מגישים: רביב חכם, 208387951
פלג חכם, 208387969

מבוא לאופטימיזציה – בעיית שיבוץ משמרות ברמנים

הצגת הבעיה

בעיית ניהול סידור העבודה של ברמנים היא תת בעיה של בעיית שיבוץ משמרות אשר הינה בעיה ידועה באופטימיזציה וידועה כבעיה np-hard.

הבעיה היא בעיית קביעת סידור עבודה אישי עבור כל ברמן כך שלכל ברמן ישנם בקשות שיבוץ שונות בהתאם לימים והשעות שנוחות לו, ואנו נרצה למצוא פתרון חוקי (פתרון שמקיים את כל האילוצים הקשים) אשר ישמור על כל אילוצים אלו.

מקור הרעיון לפרויקט הנ"ל:

לאחר השחרור, עבדנו כברמנים בבר בגן אירועים. נירו, שהיה אחראי על סידור המשמרות, היה מתלונן על הקושי להצליח לשבץ את כל הברמנים בצורה שתמקסם את רצונו של כל ברמן ובפרט סידור אשר יעמוד באילוצים הקשים. כתוצאה מכך נירו היה מתעכב באופן קבוע עם פרסום סידור העבודה, מה גם שלרוב, מרבית הברמנים לא היו שבעי רצון שכן נירו לא הצליח לעקוב אחר הבקשות של כולם.

אי לכך החלטנו להמציא מערכת אופטימיזציה אשר משתמשת באלגוריתמים אותם למדנו במסגרת הקורס, כך שתפתור את בעיית סידור העבודה ותמצא סידור שבועי, בהתאם לאילוצים הקשים והקלים שהיו בחברת הבר בה עבדנו, בצורה הטובה ביותר.

למעשה פיתחנו אתר ווב אשר יעזור לחברת הבר בכך שע"י הזנת בקשות השיבוצים השונות, של הברמנים, ובחירת המודל הרצוי החברה תוכל לקבל את הפתרון האופטימלי/ פתרון שמתקרב לפתרון אופטימלי של שיבוץ הברמנים למשמרות השונות בסידור העבודה.

נדגיש כי במסגרת העבודה בבר, הגן עבד רק בימים ראשון עד חמישי והיו 2 סוגי משמרות אפשריות במהלך היום:

-משמרת בוקר-צהרים: אירוע של ברית/ אירוע עסקים.

-משמרת צהרים-ערב: אירוע חתונה.

אילוצי החברה:

-אילוצים קשים:

- אסור לברמן לעבוד ביותר ממשמרת אחת ביום (כלומר או בוקר או ערב).
- ברמן לא יכול לעבוד משמרת ערב (חתונה שנגמרת לרוב באמצע הלילה) ובמשמרת בוקר של היום שלמחרת (ברית/אירוע עסקי אשר מתחיל לרוב בשעות הבוקר המוקדמות).
- ברמן צריך לציין מה המספר המקסימלי של משמרות שיוכל לבצע בשבוע, כך שהמספר נע בין 2 ל 5 משמרות.
- לפחות 3 ברמנים במשמרת.
- לא יותר מ 5 ברמנים במשמרת.
- נאפשר לכל ברמן לציין מהם הימים בהם הוא מעוניין לעבוד, באופן זה, ימים שהוא לא סימן מייצגים ימים בהם הוא לא יכול/לא מעוניין לעבוד כלל, לכן לא נשבץ אותו בימים אלו.

-אילוצים קלים:

- בהתאם לאופן בו עובדת חברת הבר בה עבדנו, נרצה לאפשר לכל ברמן לציין מהם הימים בהם הוא מעוניין לעבוד, אך לא בטוח שאותו הברמן יעבוד בכל ימים אלו שסימן.

נרצה להבין מהו האלגוריתם שממקסם את מידת ההצלחה של קביעת הסידור מבין כל האלגוריתמים אותם ממשנו ובהם השתמשנו, לכן נציג את תוצאת ההרצות עבור המקרים השונים.

נדגיש כי בכל המודלים השונים יש בדיוק את אותם משתני החלטה (וכמובן שהתחום של אותם המשתנים הינו זהה).

כמו כן, בכל המודלים נחשיב פתרון כחוקי במידה ועבור ההשמה שהצבנו קיבלנו פתרון שמספק את כל האילוצים הקשים אשר נגזרו ממדיניות החברה, כאשר כל אחד מהמודלים השונים ינסה לספק את האילוצים הרכים עד כמה שאפשר ובהתאם לשיטת העבודה ולמדיניות בה נוקט כל מודל.

המודלים השונים איתם נעבוד:

- אחוז משרה מקסימלי:

בהתאם למדיניות מודל זה נשאף שאחוז המשרה של כל ברמן יהיה אופטימלי (גבוהה ככל הניתן), תוך שמירה על כל האילוצים הקשים- של החברה וניסיון לספק, ככל הניתן, את הסיפוקים הרכים- של העדפות הברמנים.

ננסח את הבעיה והמודל בצורה פורמלית כבעיית אופטימיזציה:

- <u>משתני ההחלטה</u>: מדובר על בעיית תכנות בשלמים, כך שלכל ברמן ישנם **10 = 5 * 2** משתנים בינאריים (עבד\ לא עבד עבור כל ברמן, לכל יום עבודה עבור שני סוגי המשמרות שקיימים).

  כמובן שעבור משתנים אלו התחום הינו {0,1} שכן אלו משתנים בינאריים.

- <u>המטרה (**objective**):</u> מקסימום של כמות כל המשמרות של כל עובדי הבר יחדיו.

- <u>האילוצים (**constraints**):</u> אילוצים קשים של חברת הבר (מוסבר בפירוט למעלה) + אילוצים קלים אותם נשאף לספק (אך אין חובה לכך), לכן נאפשר לקיים את אילוצים אלו של כמות המשמרות אותה כל ברמן מעוניין לעבוד כך שכל ברמן יקבל לכל היותר את הכמות שביקש אך אין הכרח שיקבל בדיוק את כמות המשמרות שביקש (כלומר האילוץ יופיע עם קטן שווה ולא עם שווה ממש).

- סיפוק אילוצים (**constraint satisfaction**):

נשים לב כי המודל הנ"ל מנסה לספק את כל האילוצים (ומכאן שמו), שכל ברמן יקבל בדיוק את כמות המשמרות שביקש לעבוד בשבוע (מדובר בכמות המשמרות ולא בימים הספציפיים אותם ביקש), ובאופן זה גם להביא למינימום את כמות המשמרות שעובדי הבר קיבלנו על אף שלא סימנו את המשמרת הנ"ל בסידור העבודה. לכן במידה ויצליח, המודל יחזיר את הפתרון שמצא, אך אם לא קיימת השמה תקינה וחוקית שתספק את כל האילוצים האלו, המודל יחזיר שלא קיים פתרון.

ננסח את הבעיה והמודל בצורה פורמלית כבעיית אופטימיזציה:

- <u>משתני ההחלטה</u>: מדובר על בעיית תכנות בשלמים, כך שלכל ברמן ישנם **10 = 5 * 2** משתנים בינאריים (עבד\ לא עבד עבור כל ברמן, לכל יום עבודה עבור שני סוגי המשמרות שקיימים).

  כמובן שעבור משתנים אלו התחום הינו {0,1} שכן אלו משתנים בינאריים.

- <u>המטרה (**objective**):</u> להביא למינימום של כמות המשמרות שעובדי הבר קיבלנו, על אף שלא סימנו את המשמרת הנ"ל בסידור העבודה.

- **<u>האילוצים (constraints):</u>** אילוצים קשים של חברת הבר (מוסבר בפירוט למעלה), וכמו כן, שכל ברמן יקבל בדיוק את כמות המשמרות שמבקש לעבוד בשבוע (מדובר בכמות המשמרות ולא בימים הספציפיים אותם ביקש).

- **טיפוס הרים (Hill climbing):**

לעיתים נרצה למצוא גם פתרון שאינו הפתרון האופטימלי (מסיבות שונות).
כיוון שאנו מדברים על בעיית אופטימיזציה, נרצה לנסות גם מודלים איטרטיביים אשר ידועים כמודלים מסוג **any time**.
מודל זה
בהתאם למודל זה, נתחיל המצב התחלתי כלשהו וננסה לשפרו במהלך האיטרציות השונות. כמו שאמרנו, במודל הנ"ל אין כל הבטחה להביא למציאת פתרון אופטימלי, אך מובטח שכאשר נעצור את האלגוריתם, בכל שלב שהוא, האלגוריתם יחזיר את המצב הטוב ביותר עד כה (כיוון שבוחר בכל פעם בפעולה אשר תגדיל את הערך בצורה המרבית מהמצב הנוכחי, כלומר אם ערך השכן הטוב ביותר שווה או גדול מהערך הנוכחי, אז נבחר בשכן זה כך שהוא יהיה המצב הנוכחי החדש).
נזכיר כי על פי האלגוריתם הבסיסי, נבחר בנקודה רנדומלית כנקודת ההתחלה, ובכל איטרציה נבחר את אחד מבין היורשים (אשר ערכם גדול משל הקודקוד הנוכחי) בצורה רנדומלית. כלומר באלגוריתם הנ"ל מובטח שכאשר נעצור נהיה במקום טוב לפחות כמו המקום בו התחלנו, אך יתכן כי אם ניתן לאלגוריתם לרוץ ללא הגבלת זמן עדיין לא נגיע לאופטימום גלובלי, שכן המודל יוכל להיתקע באופטימום מקומי


```
while f-value(state) <= f-value(next-best(state))
    state := next-best(state)
```

כפי שלמדנו לאלגורית טיפוס ההרים (**Hill climbing**) ישנם מספר פיתוחים אשר יכולים להביא לשיפור בביצועי האלגוריתם, ולמזער את הבעייתיות שבאלגוריתם כגון (**first choice , random restart** , נסיגה בחיפוש) .
אנו מימשנו את המודל הנ״ל עם **random restart**, כלומר
לאחר התכנסות הרצנו שוב את האלגוריתם מנקודת התחלה רנדומלית חדשה כך שחזרנו על התהליך הנ״ל מספר איטרציות ובדקנו כיצד כמות האיטרציות השונות משפיע על ביצועי המודל.

ננסח את הבעיה והמודל בצורה פורמלית כבעיית אופטימיזציה:

- <u>משתני ההחלטה</u>: מדובר על בעיית תכנות בשלמים, כך שלכל ברמן ישנם **10 = 5 * 2** משתנים בינאריים (עבד\ לא עבד עבור כל ברמן, לכל יום עבודה עבור שני סוגי המשמרות שקיימים).
  כמובן שעבור משתנים אלו התחום הינו {0,1} שכן אלו משתנים בינאריים.
- <u>המטרה (**objective**):</u> להביא למינימום את העונש שנקבל עבור האילוצים הקשים והרכים.
- <u>האילוצים (**constraints**):</u> --- (באים ליידי ביטוי במטרה)

- **simulated annealing:**

גם האלגוריתם הנ״ל, בדומה למודל הקודם שהצגנו, שואף למצוא גם פתרון שאינו הפתרון האופטימלי (מסיבות שונות).
כיוון שאנו מדברים על בעיית אופטימיזציה, נרצה לנסות גם מודלים איטרטיביים אשר ידועים כמודלים מסוג **any time**.
בהתאם למודל זה, נתחיל במצב התחלתי כלשהו וננסה לשפרו במהלך האיטרציות השונות.
במודל הנ״ל אין כל הבטחה להביא למציאת פתרון אופטימלי, אך מובטח שכאשר נעצור את האלגוריתם, בכל שלב שהוא, האלגוריתם יחזיר את המצב הטוב ביותר עד כה (כיוון שבוחר בכל פעם בפעולה אשר תגדיל את הערך בצורה המרבית מהמצב הנוכחי, כלומר אם ערך השכן הטוב ביותר שווה או גדול מהערך הנוכחי, אז נבחר בשכן זה כך שהוא יהיה המצב הנוכחי החדש), אך בשונה מהמודל הקודם שראינו, במודל הנ״ל גם אם השכן הינו בעל ערך קטן יותר נסכים לעבור אליו בהסתברות של $e^{\frac{\Delta E}{T}}$
אשר מושפעת מהטמפרטורה (**T** – אשר מייצגת את הזמן שעבר) וממידת ההפרש בין ערכו של השכן החדש לערך השכן הנוכחי.
באופן זה, המודל יאפשר ביצוע של ״בחירה רעה״ שכזו בעיקר בתחילת דרכו, וכאשר ההפרש הינו קטן.

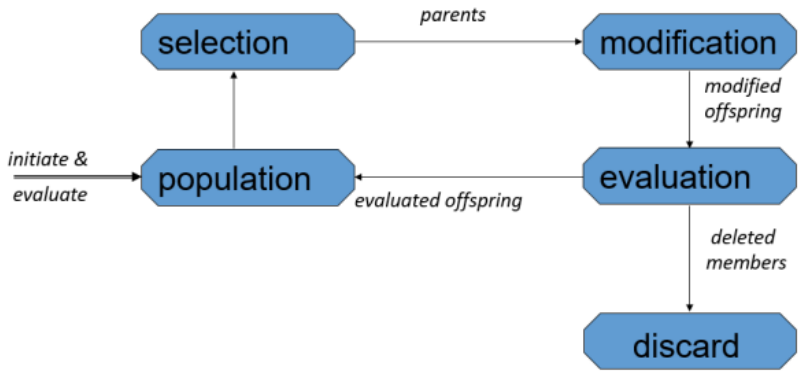
נזכיר כי על פי האלגוריתם הבסיסי, נבחר בנקודה רנדומלית כנקודת ההתחלה וגם כאן יתכן כי אם ניתן לאלגוריתם לרוץ ללא הגבלת זמן עדיין לא נגיע לאופטימום גלובלי, שכן המודל יוכל להיתקע באופטימום מקומי, אך כמובן שפחות סביר שהדבר יקרה בשימוש במודל זה, שכן בדיוק על בעיה זו מנסה להתגבר באמצעות אותם "בחירות רעות" לאזורים חדשים שלא ביכר בהם עדיין (מתוך תקווה שהדבר יוביל לקבלת תוצאה אופטימלית).

ננסח את הבעיה והמודל בצורה פורמלית כבעיית אופטימיזציה:

- <u>משתני ההחלטה</u>: מדובר על בעיית תכנות בשלמים, כך שלכל ברמן ישנם **10 = 5 * 2** משתנים בינאריים (עבד\ לא עבד עבור כל ברמן, לכל יום עבודה עבור שני סוגי המשמרות שקיימים).
  כמובן שעבור משתנים אלו התחום הינו {0,1} שכן אלו משתנים בינאריים.
- <u>המטרה (**objective**):</u> להביא למינימום את העונש שנקבל עבור האילוצים הקשים והרכים.
- <u>האילוצים (**constraints**):</u> --- (באים ליידי ביטוי במטרה)

- **האלגוריתם הגנטי (Genetic algorithm):**

שוב, כיוון שאנו מדברים על בעיית אופטימיזציה, נרצה לנסות מודל איטרטיבי אשר ידוע כמודל מסוג **any time**.
מודל זה מקבל אוכלוסייה כללית (כלומר מקבל פתרונות אפשריים) ומתוכם בוחר את המועמדים החזקים ביותר להתרבות, כאשר נעריך את החוזק של כל פתרון ובהתאם להערכה זו נבחר את החזקים ביותר. לפתרונות אשר נבחרו נבצע התאמות (כלומר זיווגים בין הפתרונות השונים שהתקבלו) בהסתברות כלשהי אותה בחרנו מראש. נחזור על התהליך הנ"ל מספר פעמים, ובאופן זה, חלק מהפתרונות יישארו (החזקים ישרדו ואותם נחזיר), וחלק מהפתרונות "ימותו" במהלך התהליך (החלשים ייעלמו).



נתחיל המצב התחלתי כלשהו וננסה לשפרו במהלך האיטרציות השונות. כמו שאמרנו, במודל הנ״ל אין כל הבטחה להביא למציאת פתרון אופטימלי, אך סביר ביותר שכאשר נעצור את האלגוריתם, בכל שלב שהוא, האלגוריתם יחזיר את המצב הטוב ביותר עד כה (כיוון שרק החזקים שורדים).

ננסח את הבעיה והמודל בצורה פורמלית כבעיית אופטימיזציה:

- <u>משתני ההחלטה</u>: מדובר על בעיית תכנות בשלמים, כך שלכל ברמן ישנם **10** = **5** * **2** משתנים בינאריים (עבד\ לא עבד עבור כל ברמן, לכל יום עבודה עבור שני סוגי המשמרות שקיימים).
  כמובן שעבור משתנים אלו התחום הינו {0,1} שכן אלו משתנים בינאריים.
- <u>המטרה (**objective**):</u> להביא למינימום את העונש שנקבל עבור האילוצים הקשים והרכים.
- <u>האילוצים (**constraints**):</u> --- (באים ליידי ביטוי במטרה)

תוצאות שהתקבלו מהרצת המודלים השונים איתם עבדנו:

נרצה להבין מהו האלגוריתם שממקסם את מידת ההצלחה של קביעת הסידור מבין כל האלגוריתמים אותם ממשנו ובהם השתמשנו, לכן נציג את תוצאת ההרצות עבור המקרים השונים בבדיקות השונות שביצענו.

בדיקה מספר 1:

בבדיקה הראשונה שערכנו, עבור 14 ברמנים שקיימים בחברה לקחנו את בקשותיהם מהאחמ"ש והכנסנו אותם כהשמות לנתונים בתוכנה שיצרנו, וכך קיבלנו את הסידורים הבאים שהתקבלו בהרצת המודלים השונים:

- עבור המודל של אחוז משרה מקסימלי:

במקרה הנ"ל המודל הצליח לספק 41 בקשות שונות של שיבוץ למשמרות השונות (מתוך 49 הבקשות שהיו). חשוב לציין שהפער תקין לגמרי שכן האלגוריתם אכן הביא את השיבוץ האופטימאלי עם כמות המשמרות המרבית שיכל לספק עבור ההשמות הללו שכן היו גם בקשות לא חוקיות אשר לא קיימו את האילוצים הקשים, כך לדוגמה, ניתן לראות כי הברמן הראשון ביקש לעבוד 5 משמרות: ביום ראשון משמרת בוקר וערב, ביום שני משמרת בוקר וערב, וביום שלישי משמרת בוקר, אך כמובן שהדרישה הנ"ל אינה מקיימת את האילוצים הקשים (2 משמרות ביום ובנוסף משמרת ערב ולמחרת משמרת בוקר) ולכן המערכת שבנינו שיבצה אותו רק ב 3 משמרות (משמרת בוקר בימים: ראשון, שני ושלישי).

Bartenders Work Schedule

Shift	Sunday	Monday	Tuesday	Wednesday	Thursday
Morning-Evening	Bartender1 Bartender2 Bartender5 Bartender6 Bartender14	Bartender1 Bartender5 Bartender6 Bartender10 Bartender14	Bartender1 Bartender5 Bartender6 Bartender9	Bartender4 Bartender5 Bartender6	Bartender2 Bartender3 Bartender4 Bartender7
Evening-Noon	Bartender7 Bartender8 Bartender11 Bartender12 Bartender13	Bartender2 Bartender8 Bartender12	Bartender2 Bartender3 Bartender11 Bartender14	Bartender8 Bartender10 Bartender13 Bartender14	Bartender9 Bartender11 Bartender12 Bartender13

- עבור המודל של סיפוק אילוצים (**constraint satisfaction**):

במקרה הנ"ל המודל אכן הצליח לספק פתרון אופטימאלי שעונה על סיפוק האילוצים הקשים והרכים, כלומר מאפשר לכל ברמן לעבוד בדיוק ככמות המשמרות שביקש לעבוד במהלך השבוע.

נשים לב, כי לשם סיפוק כל אילוצים אלו, המודל היה צריך לתת לברמן גם משמרות שאינן בהכרח המשמרת הספציפית שביקש (בימים ובשעות אשר ביקש). את משמרות אלו דאגנו לסמן בצבע אחר (צבע אדום), ובכך ניתן לראות בברור באילו פעמים המודל נתן לברמן לעבוד במשמרות שלאו דווקא ביקש, וזאת על מנת לספק את האילוץ של כמות המשמרות שאותו הברמן ביקש.

בדוגמה שלנו, ניתן לראות כי המודל חילק 8 משמרות לברמנים שונים בשעות/ ימים שאינם ביקשו, כך למשל עבור הברמן הראשון ניתנו 2 משמרות בימים/שעות שלא ביקש וזאת כיוון שרצה לעבוד 5 ימים במהלך השבוע אך בסידור הציב השמה שאינה מתיישבת עם האילוצים הקשים, שכן הברמן הראשון ביקש לעבוד 5 משמרות: ביום ראשון משמרת בוקר וערב, ביום שני משמרת בוקר וערב, וביום שלישי משמרת בוקר, אך כמובן שהדרישה הנ"ל אינה מקיימת את האילוצים הקשים (2 משמרות ביום ובנוסף משמרת ערב ולמחרת משמרת בוקר) ולכן המערכת שבנינו שיבצה אותו גם בימים רביעי וחמישי בשעות הערב, על אף שאינו סימן שמעוניין לעבוד בימים אלו, ובכך המודל אכן עמד באילוץ של כמות המשמרות שברמן ביקש כפי שהוגדר.

Bartenders Work Schedule

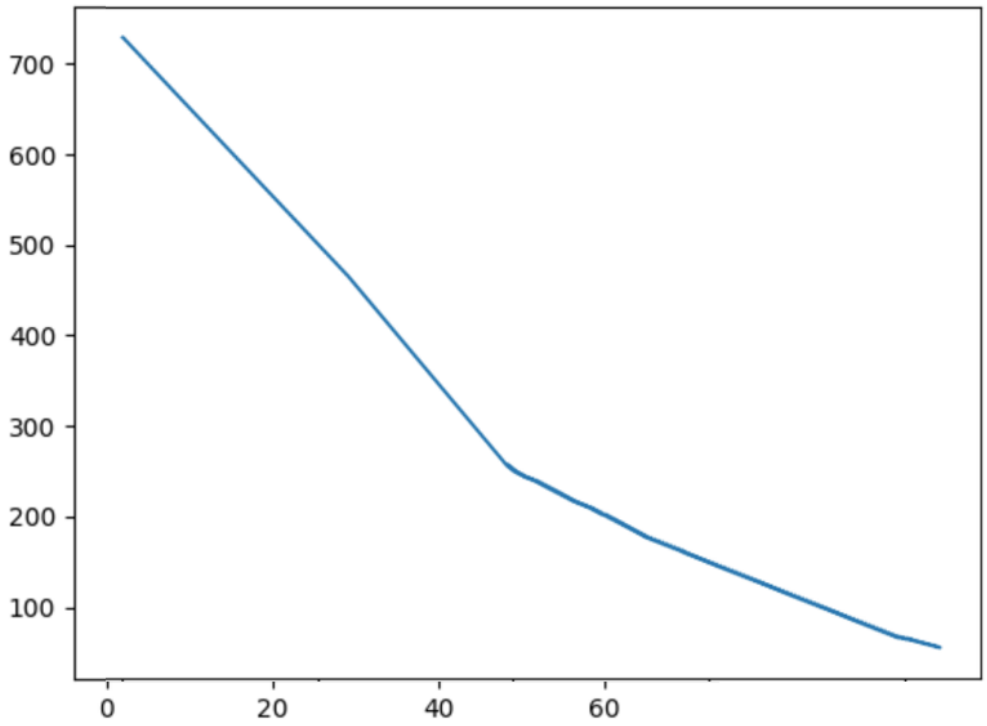
Shift	Sunday	Monday	Tuesday	Wednesday	Thursday
Morning-Evening	Bartender1 Bartender2 Bartender5 Bartender6 Bartender14	Bartender1 Bartender5 Bartender6 Bartender10 Bartender11	Bartender1 Bartender5 Bartender6 Bartender9 Bartender11	Bartender5 Bartender6 Bartender11 Bartender12	Bartender2 Bartender3 Bartender4 Bartender7 Bartender12
Evening-Noon	Bartender3 Bartender7 Bartender8 Bartender12 Bartender13	Bartender2 Bartender8 Bartender12 Bartender13 Bartender14	Bartender2 Bartender4 Bartender10 Bartender13 Bartender14	Bartender1 Bartender8 Bartender10 Bartender13 Bartender14	Bartender1 Bartender5 Bartender9 Bartender11 Bartender13

כלומר המודל הנ"ל אכן החזיר פתרון אופטימלי, אך נשים לב כי זהו פתרון שונה מהפתרון האופטימלי של המודל הראשון שראינו (של אחוז משרה מקסימלי). לכן אנו למדים מכך שישנם מספר פתרונות אופטימליים אפשריים, כאשר כל פתרון שם דגש על ערך/אילוץ אחר שחשוב לו, אותו הוא רוצה לספק.

במודל זה נרצה להשתמש למשל בחברה בה העובדים שמים מעל הכל את השכר אותו הם מעוניינים לקבל ולכן המעסיק ינסה לספק להם את כמות המשמרות אשר ביקשו, גם אם זה סותר את הימים/שעות של השיבוצים הספציפיים שהברמנים ביקשו.

- **טיפוס הרים (Hill climbing):**

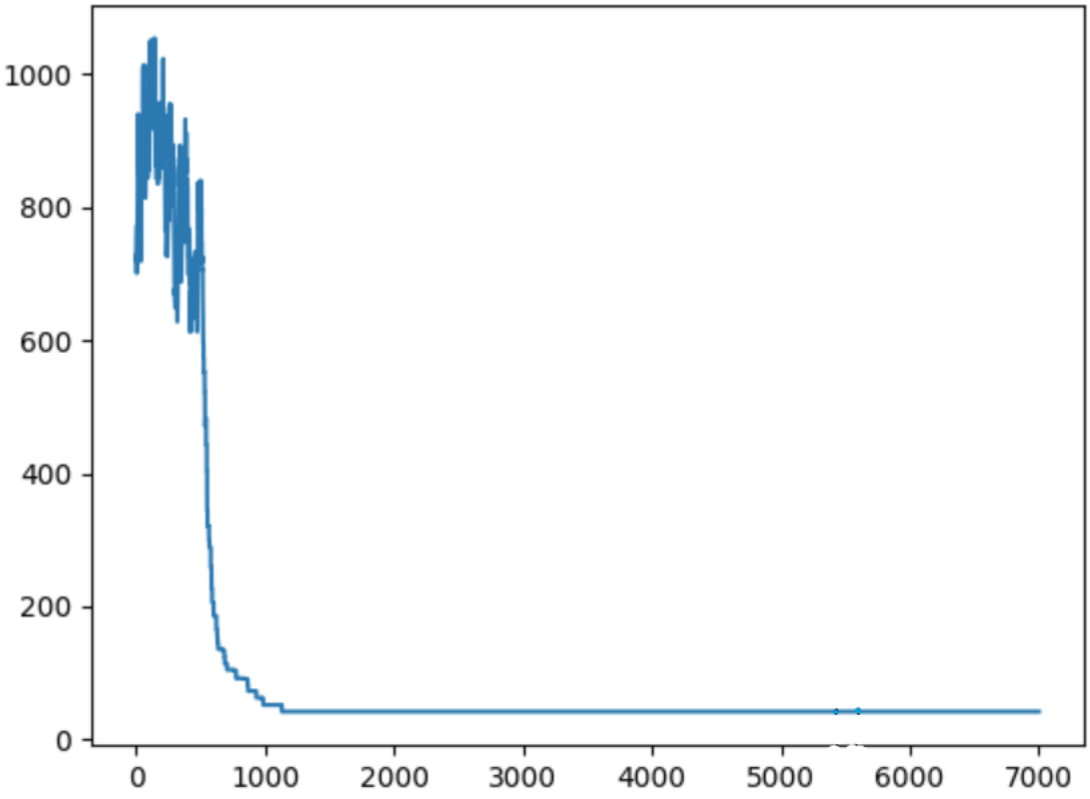
השתמשנו במודל ה **Hill climbing** ויצרנו גרף מהנתונים שאספנו, כך שכל נקודה בגרף מייצגת משמרת שבה היו ברמנים ששובצו אליה על אף שלא ביקשו. ניתן לראות כי המודל התחיל עם ערך שגבוה מ 750, והגיע בסופו של דבר לערך של 50, כלומר אכן ניתן לראות את השיפור בגרף



- **simulated annealing:**

כאן, הרצנו את המודל 6900 איטרציות, שבסופם התקבל הגרף הנ"ל. כפי שניתן לראות, המודל התחיל עם ערך התחלתי של 750 ובסופם התכנס לערך של 50. בנוסף, אפשר לראות מהגרף כי בתחילת הריצה הגרף מאוד רועש (שבשיאם הגיע לערך של 870). תופעה זו הגיונית ומתאימה לאופיו של המודל הנ"ל בו מאפשרים "בחירות רעות", כלומר מעבר

לנקודות גבוהות יותר וזאת על מנת למצוא אזורים חדשים שיאפשרו לנו לשפר, ובכך למנוע מצב של **min/max** מקומי.

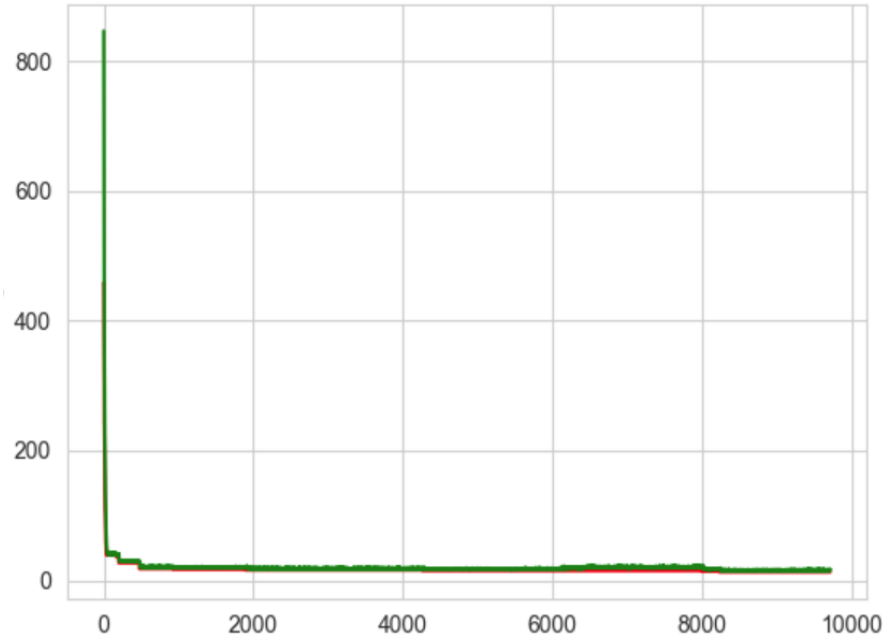


- טיפוס הרים (**Hill climbing**)עם **random restart** :

ביצענו הרצה של המודל הנ״ל עם השיפור ש **random restart** מספק למשך 35 איטרציות. אומנם פעולה זו לקחה מספר דקות של הרצה דבר שכן נרצה לתת עליו את הדעת, אך הדבר כן היה ערכי מאוד שכן בסופם קיבלנו כי הערך הסופי הינו 7 (תוצאה שנחשבת כמעט אופטימאלית) ושמה את המודל הנ״ל בראש הבדיקה שלנו למעשה.

- האלגוריתם הגנטי (**Genetic algorithm**):

הרצנו את מודל זה למשך מעל 8000 איטרציות פעולה שגם כאן אכן לקחה לנו מספר דקות אך גם בסופן קיבלנו תוצאה של 13, כלומר תוצאה שקצת פחות טובה מהתוצאה שקיבלנו באלגוריתם של טיפוס הרים עם **random restart** אך כן מתקרבת לתוצאה האופטימאלית, כאשר עיקר השיפור מתבצע במשך האלף איטרציות החדשות בהם הייתה קפיצה מערך של 900 לערך של פחות מ 50, ולאחר מכן השיפור הינו מינורי יחסית לכמות האיטרציות.



אפשר לראות שכאן מספר האיטרציות שהיינו צריכים על מנת להגיע לתוצאה שמתקרבת לתוצאה האופטימאלית הינה לפחות אלף איטרציות, כלומר כמות שגבוהה בהרבה מכמות האיטרציות שהיינו צריכים בשביל לקבל תוצאה אופטימאלית כשהשתמשנו במודל ה **random restart**.

וזו ההשמה שקיבלנו במצב זה:

Bartenders Work Schedule

Shift	Sunday	Monday	Tuesday	Wednesday	Thursday
Morning-Evening	Bartender1 Bartender5 Bartender8 Bartender13	Bartender1 Bartender5 Bartender6 Bartender8 Bartender11	Bartender1 Bartender5 Bartender6 Bartender9 Bartender10	Bartender2 Bartender5 Bartender6 Bartender8 Bartender12	Bartender2 Bartender5 Bartender6 Bartender7 Bartender12
Evening-Noon	Bartender2 Bartender3 Bartender7 Bartender12 Bartender14	Bartender2 Bartender4 Bartender12 Bartender13 Bartender14	Bartender3 Bartender4 Bartender11 Bartender13 Bartender14	Bartender1 Bartender10 Bartender11 Bartender13 Bartender14	Bartender1 Bartender9 Bartender10 Bartender11 Bartender13

כלומר סה"כ אפשר להסיק מספר מסקנות מהבדיקה הנ"ל שהרצנו:

- אלגוריתם האיטרטיבי **hill climbing** אשר משתמש ב **random restart** הינו אלגוריתם **any time** אשר במקרה הנ"ל הצליח לאחר מעט איטרציות לתוצאה השואפת לתוצאה האופטימאלית, וזאת לעומת שאר המודלים האיטרטיבים אשר היו

צריכים מספר איטרציות גדול בהרבה על מנת להגיע לתוצאות טובות, וגם תוצאות אלו היו פחות מוצלחות מאלו שהמודל של **random restart** הצליח להשיג.

שני המודלים הראשונים- כלומר המודל אשר מביא לאחוז משרה מקסימאלי והמודל של **constraint satisfaction** אשר דואג שכל ברמן יקבל בדיוק את כמות המשמרות שביקש לעבוד בשבוע ותוך כדי שואף להביא למינימום את כמות המשמרות שעובדי הבר קיבלנו על אף שלא סימנו את המשמרת הנ"ל בסידור העבודה, אכן מביאים לפתרון אופטימאלי, אך כמובן ששני הפתרונות המתקבלים מההרצות של שני מודלים אלו הינם פתרונות אופטימאליים שונים, כך שנרצה להשתמש בכל אחד מהם בהתאם לדרישה ולמטרת החברה שעושה שימוש בתוכנית זו. כלומר, במידה ואמנו מעוניינים בראש ובראשונה לספק את בקשות השיבוץ השונות באופן שיהיה לכל ברמן נוח לעבוד במשמרת ששובץ בה, נבחר במודל אשר מביא לאחוז משרה מקסימאלי שכן מודל זה מצד אחד מנסה למקסם את כמות המשמרות שמחלק לברמנים השונים שבחברה, אך מצד שני הוא משתדל שלא לשבץ ברמנים בימים/משמרות שאינן נוחות לו, כלומר זהו מודל יותר גמיש שמתחשב בצרכי הברמנים, ואילו נשתמש במודל של ה **constraint satisfaction** כאשר נרצה לשים מעל הכל את השכר השבועי אותו העובדים יקבלו (בהתאם לכמות המשמרות שרצו לעבוד) ולכן המעסיק ינסה לספק להם את כמות המשמרות אשר ביקשו, גם אם זה סותר את הימים/שעות של השיבוצים הספציפיים שהברמנים העדיפו לעבוד בהם.

בדיקה מספר 2:

בבדיקה השנייה שערכנו, בדקנו את סידור המשמרות שמתקבל עבור 20 ברמנים כאשר לברמנים אין בקשות כלל לקבלת משמרת ספציפית בסידור העבודה שהגישו, כלומר, כל ברמן רק ציין כמה משמרות הוא מעוניין לתת, אך לא ציין העדפה ליום/משמרת ספציפית.

כתוצאה מכך אכן קיבלנו מ 2 המודלים הראשונים פתרונות אופטימליים כמצופה, כאשר זמן הריצה היה נמוך מאוד עבור 2 מודלים אלו.

לאחר מכן, הפעלנו את המודלים האיטרטייבים. באופן זה קיבלנו כי המודל שעובד לפי **random restart** הצליח להגיע לערך האופטימלי (כלומר לתוצאה של 0) לאחר 4 איטרציות.

כמו כן, עבור המודל של **simulated annealing** קיבלנו את הערך האופטימלי, לאחר הרצה של 3100 איטרציות. בנוסף, כאשר הרצנו את **Genetic algorithm** קיבלנו את התוצאה האופטימלית לאחר פחות מ 1000 איטרציות.

בדיקה מספר3:

בבדיקה השלישית שערכנו, בדקנו את סידור המשמרות שמתקבל עבור 10 ברמנים כאשר בקשות המשמרות של הברמנים יוצרות מצב בו בימים שני ושלישי עבור המשמרת בוקר ישנם רק 2 ברמנים שיכולים לעבוד (וכידוע השמה זו אינה חוקית שכן בכל משמרת נדרשים לפחות 3 ברמנים, וזהו כידוע אילוץ קשה). הכנסנו את משמרות אלו כהשמות לנתונים בתוכנה שיצרנו, וכך קיבלנו את הסידורים הבאים שהתקבלו בהרצת המודלים השונים:

- עבור המודל של אחוז משרה מקסימלי:

במקרה הנ"ל קיבלנו מצב בו ישנם אילוצים קשים שלא יכולים להתקיים (כיוון שבימי שני ושלישי במשמרת בוקר, לא הייתה אופציה ליצור השמה בה נקבל 3 עובדים לפחות ושבמקביל השיבוץ יישאר חוקי בהתאם לייתר האילוצים הקשים הקיימים במערכת).
לכן התוצאה שהתקבלה במקרה הנ"ל הינה שאין שאין שיבוץ חוקי שיצליח לספק את כל אילוצים הקשים במקרה זה ולכן לא ניתן לבנות לו"ז משמרות כפי שניתן לראות מהפלט שהתקבל:

Bartenders Work Schedule

Shift	Sunday	Monday	Tuesday	Wednesday	Thursday
Morning-Evening					
Evening-Noon					

- עבור המודל של סיפוק אילוצים (**constraint satisfaction**):

במקרה הנ"ל המודל אכן הצליח לספק פתרון אופטימאלי שעונה על סיפוק האילוצים הקשים והרכים, שכן המודל אפשר שיבוץ של ברמן גם במשמרות שאינן בהכרח המשמרת הספציפית שביקש (אלו הם המסומנות בצבע אדום), ובכך ניתן לראות בברור באילו פעמים המודל נתן לברמנים השונים לעבוד במשמרות שלאו דווקא ביקשו, וזאת על מנת לספק את האילוץ של כמות המשמרות שכל ברמן ביקש, ולספק את האילוץ שיהיו בכל משמרת לפחות 3 ברמנים.
ניתן לראות שכעת גם בימים שני ושלישי עבור משמרת בוקר נמצאה השמה חוקית.
בדוגמה שלנו, ניתן לראות כי המודל חילק 2 משמרות לברמנים שונים בשעות/ ימים שאינם ביקשו, כך למשל עבור ברמן מספר 7 ניתנה משמרת בוקר ביום שני (למרות שלא ביקש כלל לעבוד ביום זה) וזאת כיוון שרצה לעבוד 4 ימים במהלך השבוע (אך בסידור הציב השמה שאינה מתיישבת עם האילוצים הקשים, שכן הברמן ביקש לעבוד במשמרות: ביום שלישי משמרת בוקר, ביום רביעי משמרת בוקר וערב, וביום חמישי משמרת ערב, אך כמובן שהדרישה הנ"ל אינה מקיימת את האילוצים הקשים) וכמובן בגלל שלא היו 3 ברמנים שיעבדו ביום שני במשמרת הזו. כמו כן, לברמן מספר1 ניתנה משמרת בוקר ביום שלישי (למרות שלא ביקש כלל לעבוד ביום זה) וזאת כיוון שרצה

לעבוד 5 ימים במהלך השבוע (אך בסידור גם הוא הציב השמה שאינה מתיישבת עם האילוצים הקשים) וכמובן בגלל שלא היו 3 ברמנים שיעבדו ביום שלישי במשמרת הזו ולכן המודל הנ"ל יצר שיבוץ בו הברמנים הללו עובדים גם בימים שני ושלישי בהתאמה בשעות הבוקר, על אף שאינם סימנו שמעוניין לעבוד בימים אלו, ובכך המודל אכן עמד באילוץ של כמות המשמרות שברמן ביקש ובאילוץ המרכזי של לפחות 3 ברמנים במשמרת כפי שהוגדר.

באופן זה קיבלנו את השיבוץ הנ"ל:

Bartenders Work Schedule

Shift	Sunday	Monday	Tuesday	Wednesday	Thursday
Morning-Evening	Bartender1 Bartender2 Bartender3	Bartender1 Bartender3 Bartender7	Bartender1 Bartender3 Bartender7	Bartender1 Bartender2 Bartender3	Bartender1 Bartender2 Bartender5 Bartender6
Evening-Noon	Bartender8 Bartender9 Bartender10	Bartender5 Bartender8 Bartender9	Bartender4 Bartender6 Bartender8 Bartender10	Bartender4 Bartender7 Bartender8	Bartender4 Bartender7 Bartender8

- **טיפוס הרים (Hill climbing):**

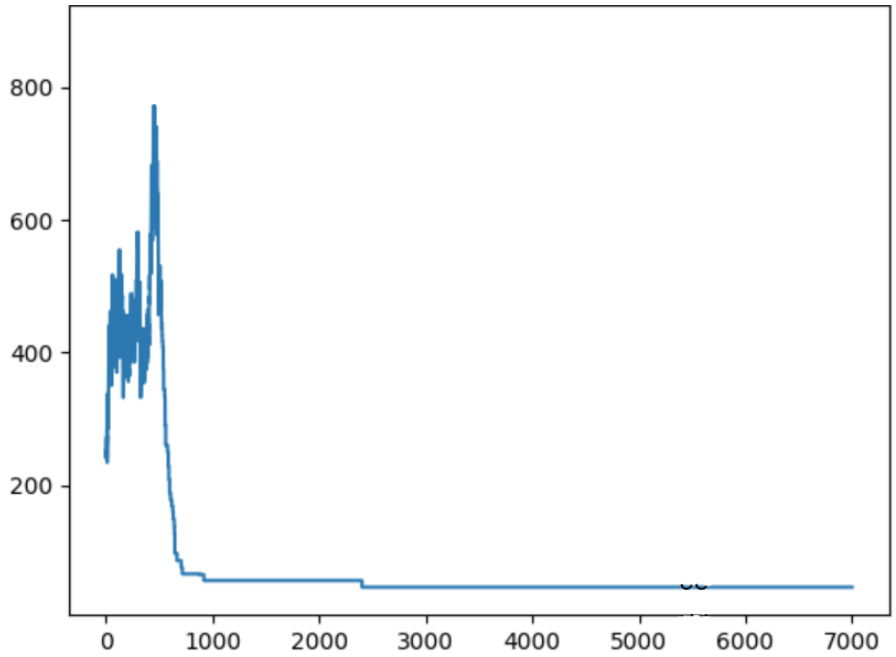
השתמשנו במודל ה **Hill climbing** ויצרנו גרף מהנתונים שאספנו. מהגרף המתקבל ניתן לראות כי המודל התחיל עם ערך שגבוה מ 270, והגיע בסופו של דבר לערך של 21, כלומר אכן קיבלנו שיפור שבא לידי ביטוי בגרף.

- **simulated annealing :**

כאן, הרצנו את המודל 6900 איטרציות, שבסופם התקבל הגרף הנ"ל. כפי שניתן לראות, המודל התחיל עם ערך התחלתי של מעל 220 ובסוף התכנס לערך של כ 40.

בנוסף, אפשר לראות מהגרף כי בתחילת הריצה הגרף מאוד רועש (שבשיאו הגיע לערך של 770).

תופעה זו הגיונית ומתאימה לאופיו של המודל הנ"ל בו מאפשרים "בחירות רעות", כלומר מעבר לנקודות גבוהות יותר וזאת על מנת למצוא אזורים חדשים שיאפשרו לנו לשפר, ובכך למנוע מצב של **min/max** מקומי.



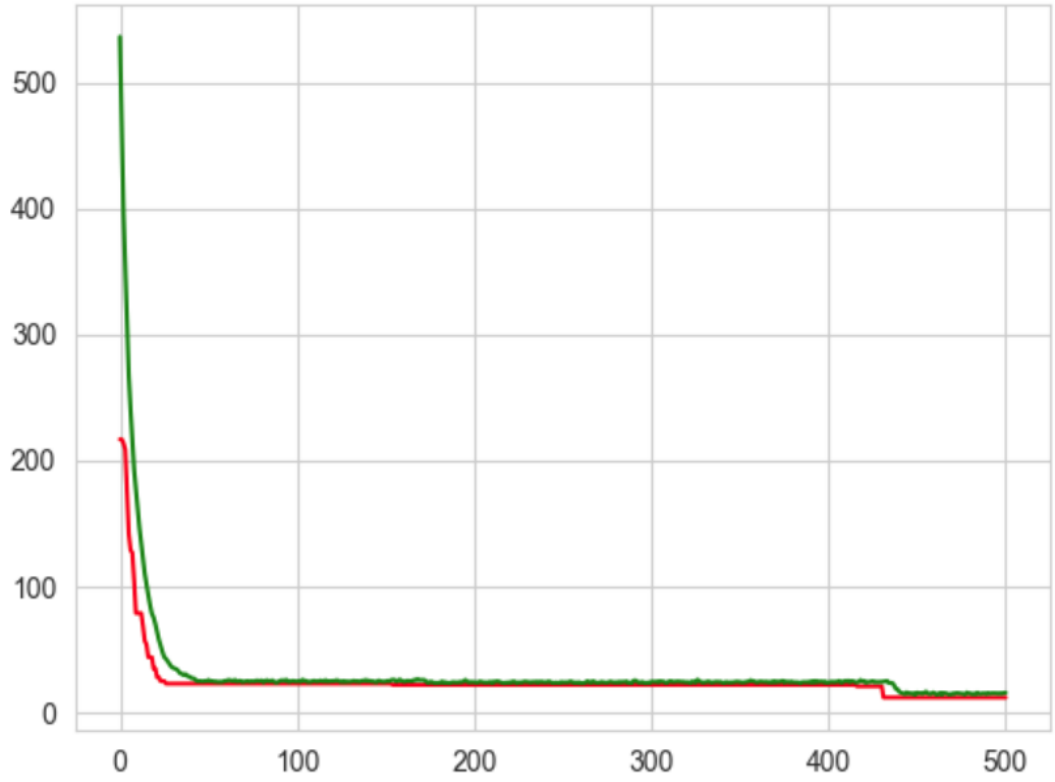
- טיפוס הרים (**Hill climbing**) עם **random restart** :

ביצענו הרצה של המודל הנ״ל עם השיפור ש **random restart** מספק למשך 35 איטרציות. שוב, זמן ריצת הפעולה גדול משל קודמיו, אך קיבלנו כי הערך הסופי הינו 12 (תוצאה שנחשבת כמעט אופטימאלית) ושמה את המודל הנ״ל בראש הבדיקה.

- האלגוריתם הגנטי (**Genetic algorithm**):

הרצנו את מודל זה למשך מעל 450 איטרציות פעולה שגם כאן אכן לקחה זמן מה אך בסופן קיבלנו תוצאה של 16, כלומר תוצאה שקצת פחות טובה מהתוצאה שקיבלנו באלגוריתם של טיפוס הרים עם **random restart** אך כן מתקרבת לתוצאה האופטימאלית, כאשר עיקר השיפור מתבצע במשך 42 האיטרציות הראשונות בהם הייתה קפיצה מערך של 570 לערך של פחות מ 45, ולאחר מכן השיפור הינו מינורי יחסית לכמות האיטרציות, דבר המעיד על כך שבתחילת ההרצה קל יותר למודל ליצור את ״המוטציה״ מהפתרונות ״החזקים״ אשר מביאה לשיפור, ואילו בהמשך ההרצה ובפרט לקראת סוף הרצת כל כמות האיטרציות קשה יותר למודל ליצור את המוטציה הרצויה לכן יש האטה בקצב השיפור, דבר אשר גורם למודל להתכנס בקצב איטי יותר אל התשובה האופטימאלית אותה מנסה להחזיר. באופן זה, כאשר המשכנו את הריצה לעוד איטרציות הערכים אומנם השתפרו בקצב איטי אך כן היה שיפור, עד אשר הערך האופטימאלי שהתקבל מהרצת המודל היה 13 אך הדבר דרש ריצה ארוכה של למעלה

מ5000 איטרציות (כלומר שיפור יחסית קטן מאוד יחסית לכמות האיטרציות שהוספנו מהרצת ה 450 איטרציות הראשונות).



מסקנות:

במהלך ביצוע העבודה יצרנו מודלים שונים ובדקנו את הביצועים והתוצאות שהם הניבו עבור בקשות סידורי עבודה שונות של הברמנים באמצעות הרצת דוגמאות שמתארות תרחישים שונים.

מהמודלים השונים שבנינו ומההרצות השונות שהפעלנו עליהן, קיבלנו שכולם אכן פועלים בצורה טובה אשר עוזרת לנו לפתור את בעיית סיפוק האילוצים זו, אך יש לבחור בחוכמה את המודל הרלוונטי איתו נרצה להשתמש בהתאם למקרה הספציפי והבעיה אותה אנו רוצים לפתור. באופן זה הבנו שעבור מקרה בו נרצה מודל המתנהג באופן גמיש יותר אשר מנסה להקל על הברמנים שעובדים בחברה ומנסה לספק ככל הניתן גם את האילוצים של בקשות השיבוץ מטעם הברמנים השונים, ותוך כדי שואף ליצור מספר מקסימלי של שיבוצים למשרות השונות, נעדיף להשתמש בודל הראשון של אחוז משרה מקסימלי.

לעומת זאת, כאשר נרצה מודל נוקשה יותר ששם לו כמטרה לספק בראש ובראשונה את האילוצים מטעם חברת הבר (האילוצים הקשים), מודל הדואג שכל ברמן יקבל בדיוק את כמות המשמרות שביקש/

מחויב לעבוד בשבוע, וכמו כן לאחר מכן גם שואף להביא למינימום את כמות המשמרות שעובדי הבר קיבלנו על אף שלא סימנו את המשמרת הנ״ל בסידור העבודה, אזי נרצה להשתמש במודל השני של ה **constraint satisfaction**.

כמו כן, כמובן שתמיד נעדיף להשתמש בשני המודלים הראשונים האלו שהצגתי, אשר מצליחים להביא פתרון אופטימאלי, כל עוד אכן יש השמה שפותרת ועונה על כל האילוצים הקשים, שכן אלגוריתמים אלו מצליחים להחזיר פתרון אופטימאלי במקרים שכאלו עם ביצועים טובים מאוד.

לעומת זאת, במקרים בהם אין השמה שפותרת ועונה על כל האילוצים הקשים נרצה להשתמש דווקא באלגוריתמים האיטרטיבים שהיו במודל הטיפוס הרים (**Hill climbing**) עם **random restart** ומודל ה **Genetic algorithm** אשר מחזירים פתרון ששואף לפתרון האופטימאלי, שכן בהינתן השמה וכמות איטרציות נכונה וסבירה, אלגוריתמים אלו מצליחים להחזיר פתרון שמתקרב מאוד לפתרון האופטימאלי וגם הביצועים שלהם נחשבים טובים.

כמו כן, מהדוגמאות שהרצנו עולה כי התוצאות המתקבלות מהרצת המודלים השונים עם השמות שונות, הינן תמיד בשלמים (הדבר נכון גם כאשר הרצנו עם השמה המכילה מספרים לא שלמים).

נשים לב כי האלגוריתמים האיטרטיבים, אשר מספקים תשובה ששואפת לתשובה האופטימלית, הביאו ערכים שמתקרבים מאוד לתוצאה הרצויה , כלומר הביצועים שלהם היו טובים בהחלט (כמובן עבור הצבת כמות איטרציות מתאימה ורלוונטית בהתאם לכל אלגוריתם), לכן כאשר יש תרחיש בו לא כל האילוצים הקשים מתקיימים שימוש באלגוריתמים אלו הינו פתרון הולם ונכון בהחלט.