Jason באופן הבא:

לכל קורס יש מס' קורס, "עונש", שם, כינוי וקבוצות לימוד, כל קבוצה יש מס' ושעות הלימוד


```
{
        "course_id": "90901",
        "penalty": "10000",
        "name": "Differential and Integral Calculus 1",
        "nickname": "Dif&Calc 1",
        "group": [
                {
                        "group_number": "1",
                        "time_slots": [
                                {
                                        "startTime": "17:00",
                                        "endTime": "21:00",
                                        "day": "WEDNESDAY"
                                },
                                {
                                        "startTime": "18:00",
                                        "endTime": "20:00",
                                        "day": "TUESDAY"
                                }
                        ],
                },
                {
                        "group_number": "2",
                        "time_slots": [
                                {
                                        "startTime": "08:00",
                                        "endTime": "10:00",
                                        "day": "TUESDAY"
                                },
                                {
                                        "startTime": "13:00",
                                        "endTime": "16:00",
                                        "day": "WEDNESDAY"
                                },
                                {
                                        "startTime": "09:00",
                                        "endTime": "10:00",
                                        "day": "THURSDAY"
                                }
                        ],
                },...
        },...
```

12.2 מדדים

כדי למדוד את איכות האלגוריתם שלנו עבור בניית מערכת שעות, נשנה באלגוריתם בכל איטרציה את השיבוץ במערכת כך שבסופו של התהליך התקבל שיבוץ אופטימלי.

האלגוריתם שלנו ייעצר כאשר הוא יקבל סך ערך "עונשים" (**fitness**) מינימלי ככל הניתן, וזה למעשה ייתן מענה אופטימלי לבעיה.

12.3 צורת הבדיקה

איך נעשה זאת?

- נגדיר משקל "עונש" (**Penalty**) עבור כל הפרה של אילוץ (אישי/מערכת ועוד..).
  מחלקת course יופיע: `private int penalty;`
- נסכום בכל איטרציה את סך ה"עונשים" הכולל של ההפרות שהתקבלו מהשיבוץ הנוכחי ונחשב **fitness**:

```
// Calculate the penalty for Course clashes
private double calculateFitness() {
    double penalty = 0.0;
    double numClashes = Course.calculateNumClashes();
    int numChosenCourses = chosenCourses.size();
    int totalNumCourses = Course.getAllCourses().size();

    // Calculate the penalty for not considering each Course
    for(int i=0; i<this.solutionPosition.length;i++) {
        if(this.solutionPosition[i].value ==posValue.DONT_TAKE_COURSE.value)
            penalty += Course.getCourseByIndex(i).getPenalty();
    }

    double totalPenalty = Course.totalPenalty();
    // Calculate the fitness based on the penalty, number of clashes, and num
    double fitness = 1.0 / (totalPenalty - penalty + 1.0);
    fitness *= 1.0 / (numClashes + 1.0);
    fitness /= (double)numChosenCourses / (double)totalNumCourses;

    return fitness;
}
```

- נחשב כיוון חדש בכל איטרציה לשינוי מבנה הקורסים הנוכחי לאור הפתרון הכי טוב ונעדכן את הפתרון הכי טוב שפגשנו עד כה:

```
public void updatePosition() {
    for (int i = 0; i < Course.getAllCourses().size(); i++) {
        // Update the gene value based on the current velocity
        Solution.posValue currentGeneValue = currentSolution.getGenePosValue(i);
        Solution.posValue newValue = currentGeneValue;

        if (velocity[i]!=0) {
            if (velocity[i] > PSOAlgorithm.THRESHOLD_NEW_COURSE &&
                    currentSolution.canAddCourse(i)) {
                newValue = Solution.posValue.TAKE_COURSE;
            } else {
                newValue = Solution.posValue.DONT_TAKE_COURSE;
            }
        }
        // Update the gene value in the current solution
        currentSolution.setGenePosValue(i, newValue);
    }
    // Update the personal best solution if the new solution is better
    if (currentSolution.getFitness() < bestSolution.getFitness()) {
        bestSolution = new Solution(currentSolution);
    }
}
```

- נעדכן את המיקום עד אשר תתקבלנה מערכות שעות אופטימלית.

13 תוצאות

13.1 מערכת שעות ללא אילוצים

הדגמה הנוכחית נגדיר שיש רק **1000** חלקיקים עם **200** סיבובים

```
public static final int MAX_ITERATIONS = 500; // Maximum number of iterations
public static final int POPULATION_SIZE = 1000; // Population size
```

אחרי חישוב וטעינה של הקורסים קיבלנו מערכת שעות אופטימלית:

```
2
Fitness: 7.107178396588429E-9
Position: [100100010100100]
Schedule:
              |  SUN            |  MON            |  TUE            |  WED            |  THU            |
 8:00 -  9:00 |                 |  #1# Dig Logic  |                 |   #1# Lin Alg   |                 |
 9:00 - 10:00 |                 |  #1# Dig Logic  |                 |   #1# Lin Alg   |                 |
10:00 - 11:00 |                 |                 |                 |                 |  #1# Dig Logic  |
11:00 - 12:00 |                 |                 |                 |                 |  #1# Dig Logic  |
12:00 - 13:00 |  #2# Intro CS   |  #2# Intro CS   |                 |                 |  #1# Dig Logic  |
13:00 - 14:00 |  #2# Intro CS   |  #2# Intro CS   |  #1# Dis Math   |                 |  #1# Dig Logic  |
14:00 - 15:00 |  #2# Intro CS   |                 |  #1# Dis Math   |                 |                 |
15:00 - 16:00 |  #2# Intro CS   |  #1# Dis Math   |                 |                 |                 |
16:00 - 17:00 |                 |  #1# Dis Math   |                 |                 |                 |
17:00 - 18:00 |                 |  #1# Dis Math   |                 | #1# Dif&Calc 1  |                 |
18:00 - 19:00 |                 |  #1# Dis Math   | #1# Dif&Calc 1  | #1# Dif&Calc 1  |   #1# Lin Alg   |
19:00 - 20:00 |                 |                 | #1# Dif&Calc 1  | #1# Dif&Calc 1  |   #1# Lin Alg   |
```

שימו לב לשיפור של ה **fitness** יחסית להדגמה שהייתה לנו 11.3

13.2 מערכת שעות עם אילוצים אישיים

הכנסנו שלושה אילוצים אישיים כמו שניתן לראות בתמונה הבאה:

```
Personal Constraints:
- GYM:
    Day: TUESDAY
    Start Time: 17:00
    End Time: 20:00
- FRIENDS:
    Day: THURSDAY
    Start Time: 15:00
    End Time: 20:00
- SWIMMING:
    Day: SUNDAY
    Start Time: 08:00
    End Time: 10:00
```

המערכת שעות שקיבלנו עם האילוצים היא:

```
################ START - Top Solutions WITH PERSONAL CONSTRAINT #####################

################ Special Solution 1 >:
Fitness: 5.864219257434558E-8
Position: [000000010100100]
Schedule:
              |  SUN            |  MON            |  TUE            |  WED            |  THU            |
 8:00 -  9:00 |    SWIMMING     |  #1# Dig Logic  |                 |                 |                 |
 9:00 - 10:00 |    SWIMMING     |  #1# Dig Logic  |                 |                 |                 |
10:00 - 11:00 |                 |                 |                 |                 |  #1# Dig Logic  |
11:00 - 12:00 |                 |                 |                 |                 |  #1# Dig Logic  |
12:00 - 13:00 |  #2# Intro CS   |  #2# Intro CS   |                 |                 |  #1# Dig Logic  |
13:00 - 14:00 |  #2# Intro CS   |  #2# Intro CS   |  #1# Dis Math   |                 |  #1# Dig Logic  |
14:00 - 15:00 |  #2# Intro CS   |                 |  #1# Dis Math   |                 |                 |
15:00 - 16:00 |  #2# Intro CS   |  #1# Dis Math   |                 |                 |     FRIENDS     |
16:00 - 17:00 |                 |  #1# Dis Math   |                 |                 |     FRIENDS     |
17:00 - 18:00 |                 |  #1# Dis Math   |       GYM       |                 |     FRIENDS     |
18:00 - 19:00 |                 |  #1# Dis Math   |       GYM       |                 |     FRIENDS     |
19:00 - 20:00 |                 |                 |       GYM       |                 |     FRIENDS     |

################ END - Top Solutions WITH PERSONAL CONSTRAINT #####################
```

מדהים !!!

14 סיכום ומסקנות

במהלך הפרויקט שאפנו ליצירת מערכת שעות אופטימלית ככל הניתן, אשר תתאים לצרכיו האישיים של הסטודנט.

ניתן לראות כי הושגה המטרה, והאלגוריתם נותן לנו את המערכת עם ה-fitness הנמוך ביותר. כך למעשה מתקבלת התוצאה האופטימלית ביותר.

על המערכת שעות האופטימלית מוכלים אילוצי הסטודנט, ומתקבלת לנו מערכת שעות אופטימלית המתחשבת באילוצים.

מימוש אלגוריתם הPSO היה מאוד מורכב וקשה להבנה ולביצוע מהיותו אלגוריתם תיאורטי. בנוסף הדבר לקח לנו הרבה מאוד זמן, כמוכן הוא מהווה חלק עיקרי מהפרויקט ומשפיע על תוצאותיו.

אפשר יהיה לשפר את המערכת בכך שהיא תוכל לאפשר לכמה משתמשים להיכנס, ולמעשה תזכור כל סטודנט האילוצים שלו והקורסים שאותו עליו לבצע.

ניתן יהיה לעשות את המערכת בצורה יותר אטרקטיבית וקלה למשתמש. לדוגמה: הוספת GUI , ייצוא המערכת לקובץ Excel אשר נשלח במייל.

15 רשימת מקורות

1. A Review of Optimization Algorithms for University Timetable Scheduling

https://www.etasr.com/index.php/ETASR/article/view/3832

2. Adaptive Genetic Algorithm for High School Time-Table

https://www.proquest.com/docview/2570431331?pq-origsite=primo

3. Timetable Scheduling Using Particle Swarm Optimization

https://www.researchgate.net/publication/224647262_Timetable_Scheduling_Using_Particle_Swarm_Optimization

4. Explanation about PSO

https://en.wikipedia.org/wiki/Particle_swarm_optimization

5. Computer science study program at Afka College

תוכנית לימודים מסלול יום מדעי המחשב -תשפ"ג