Ant colony optimization

by Jonathan Fibush & Elad Israel

<u>הבעיה:</u>
בעיית הסוכן הנוסע (Travelling Salesman Problem- TSP) היא בעיה ידועה בתורת הגרפים ובתורת הסיבוכיות, המעלה את השאלה הבאה: "בהינתן רשימת ערים והמרחק בין כל שתי ערים, מהו המסלול הזול ביותר, אשר יעבור בכל עיר פעם אחת, ויחזור לעיר ממנה התחיל?"
הבעיה נכללת במחלקת הסיבוכיות של בעיות שהן NP-קשות, והיא אחת מהבעיות המרכזיות בתחום האופטימיזציה.

<u>קלט:</u>
גרף ממושקל מכוון ושלם(כל זוג צמתים מחוברים בקשת), בו כל עיר היא צומת, שביל בין עיר לעיר היא קשת, והמרחק בין כל שתי ערים מצוין על ידי משקל הקשת המחברת ביניהן.

אנחנו בחרנו להתמקד בגרסה הא-סימטרית של הבעיה(יותר קשה מסימטרית), בה ייתכן ודרך לא תתאפשר לשני הכיוונים, או שהמרחק בכיוונים שונים יהיה שונה. דוגמאות לדרכים בהן הבעיה היא א-סימטרית בעולם האמיתי הן כבישים חד סטריים, נתיבי תעופה בין ערים עם זמני הגעה או עלויות שונות, תאונות דרכים ועוד. במקרה שלנו ניתן להגיע מכל עיר לכל עיר, אך המרחק מעיר א' לעיר ב' ייתכן ששונה מעיר ב' לעיר א'.

<u>פלט:</u>
המסלול עם המשקל הנמוך ביותר, שיצא ויחזור לאותו צומת לאחר שעבר בכל צומת אחר פעם אחת בדיוק.
כלומר בהינתן גרף ממושקל שלם מכוון, מהו המעגל ההמילטוני שמשקלו הוא הקטן ביותר.

כדי להפוך את הבעיה ליותר 'מעניינת', בנוסף לTSP הרגיל מימשנו וריאנט נוסף- Optional TSP.
בוריאנט זה חלק מהערים יהיו חובה וחלק לא. הסוכן חייב לעבור בערים שהם mandatory, אבל ישנן ערים נוספות שהוא יכול להגיע אליהן. מעבר דרך ערים אלו יכול לעזור לו ולקצר את המסלול או דווקא להאריך אותו.

<u>פתרון:</u>
הקווים המנחים לטיפול בבעיות NP-קשות הם:
- תכנון אלגוריתמים למציאת פתרון מדויק (יעבדו בזמן סביר רק עבור קלטים קטנים)
- תכנון אלגוריתמים היוריסטים למציאת קירובים טובים לפתרון, אך לא פתרונות מדויקים מוכחים.
- מציאת מקרים פרטיים של הבעיה להם קיימים פתרונות מדויקים.

אנו בחרנו להתמקד באלגוריתם היוריסטי בשם "אופטימיזציית קן הנמלים" על מנת למצוא פתרון מקורב לבעיה:

בעולם הטבע הנמלים בתחילה משוטטות באופן אקראי על פני הקרקע בסביבת הקן. אם נמלה מוצאת מקור מזון היא נושאת ממנו חלק ומותירה שובל של פרומונים לאורך המסלול שבין הקן למקור המזון. נמלים משוטטות אחרות שנתקלות בשובל עשויות לבחור להמשיך לטייל לאורכו בסבירות גבוהה בהתאם לרמת הפרומון לאורך השביל. כל נמלה שמתווספת לאורך השביל מחזקת את כמות הפרומון שבו. כאשר מתכלה המזון, נמלים ימשיכו לשוטט באופן אקראי לכיוונים אחרים ולא לחזור על עקבות הפרומון לאורך השביל ולאחר זמן הוא יתנדף. בצורת הסריקה הזו, נמלים יעדיפו לטייל לאורך מסלולים קצרים יותר במרחק היות שתדירות מעבר הנמלים בהן גבוה יותר בהשוואה למסלולים ארוכים במרחק ולכן ריכוז הפרומון בהם ישמר ברמות גבוהות יותר בהתאם.

בהקבלה לבעיה שלנו:
הסוכן-נמלה צריך לבחור את העיר הבאה בכל צעד, ולבקר בכל עיר פעם אחת בדיוק.

בכל סיבוב יוצאת לדרך קבוצה של סוכני נמלים. ככל שהעיר רחוקה יותר, הסיכוי שסוכן יבחר אותה קטן יותר. עם זאת, ככל שיש יותר פרומונים על הדרך בין שתי ערים, הסיכוי שהסוכן-נמלה יבחר בעיר הזאת גדל. צורת הפקת הפרומונים שונה מעט מזאת של הנמלים, שכן לסוכנים ממוחשבים יש יתרון – הם יודעים מה אורך המסלול שהם עשו, והאורך משפיע על הפרומונים שהם משאירים.

בדומה לנמלים- לאחר שהגיע לעיר האחרונה, הסוכן מפקיד פרומונים על כל צעד מהדרך שעבר, כך ככל שהמסלול היה קצר יותר הוא יפקיד יותר פרומונים בכל צעד מהדרך. בכל "סיבוב" כמה סוכנים עובדים במקביל, ובסופו הפרומונים מתאדים מעט (כמו שקורה בטבע) וערכם קטן. כך יוצא שככל שמשתמשים פחות במסלול מסוים, הסיכוי שישתמשו בו בעתיד נמוך יותר, ולהפך.

כדי להפוך את גישת הפתרון ליותר 'מעניינת', בנוסף לACO הרגיל מימשנו וריאנט נוסף אשר קראנו לו Max ACO.
בACO רגיל אנו מנסים למזער(להביא למינימום) את עלות המסלול כולו(סכום המרחקים בין כל הערים בהן הנמלה טיילה במסלול). בוריאנט הMax מחיר המסלול מוגדר להיות המחיר המקסימלי בין זוג ערים שהסוכן ביקר בהן בסמיכות, כלומר ה"קשת" הכבדה ביותר בקרב קשתות המעגל ההמילטוני.

לבסוף נשלב את 2 הוריאנטים- Optional TSP עם Max ACO לכדי וריאנט חדש- Combined.
בוריאנט זה גם הבעיה שאותה אנחנו רוצים לפתור יכולה להכיל ערים אופציונליות, וגם בדרך הפתרון ננסה למזער את המרחק הגדול ביותר בין הערים במסלול.

<u>נתחיל מניתוח כללי של יתרונותיו וחסרונותיו של ACO:</u>

יתרונות ונקודות החוזק של ACO:

- ✓ קירוב פתרון של אלגוריתמים NP-קשים!
- ✓ יכולת חיפוש מקבילית בצורה פשוטה.
- ✓ התכנסות מובטחת.
- ✓ מוצא פתרונות טובים במהרה!
  מנגנון פידבק חיובי- תזוזה של נמלה בכיוון הנכון יגרום לנמלים אחרות לעקוב אחריה (בגלל הפרמונים) ובכך להגדיל את ההשפעה החיובית של השיפור בפתרון שהנמלה מצאה.
- ✓ יכול לשמש לפתרון בעיות דינמיות בזכות יכולת הסתגלות מהירה לשינויים כגון הוספת/מחיקת קשתות ועדכון משקלים.
  לדוגמא כשמעדכנים קשת עם עלות גבוה להיות נמוכה יותר, בשלב כלשהו הנמלים המשוטטות יזהו שיש דרך יותר קצרה ללכת בה וילכו בה, והנמלים האחרות יעקבו אחר הפרומונים שלהם וכך המסלול יתקצר.
- ✓ אמין וקל לשילוב עם אלגוריתמים אחרים

חסרונות ומגבלות הכוח של ACO:

- ✗ כולל החלטות רנדומליות שעלולות להסיט מהפתרון האופטימלי ולשנות את התוצאה בין איטרציות
- ✗ זמן ההתכנסות אינו ידוע מראש.
- ✗ זמן ריצה $O(n*(n-1)*m*T/2)$ ולכן פחות מתאים לבעיות מקנה מידה גדול.
  בנוסף, בבעיות גדולות מידי נמלים אינדבידואליות יטיילו בצורה רנדומית, מה שיפחית את יעילות האלגוריתם ויקח זמן רב למציאת האופטימום.
- ✗ הרבה היפר-פרמטרים. קשה לכוון את כולם כך שנקבל את הפתרון האופטימלי, וכיוון אינו נכון יכול להוביל לאופטימום מקומי, התכנסות מוקדמת העלולה לגרום לפספוס הפתרון האופטימלי.

תוצאות:

הפרמטרים בבדיקה זו:

מס' נמלים: 10

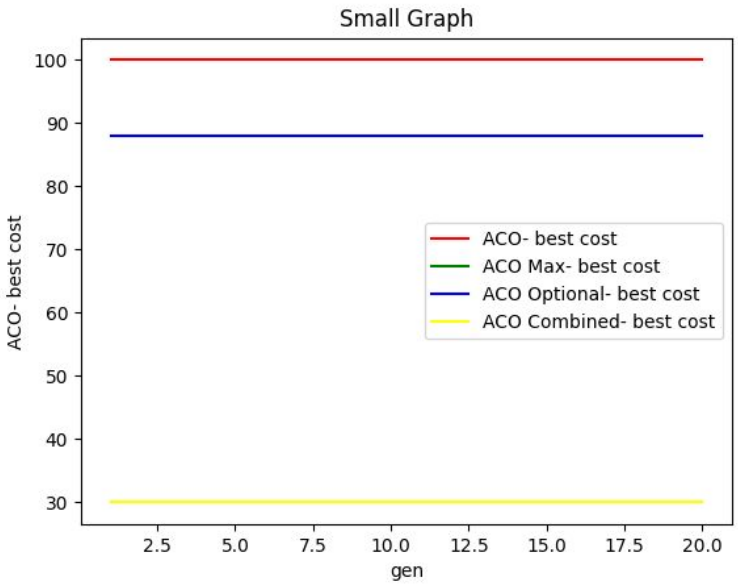
מס' איטרציות (דורות): 20

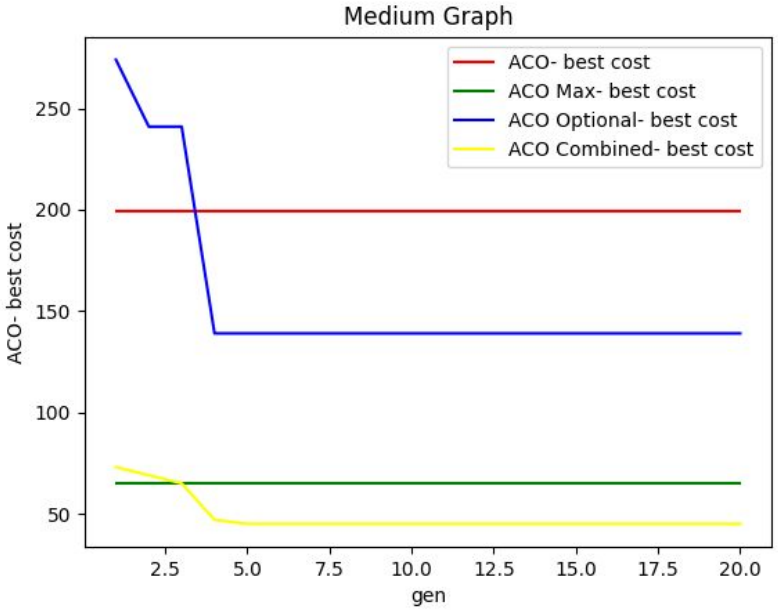
אלפא (חשיבות יחסית של הפרומונים): 1

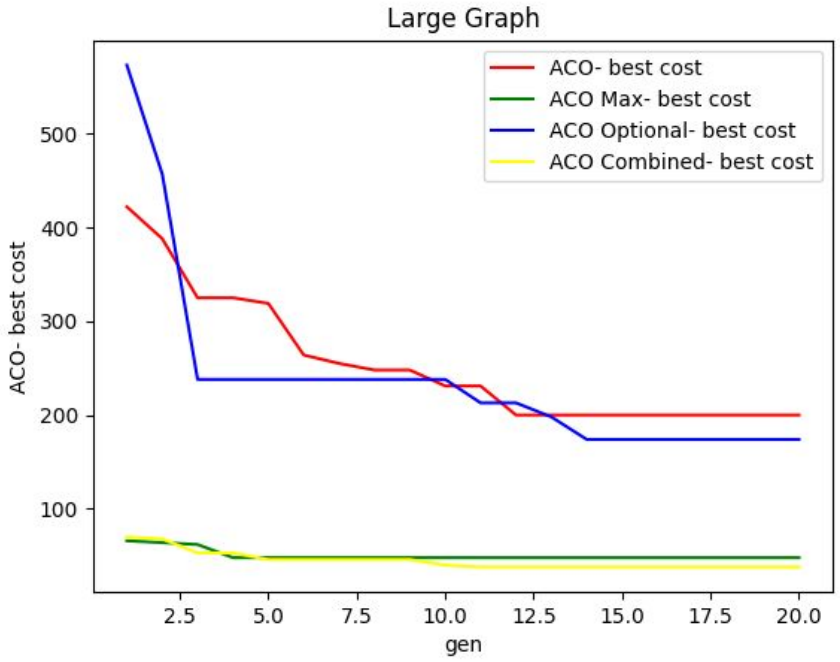
בטא (חשיבות יחסית של הפונקציה ההיוריסטית): 1

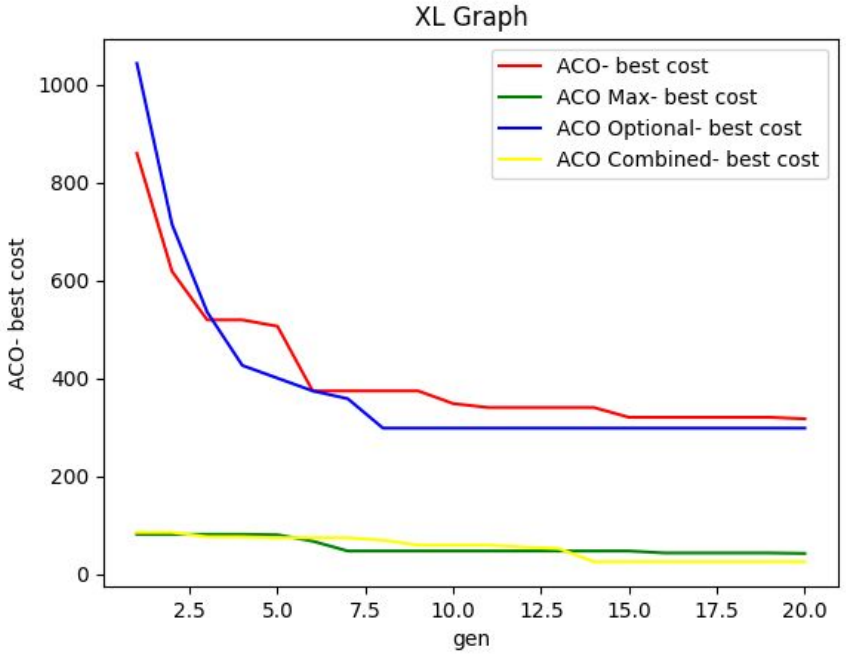
רו (בכמה מכפילים את הפרומונים מהאיטרציות הקודמות- בכמה הם 'מתאדים'): 0.5

אינטנסיביות הפרומונים: 1









הסבר התוצאות:

- **הוספת קודקודים אופציונליים מעלה את הcost ההתחלתי (בסבבים הראשונים) אך מורידה את הcost הסופי ומביאה לתוצאה טובה יותר.**

הסבר:

בכל הגרפים ניתן לראות שבאיטרציות הראשונות העלות של ACO Optional גדולה משל ACO הרגיל, זאת בשל העובדה שבגרף(/מטריצה) של Optional ישנם יותר

קודקודים מכיוון שאנחנו מוסיפים קודקודים אופציונליים לגרף המקורי, ולכן הנמלים בדורות הראשונים עוברות על גרף יותר גדול כאשר אין להם "רמז" לאן ללכת מכיוון שעוד אין פרומונים(או אין מספיק פרומונים בכיוון המסלול האופטימלי). כלומר באיטרציות הראשונות הוספת קודקודים אופציונלים רק מגדילה את הגרף ובכך רק מעלה את הcost.
לעומת זאת ככל שמתקדמים בדורות- הנמלים מנצלות את הדרכים הנוספות שניתנו להן ומשתמשות בהן לקצר את הדרך תוך כדי מעבר על כל הקודקודים הmandatory , ומדלגות על הדרכים האופציונליות שמאריכות את הדרך ולכן עלות המסלול שהן מוצאות מתקצר!

לכן:
כאשר אין באפשרותנו לבצע מספר איטרציות גדול- נעדיף שלא להשתמש ב Optional (כלומר להוסיף קודקודים אופציונלים מעבר לקודקודים שאנו חייבים לעבור עליהם במסלול).
לעומת זאת עבור מספר דורות גדול נעדיף להוסיף קודקודים אופציונלים לשיפור המסלול.

- **ככל שגודל המטריצה עולה- ההבדלים בין הביצועים של השיטות השונות מצטמצם.**

הסבר:
יש לשים לב שמדד הcost שונה עבור השיטות השונות: עבור ACO וACO Optional הcost של מסלול בו טיילה הנמלה הוא סכום המרחקים בין כל הערים בהן היא ביקרה, וזהו מה שאנחנו מנסים להפוך למינימום.
לעומת זאת עבור ACO Max וACO Combined הcost של מסלול הוא המרחק הכי גדול בו היא עברה במסלול. כלומר המסלול האופטימלי יהיה המסלול בו המרחק הגדול ביותר בין כל זוג קודקודים הוא המינימלי.

מכאן ניתן להבין את המסקנה הנ"ל: ניתן לראות שככל שגודל המטריצה עולה- הפער בין ACO וACO Optional, והפער בין ACO Max ובין ACO Combined מצטמצם.

- **ככל שגודל המטריצה עולה- לוקח לאלגוריתמים יותר זמן(יותר דורות) להתכנס לכיוון הפתרון האופטימלי. יוצא דופן הוא ACO Optional שעל גרפים גדולים מאוד הוא דווקא התכנס יותר מהר מגרפים בינוניים.**

הסבר:
עבור מטריצה קטנה מאוד לכל האלגוריתמים הספיקה איטרציה 1 כדי להתכנס על הפתרון האופטימלי!
עבור מטריצה בינונית כל האלגוריתמים התכנסו לאחר 1-4 דורות.
למטריצה גדולה לקח להם 4-14 דורות. למטריצה גדולה מאוד 6-16 דורות.