

דו"ח מעבדה 3

מעבדה בבינה מלאכותית

מגישים:

עדן טאוב , 315477794
עדן מورد, 316451012

הציגת הבינוסי:

1. הבעיות איתן התמודדנו בינויו:

- a. מציאת מינימום גלובלי (או קירוב) לפונקציית Ackley.
- b. מציאת אופטימנה גלובלית (או קירוב) לבעיית ה-CVRP - Constrained Vehicle Routing Problem.

2. האלגוריתמים שבעזרתם התמודדנו עם הבעיות הנ"ל:

- a. Multi Stage Heuristic
- b. Iterative Local Search, using the following meta-heuristics:
 - i. Tabu Search : TS
 - ii. Discrete Ant Colony Optimization: ACO
 - iii. Simulated Annealing : SA
- c. a Genetic algorithm utilizing the Islands Model, based on **LAB2**
- d. Adaptive Large Neighborhood Search (ALNS)
- e. Branch and Bound, using the Limited Discrepancy Search (LDS) Heuristic.

3. הדריכים בהן מדדנו את רמת ההצלחה של האלגוריתמים אותם מישנו:

- a. ערך פונקציית ה
 - עלות של הפתרון הסופי.
- b. ה

הפער

(בاقזים, או באופן יחס) בין הפתרון הסופי שנutan כל אלגוריתם לבעה, לבין הפתרון האופטימלי הנוכחי.
- c. זמן ריצה, לפחות 2 מדדים מרכזיים -
 - i. זמן ריצה עיבודי בלבד (CPU TIME)
 - ii. זמן ריצה כולל פלט וקלט, גרפיקות (Elapsed Time)
- d. פרט המסלולים - עבור כל רכב, רשימת אינדקסי הערים שבahn בירק (כולל depot)
- e. מספר כלי הרכב בשימוש עבור הפתרון - אומנם מספר המסלולים האופטימלי מצין כבר בקובץ הבעיה, אך ניסינו גם למצוא פתרון (קירוב) אופטימלי עבור מספר שונה של רכבים.
- f. מדדים סטטיסטיים -
 - i. מדדים סטטיסטיים - ערך ה-fitness הטוב ביותר, הממוצע, והגרוע ביותר לכל איטרציה של כל אלגוריתם.
 - ii. עבור בדיקת האלגוריתמים על פונקציית Ackley: 1. ערך מינימלי שהאלגוריתם הגיע אליו, ביחד עם הפתרון שנמצא, וזמן הריצה תחת ההגדרות הנוכחיות:

$$a = 20, b = 0.2, c = 2\pi, d = 10 \text{ and } x_i \in [-32.768, 32.768] \forall i \in [1, d]$$

מבוא -

- בעית ה-CVRP מאופיינת בכך בקייעת מסלולים לרכיבים בעלי קיבולת מוגבלת כך שכל הילקוחות יקבלו את המשלוחים, בשימוש במספר מינימלי של כל רכב, תוך ניצול מקסימלי של הקיבולת של כל רכב.
- **הקלט** לבעיה מתkeletal בצורה מטריצת מרחוקים בין הנקודות (המסלול מוגדר להיות מחסן → ללקוחות → מחסן), וכן הדרישות של כל 'ילקוח' מבחינה כמה קיבולת אותו צורך צריך לתפוס ברכב המשלוחים. וכמו כן, הקיבולת של כל רכב.
- **הפלט** הינו אוסף (רשימה) של מסלולים שבעצם מייצגת את המסלול של כל רכב בפתרון המוצע, שבוסףו כל רכב חוזר למיחסן. בנוסף, הפלט יכול גם את העלות הכלולת של הפתרון.
- **קושי חישובי** - כפי שראינו, בעית ה-CVRP שיכת לחלוקת הביעות שנן E-NP-COMPLETE. אך אין אלגוריתם הפותר את הבעיה ומבטיח אופטימום גלובלי, בזמן פרקטטי (פולינומי).
- **מטה-היוריסטיות** - בניסוי זה נעזרו בגישות שונות על מנת למצוא פתרונות איקוטיים (שושאפים לאופטיממה גלובלית) עבור בעיה זו, בזמן ריצה סביר, בהינתן החומרה שלנו (מחשבים אישיים, ממוצעים). בין הגישות שניסינו - ILS, GA, ALNS ועוד.
- **פונקציית Ackley** - זהה פונקציה רב ממדית עם נקודת מינימום גלובלית יחידה והמנן נקודות מינימום מקומיות. פונקציה זו שימשה אותנו כמבחן (Benchmark) לבחינת הביצועים של המטה-היוריסטיות בעת ניסין לצאת מואופטיממה לokaלית, בחיפוש אחר האופטיממה הגלובלית - במסגרת הזמן הסביר.

מטרת הניסוי -

1. המטרה של הניסוי היא לנסות למצוא אופטימום גלובל (או קירוב לאופטיממה גלובלית) בעזרת אלגוריתמי חיפוש ומטה-היוריסטיות, בשילוב (בחלק מהניסויים) היוריסטיות רב-שלביות. (Multi-Stage Heuristics, MSH)
2. שימוש בפונקציית Ackley ככלי לאיימות ובדיקה ראשוניים של האלגוריתמים וההיוריסטיות, לפני בדיקת האלגוריתמים על בעית ה-CVRP עצמה.

הערכת זמני ריצה של כל חלק בכל אלגוריתם-

הערות	סיבוכיות	פעולה	שלב
בסיס לכל השיטות (,,ALNS, ILS, GA, B&B	$O(n^2)$	Most Connected + Nearest-Neighbor Insertion	פתרון התחלתי עבור אתחול האוכלוסייה
B&B seeding ל-GA seeding ל-K-Means Seeding <i>לייצור אשכולות ללקוחות ראשוניים</i>	$O(n \cdot k \cdot \text{iter})$	K-Means Seeding	מייפוי אשכולות בעת אתחול
אחד מסלולים לפי חסכון במרחב	$O(n^2 \log n)$	Clarke-Wright Savings Seeding	אתחול חסכון לפי קלארק-רייט
הסרת K לקוחות (Ruin) ALNS-ב	$O(k)$	Random Removal	(Ruin)

הסרת לקוחות סמוכים על בסיס KNN	$O(n^2)$	Shaw Removal	הסרת שכנים (Shaw Removal)
הסרת הלוקחות שהסתרם מקסימלית את עלות המסלול	$O(n^2)$	Worst-Distance Removal	הסרת מרחק גראן
הכנסת לקוחות בנקודה הזולות ביותר לכל מסלול	$O(n^2)$	Greedy Insertion	הכנסה חמדנית
הוסף על בסיס הבדל עלויות בין k להכנסות הזולות ביותר	$O(k \cdot n^2)$	Regret-k Insertion	חרטת-k (Regret-k) Insertion
שיפור רצף פנימי במסלול MSH, ALNS, ILS, (GA, B&B ייחיד)	$O(n^2)$	Two-Opt	שיפור מקומי
העברה לקו ממסלול למסלול (ILS, ALNS)	$O(n^2)$	Relocate	העברה
החלפת שני לקוחות בין מסלולים (ILS, ALNS)	$O(n^2)$	Swap	החלפה
החלפת תתי-רצפים בין שני מסלולים (ILS)	$O(n^2)$	Cross-Exchange	החלפה נגדית
ח齊ית רצף בין שני הורים לבניית פתרון צאצא (GA)	$O(n)$	SCX Crossover	שילוב רצפים
החלפת זוג לקוחות אחד-אחד בין שני מסלולים (GA)	$O(n^2)$	Swap-Star	החלפת מאקרו
ליטוש סופי לשיפור כולל (GA)	$O(n^2)$	Full Polish (Swap-Star + Two-Opt + Merge)	ליטוש מלא
ליטוש קל לפני הכנסת פריטים חדשים (GA)	$O(n^2)$	Light Polish (Two-Opt בלבד)	ליטוש חלק
פתרון בעיית TSP ב- MSH	$O(n^2)$	Cheapest Insertions	בניית מסלול
פתרון בעיית TSP ב- MSH	$O(n^3)$	Christofides	בניית מסלול

הסבר קצר לכל אופרטור ואופן פעולהו

★ פתרון התחלתי

Most Connected Node + Nearest Neighbor

משתמשים בפונקציית `most_connected_node` לבחירת הלקוח הראשון, ולאחר מכן במבנה מסלול גרידית לפי הקרבה (`nearest-neighbor`). מספק פתרון התחלתי מהיר וaicותי בעל סיבוכיות $O(n^2)$, משמש בסיס לכל השיטות.

K-Means Seeding ★

קובץ הלקוחות לאשכולות על-פי מיקום הגיאומטרי כדי לחלק אותם לנטיים ראשוניים. שימוש ב-`scikit-learn KMeans` בהדרות קבועות, `k=iter(O)` מתאים לגנטורים המסתמיכים על `population-based`.

Clarke-Wright Savings ★

פועל על חישוב חיסכון זוגי לכל זוג ליקוחות ($O(n^2)$), מבנה מסלולים ע"י מיזוג כל זוג שמניב חיסכון מקסימלי, יוצר פתרונות Zarzim וaicותיים כתשתית $L-B&B$.

Random Removal ★

במהלך ALNS, הסרת k ליקוחות אקראיים ($O(k)$) מאפשרת גיון גבוה בבדיקות החיפוש ושבירה של מבני פתרון שהם כמעט אופטימליים.

Shaw Removal ★

מבוסס על קרבה: בוחרים ליקוח אקראי ואז מסירים $1-k$ מהליקוחות הקרובים אליו לפי `neighbor_dict`. יוצר קבועות הקשורות להחלפה מהירה במנגנון השיקום.

Worst-Distance Removal ★

עבר כל ליקוח בודדים, מחשבים את הגידול בעלות הסיבוב ללא הלקוח ובוחרים להסיר את אלה שייצרים את הגידול הגדול ביותר. מאפשר התמקדות בהסרות שיפיעלו השפעה ממשמעותית.

Greedy Insertion ★

לאחר השלב הרע (`ruin`), מכל נקודה בוחרים את הכניסה הזולה ביותר לפי $O(\Delta^2)$, מבטיח הכנסת ליקוחות חזקה במסלול הכי חסכוני.

Regret-k Insertion ★

מחשבים $-k$ האפשרויות הזולות ביותר, ומשווים את ההפרש (`regret`) בין האפשרות הזולה ביותר והשנייה וכן הלאה, ובוחרים את הליקוח עם החרטה הגדולה ביותר. משפר את מבנה הפתרון בטוויה הארוך.

Two-Opt ★

החלפה של מקטעים הפוכים בתווך המסלול לבדיקה האם נוצרת חסכה. לולאה עד איזיפור, $O(n^2)$, נמצא בכל המודלים לשיפור מהיר של מסלולים.

Relocate ★

העברת לקוח יחיד ממסלול אחד למסלול אחר במיקום אקראי, בדיקה לא-ઇחריגה מקיבולת. אפשר שינויים גלובליים קלים בפתרון, בשימוש ב-LSA ו-ALNS .

Swap ★

החלפת שני לקוחות בין שני מסלולים, תוך וידוא קיבולת, מאפשר חילופי נקודות המשפרים עלות כוללת. חלק בלתי נפרד מ-LSA ו-ALNS .

Cross-Exchange ★

בחירת תת-רצפים משני מסלולים והחלפתם במלואם, מאפשרת שינוי מבנה גדולים יותר מאשר swap רגילים .

SCX Crossover ★

ב-GA, מאהד רצפים משני הורמים: בכל צעד בוחרים את העוקב הקרוב יותר בין ההורים או לפי nearest feasible, יוצר צאצא מתואם היטב עם שניהם .

Swap-Star ★

בשלב המקביל ל-crossover, מוחפשים זוג לקוחות משני מסלולים שמייטיבו כדי הרבה את העלות כאשר מוחלפים, ובמציעים החלפה אחת בכל דור; מעלה גיוון סדק ותחרות בין מסלולים .

Full Polish ★

שילוב של Swap-Star, Two-Opt ויחוג מסלולים (merge) לאופטימיזציה מקיפה על פתרוןשלם, משמש בעיקר כ-seeding ו-elitism ב-GA .

Light Polish ★

גרסה מצומצמת של full polish שפעילה רק Opt-Two על כל מסלול, יעילה ומהירה לפני הכנסת פריטים חדשים לאוכלוסייה.

Cheapest Insertion ★

בנייה מסלול בשיטה greedy ע"י הוספת כל לקוח חדש למיקום הקיים שגורם לעלייה הקטנה ביותר בעלות. משתמשים ב-HSM כדי לפתור את TSP .

Christofides ★

במקרה של מספר clusters נמוך משתמשים כדי לפתור את TSP באופן שמותאם ל-CVRP. פועל על גרף מלא ע"י מציאת עץ פורש מינימלי, התאמת בזוגות מזירים, ויצירת מסלול איאלי-ריאני שמהווה קירוב של עד 1.5 מהאופטימום.

פרמטרים הנחוצים לשינוי בכל אלגוריתם שמייחבו -

ILS.py

- max_trials – כמה נסיניות שיפור עושם לפני שעוברים הלאה.
- nAnts – כמה "נמלים" מוחפשות פתרונות.
- th – מספר האיטרציות שבו פתרון נמצא בראשימת טאבו.
-

GA_with_islands.py

- POP_SIZE – גודל האוכלוסייה בכל אי.
- ISLANDS – מספר האיים (תת-אוכלוסיות).
- MIGRANTS – כמה פרטים עוברים בין איים בכל הגירה.
- MIGRATE_EVERY – בת כמה דורות עושים הגירה.
- GENS – כמה דורות רצים בסך הכל.
- FLOOR_MUT_BASE_MUT – שיעור המוטציה ההתחלתי והמינימלי.
- FLOOR_TWOOPT_BASE_TWOOPT – שיעור ה- $\frac{1}{2}$ -opt-סואן ההתחלתי והמינימלי.
- DECAY_RATE – כמה כל פרמטר קטן בכל דור.
- HYPER_LEN_HYPER_MUT, HYPER_TWOOPT – שיעורי מוטציה ו- $\frac{1}{2}$ -opt-סואן דור ההיפר.
- STAG_THRES – דורות ביל שיפור עד שימושיים מצב "הייר".
- NOVELTY_W – כמה לוקחים בחשבון גיון בבעיה.
- W_DIST, W_CAP, W_BAL – משקל למרחק, קיבולת ואיזון בכושר.
- CROSS_MODE – שיטת חצית גנים (למשל "scx").
- AGE_THRESHOLD – גיל מקס'ימום לפני שאתחל את הפרט.

BranchBound_LDS.py

- time_limit_sec – כמה זמן מרכיבים בסה"כ.
- max_discrepancies – כמה חריגות (LDS) מותרות.
- k_knn – כמה שכנים לוקחים בשלב החיפוש.
- relax_factor_init ו- relax_factor_final – כמה מרחיבים את הגבול התיכון בתחליה ובסוף.
- num_restarts – כמה פעמים מתחילה שוב מזיקה חדשה.
- seed_attempts – כמה זריקות שונות מנסים בכל שיטת seed.
- kmeans_iter – כמה איטרציות להרצת KMeans צדקה.
- weight_distance, weight_capacity, weight_balance – משקל למרחק, לקבולת ולאיזון בפונקציית הבסיס.

ALNS.py

- ALNS_ITERATIONS_DEFAULT – כמה איטרציות בסך הכל.
- REMOVE_FRACTION_BOUNDS_DEFAULT – טווח האחוזים להסרה בכל השמדת חלק.
- REMOVE_FRACTION_MAX – אחוז מקסימום שאפשר להסרה.
- REMOVE_FRACTION_ADAPT_STEP – בכמה משנים את אחוז ההסרה בזמן ריצה.
- EXTRA_TWOOPT_PROBABILITY_DEFAULT – סיכוי נוסף להפעיל two-opt אחרי השחזור.
- WEIGHT_DECAY_DEFAULT – כמה מצמצמים משקלים של השמדות/החזירות.
- WEIGHT_UPDATE_INTERVAL_DEFAULT – כל כמה איטרציות מעדכנים משקלים.
- KNN_NEIGHBORS_DEFAULT – כמה שכנים ב-alval Shaw Removal.
- REGRET_K_DEFAULT – k בשיטת "regret-k" להחזקה.
- TEMPERATURE_EPSILON ו- TEMPERATURE_INITIAL – טמפרטורה התחלתית ותקלה לעצירה בסימולט אנילינג.
- STAGNATION_THRESHOLD – כמה איטרציות בלי שיפור לפני שمعدכנים.
- VEHICLE_PENALTY – קנס על שימוש ביוטר רכבים מאשר המotor.

בעיות לדוגמה שפתרנו כדי לבדוק את נכונות ויעילות האלגוריתמים אותם מישנו -

קבוצה	שם קובץ	n	K	קיבולת	Opt
Beginner	P-n16-k8	16	8	35	450
Intermediate	A-n32-k5	32	5	100	784
Beginner	E-n22-k4	22	4	6000	375
Advanced	X-n101-k26	101	26	206	27591
Intermediate	A-n80-k10	80	10	100	1763
Advanced	M-n200-k17	200	17	200	1275

החומרה עליה הרכינו את האלגוריתמים -

עדן - מחשב אישי, ללא מאיץ גרפי, מעבד אינטל דור 9, 16GB זיכרון.
עדן - מחשב אישי, ללא מאיץ גרפי, מעבד אינTEL דור 10, 32GB זיכרון.

הציגת האלגוריתמים

MSH

ההיוריסטיקה היא MKP Two step heuristic based on clusters stages (רכבים) בニアוריסטייקות clusters stages (רכבים) בニアוריסטייקות.

הלקוחות שייכים ל-clusters (רכבים) באמצעות פתרון בעית MKP. במקרה של בעית CVRP פונקציית הערך כוללת:

1. עדיפות למרחק קרוב מהמחסן (depot)
2. חיסכון ממוצע עם שכנים
3. יחס ביקוש/קיבולת

כך מתאפשרת חלוקה שמקדמת איסוף חכם יותר שמירת מגבלות הקיבולת.

שלב שני: תכנון מסלול

TSP מוגדר כתת בעיה של cluster לאחר קיבוץ הלקוחות, כלומר עבור כל אשכול לקוחות שהוקצה לרכב מסוים צריך לפתור את TSP. עושים זאת בעזרת היוריסטייקות אלו:

1. Nearest neighbor
2. Cheapest insertion
3. Chirstofides
4. opt-2

כך נבחר המסלול הטוב ביותר לכל אשכול.

השיקולים מאחוריו ההיאוריסטייקה:

- פונקציית הערך משלבת מרחק, חיסכון וביקוש כדי להעדיף שיבוץ יעיל.
- שימוש במספר היוריסטייקות לפתרית TSP משפר את איכות המסלול ומעבירה את איזור הפתרון.
- עבור ACKLEY ניתן לבחור את מספר ה-clusters ולכן ניתן לגאון את מרחב החיפוש.
- עבור CVRP, ניתן להבטיח שכל רכב לא יחרוג מהמגבלה (קיבות) שלו בשל השימוש ב-MKP. בנוסף, לאחר שכל הלקוחות חולקו ל-clusters, כל cluster הופך לעית TSP קטנה יותר וקלה יותר לפתירה מאשר הבעיה CVRP הגדולה.

סיבוכיות:

- שלב 1: $O(k \cdot n^2)$ כאשר n הוא מספר הלקוחות ו- k מספר הרכבים
- שלב 2:

Nearest Neighbor - $O(n^2)$

Insertion - $O(n^2)$

Chistofides - $O(n^3)$

Opt -2 - $O(n^2)$

כאשר משתמשים ב-MSH בלבד הפתרון לרוב לא קרוב לאופטימום (במיוחד בעקבות קשות יותר). עם זאת, היא מספקת פתרון התחלתי חזק ובעזרתו ניתן לצמצם את זמן ההתקנסות באלגוריתמים אחרים כגון LS או עם מטה הייריסטיקות.

:**Most Connected Node + Nearest Neighbor** הפתרון התחלתי שהוא מפנה יותר טוב משיטת ניטות. ניתן לראות זאת בחלק של התוצאות.

ILS

הweeneyון המרכזי של LS הוא להתחיל מפתרון ראשוני ולשפר אותו על ידי חיפוש לוקאלי ואז להחזיר שינויים לפתרונות המתקבלים כדי לצאת ממינימום מקומי. התהליך חוזר על עצמו עד שנגמר זמן החיפוש.

Parallell_hybridils מרים את Tabu Search , Simulated Annealing , ACO בו זמנית במקביל על מנת לקבל את הפתרון הכי טוב ולהרחב את החיפוש.

• ACO

מבוסס על האלגוריתם שניתן בהרצאה. בכל איטרציה נבנית אוכלוסייה נמלים מייצרות פתרונות. כל נמלה בונה פתרון על פי הסתברות שימושית מידע פרומון ומרחקים. הפתרון הכי טוב שמייצרת האוכלוסייה עובר שיפור על ידי חיפוש לוקאלי והפרמטרים משתנים בהתאם (גם תוך כדי בניית האוכלוסייה וגם לפני שהיא נבנית חוץ באיטרציה הראשונה).

• מספר הנמלים מוחלט לפי גודל הבעיה -

```
nAnts = max(10, int(0.3 * num_nodes)) # Number of ants, at least 10  
                                or 30% of nodes
```

פרמטרים משתנים דינמית לאורך הריצה ורכיבי ההתחלתי:

- Alpha = 1.0
- Beta = 2.0
- Rho = 0.05
- Q = 1.0

- האלגוריתם משנה את הערכים באופן דינמי בהתאם למצב -
אם יש שיפור - exploitation •

```

if current_best_cost < best_cost:
    best_solution = copy.deepcopy(current_best)
    best_cost = current_best_cost
    # Exploitation phase - focus more on the pheromone
    alpha = min(3.0, alpha * 1.1) # Increase pheromone importance
    beta = max(1.0, beta * 0.95) # Decrease heuristic importance
    Q = min(5.0, Q * 1.05) # Increase pheromone deposit factor
    rho = max(0.01, rho * 0.95) # Decrease pheromone evaporation rate
    iteration_without_improvement = 0 # Reset counter if improvement is found
else:
    iteration_without_improvement += 1

```

- אם אין שיפור - exploration •

```

# If no improvement, adjust parameters for exploration
if iteration_without_improvement == max(5, nEpochs // 10): # Update parameters when stuck for 10% of epochs
    # Exploration phase - focus more on the heuristic
    alpha = max(0.5, alpha * 0.9) # Reduce pheromone importance
    beta = min(5.0, beta * 1.1) # Increase heuristic importance
    Q = max(0.5, Q * 0.95) # Decrease pheromone deposit factor
    # For large problems, use faster evaporation
    if num_nodes > 50:
        rho = min(0.2, rho * 1.5) # Increase evaporation rate
    else:
        rho = min(0.5, rho * 1.05) # Increase pheromone evaporation rate

```

שיפורים שנוספו לאלגוריתם:

- רנדומליות לפתרונות שהנמלים מייצרים כך שmedi פעם הם לא מסתמכים על מידע פרומון
ומרחקים •
- אתחול השבילים של הפורומון (באופן חלקי, נשמר חלק מהזיכרון) באופן רנדומלי -
randomly reset pheromone trails •
- pheromone smoothing to level path differences* •
- הערכים של הפרמטרים משתנים דינמית לפי מצב הפתרון •
- אתחול הפתרון ההתחלתי לפתרון הכל' טוב והרצה של `sco` ליריצות קטנות יותר. •
- באייטרציה הראשונה הפתרון ההתחלתי הוא MSH - אם הפתרון יציב, נכון וטוב יותר
מהשיטה האחרת. •

ה יתרונות של שימוש ב-ACO עבור CVRP:

- לעומת היוריסטיקות אחרות ACO מייצר אוכלוסייה של נמלים ולכך מספר הפתרונות המיוצרים בכל אייטרציה גדול בהרבה. •
- שילוב בין זיכרון (פרומוניים) למרחקים - מאפשר בניית פתרונות שמאזינים בין ניסיון קודם לאינטואציה חישובית. אם פרומון מצביע סביר רצפים טובים של לקוחות, זה עוזר בשימור תתי-מסלולים אפקטיביים. •

- COO טוב בזיהוי תבניות טובות גם אם לא פתרוןשלם, ומנצל אותן בבנייה עתידית.

- Simulated annealing

מבוסס על האלגוריתם שניתן בהרצאה.

шиיפורים שנוספו:

- פרמטרים משתנים דינמיות.
- אתחול פתרון אם אין שיפור לאחר מספר דורות קבוע מראש
- כל כמה דורות מתחלימים את הפתרון לנקודת התחליה שונה

הפרמטרים :

- טמפרטורה - בהינתן פתרון התחלתי, הטמפרטורה מתחילה כך שההסתברות לקבל פתרון גרוע היא 0.5 (אם זה לא עובד אז 10) .
- אלפא - 0.95

שינויי הפרמטרים באופן דינמי:

- אם אין שיפור - קירור : מורידים את הערך של אלפא והטמפרטורה תרד.
- אם יש שיפור - חימום : מעליים את הערך של אלפא והטמפרטורה עולה.

יתרונות על שימוש ב- Simulated annealing עבור CVRP:

- בשיטה זו אנו שולטים באופן מדויק על מצב חיפוש או ניצול, מאפשרים לאלגוריתם להחליט متى צריך לחפש ומתי צריך לנצל לפי התוצאות ברגע אמת. למשל, הטמפרטורה מאפשרת הסתגלות לאיוצים הקשים של CVRP.
- האלגוריתם מקבל פתרונות "גروعים" אך זה פותח את האפשרויות לחיפוש מרחבים אחרים ובעיות כגון CVRP זה מאפשר לצאת ממינימום מקומי.
- מתאים לשיפור פתרונות קיימים (لكن אפשר לראות שיפור בתוצאות שלא עם MSH).
- דבר הכרחי בעיות מסווג CVRP כי

- Tabu search

מבוסס על האלגוריתם שניתן בהרצאה.

шиפורים שונים:

- פרמטרים דינמיים

- במקרה שאין שיפור באופן רנדומלי נבחר האם לפעול בשיטת Intensification או Diversification . במקרה של Solution Intensification - נבחר הפתרון הכי טוב ברשימת טאבו או בסיכוי של 0.5 הפתרון הכי טוב עד כה גם אם הוא לא בטאבו. במקרה של Solution Diversification - מיצרים פתרון חדש (יכול להיות בטאבו או לא) ומשתמשים בו.

פרמטרים:

```
# Dynamic upper bound based on problem size
    max_tenure = min(50, max(20, int(np.sqrt(num_nodes) * 3)))
# Scale initial tenure with problem size
    n = min(max(7, int(np.sqrt(num_nodes) * 1.5)), 20)
```

שינויי פרמטרים:

- אם יש שיפור - הזיכרון מצטמצם
- אם אין שיפור - הזיכרון גדל

ה יתרונות של Tabu Search על CVRP:

- רשימת טאבו מונעת מהלכים חוזרים ولكن לא נתקיים על פתרון מסוים. בעיות כמו CVRP חשוב לשמור על גיון פתרונות.
- האלגוריתם נוע בין מצב של חיפוש וניצול על ידי שליטה בגודל הזיכרון.

ה יתרונות בהרצאת שלושת מטה היוריסטיות עם SLS:

- שלושת היוריסטיות רצות במקביל - מרחיב את מרחב החיפוש באופן משמעותי על ידי שימוש ביחידות של כל מטה היוריסטיקה וכן ניתן לפטור בעיות גדולות כמו CVRP.
- הפרמטרים משתנים דינמית וכן האלגוריתם מתאים את עצמו לכל מצב שבו הוא נמצא, אם כל הזמן הוא משתפר אז האלגוריתם מתרץ במצב ניצול ומנסה להמשיך לשפר את הפתרון הכי טוב, אם אין שיפור אז האלגוריתם מרחיב את אזור החיפוש.
- שיפור מקומי לפתרונות

GA with Islands

מבוא

מודל ה-GA במודל האיים מבוסס על פירוק האוכלוסייה הראשונית למספר איים (תתי-אוכלוסיות) שפועלים במקביל ומוחלפים ביניהם פרטימ בתדרות מוגדרת. כל אי מתנהל אוכלוסייה נפרדת – אתחול (אקראי או מונחה, למשל באמצעות KMeans), ברירה גנטית, crossover, מוטציות (swap, relocate, inversion) ושיפור לוקלי (2-opt, מיזוג מסלולים). מדי כמה דורות מבוצעת הגירה של אחוז קטן מהאוכלוסייה לפי טופולוגיה מבוקרת (טבעת, crossover, star fully-connected ועוד), מה שמאפשר לשלב בין שימור גיאון לבין העברת מידע מבטיח בין איים.

עקרונות הפעולה

- אתחול: לכל אי אתחול עצמאי, כשהבחירה בין אתחול אקראי או מונחה (KMeans) יכולה לקדם חקירה מכוונת של אזורים מבטחים.
- איטרציה גנטית: בכל דור מבוצעים שלבי ברירה, רבייה (crossover), מוטציות אדפטיביות (עם אפשרות להיפר-מוטציה כישיש stagnation) ושיפור לוקלי (2-opt, relocate, merge-routes) להאצת הניצול המקומי.
- הגירה: בתדרות ובסדר שנקבע מראש (או אדפטיבית), מעברים פרטימ מבטחים בין איים על בסיס טופולוגיה מבוקרת, לעיתים באופן אסינכרוני כדי למנוע בלוקים.

Exploration VS Exploitation

חקירה (exploration) היא היכולת לגלוות אזורים חדשים במרחב הפתרונות, ניצול (exploitation) מתמקד בשיפור אינטנסיבי של הפתרונותות שכבר נמצאו.

- שיעור הגירה גבוה ושיעור מוטציה מוגבר יוצרם גיאון רב וחקירה נרחבות: כל אי חוקר מרחבים שונים במקביל בלי להתנגש עם האחרים.
 - שיפור לוקלי תדרי וברירה מוקדدة (strong selection pressure) מובילים לניצול עיל של נקודות אופטימום מקומיות.
- האתגר הוא למצוא את נקודת האיזון: יותר exploration מאיטים את התכנסות, יותר exploitation מגדים את הסיכון להיתפס במינימום מקומי. באמצעות מדידה דינמית של גיאון (entropy גנטית, מגוון גנים) ניתן לכוון פרמטרים אדפטיביים כמו תדרות ההגירה, גודל ההגירה ושיעור ההיפר-מוטציה.

פתרונות

- שמירה על גיון גבוה שמנעת תקיעות מוקדמות
- בריחת מינימום מקומי בזכות הגירה והיפר-מוטציה
- יכולת מקבילית טבעית (distributed multi-core או distributed)
- גמישות בבחירה טופולוגית הגירה ופרמטרים אדפטיביים

חסרונות ואתגרים

- ניהול פרמטרים מורכב (מספר איים, תדיות/פודל הגירה, שיעורי מוטציה)
- עלות חישובית גבוהה יותר לעומת GA שטוח
- תיאום הגירה בסביבות מבזירות עלול ליצור צוואר בקבוק
- ערך מדויק של מדדי גיון דורש חישוב נוסף

התאמת ל-CVRP

לביעות כמו CVRP, שבהן מרחב הפתרונות עצום ומטען הלוקחות מגביל את המסלול, מודל האיים מאפשר:

- חיפוש לוקלי באמצעות 2-opt ו-merge-routes לשיפור מהיר של כל מסלול
- שמירה על תת-אוכליות הממקדות באזוריים שונים של המרחב
- העברת פתרונות מבטיחים בין איים תוך שמירה על מגבלות הקיבולת

מדוע לא אימצנו גישה ממטיית

ויתרנו על Memetic Algorithm כדי להימנע מעלות חישובית גבוהה בכל דור ושמירה על סקלබיליות מקבילית – שילוב של local search כפעולה ב-island-model נזון ניצול מהיר בלבד ביעילות.

מסקנות

מודל ה-GA באים מספק מסגרת חכמה לאיזון בין exploration ל-exploitation, מאפשר בრיחת מינימום מקומי ושומר על גיון עמוק. התאמת דינמית של פרמטרים ועיצוב טופולוגיה הגירה נכונה תורמים להשגת פתרונות איכותיים ומהירים בעיות קומבינטוריות כמו CVRP.

ALNS

מבוא

ALNS (Adaptive Large Neighborhood Search) הוא מטודיה היוריסטיקת המבוסס על גישת Ruin & Rebuild, המשלבת בחירה אדפטיבית של אופרטורים עם קרייטריון קבלה על בסיס Simulated Annealing. המטרה: לאפשר חילוף בין אינטנסיביות חיפוש (exploitation) לבין פתיחות לחקר אזוריים חדשים (exploration), ולברוח ממינימום מקומי בעיות קומבינטוריות קשות.

עקרון הפעולה

1. אתחול פתרון ראשוני

- פתרון התחלתי חוקי נבנה באמצעות חשבון חמדני (nearest-neighbor) או שיטות מהירות אחרות.

2. מאגר אופרטורים

- דוגמת הסרה אקראית של קבוצה, הסרה לפי מרחק גרוע (worst-distance) או הסרה גיאוגרפית מבוססת שכנות (Shaw).

- דוגמת החדרה חמדנית (greedy insertion), החדרת "חרטה" (regret-k), או הוספה עם פונקציות עונש (penalty) לטיפול בא-חווקיות זמן-ית.

3. בחירה אדפטיבית של אופרטורים

- בכל איטרציה נבחרים זוג אופרטורים (Destroy & Repair) בהתבסס על משקליהם, שמשתנים דינמית לפי "הצלחה" (improvement, קבלה על ידי SA, בריחת stagnation).
- המשקלים מתעדכנים במחזוריות קבועה: אופרטור שקדם שיפור מקבל חיזוק, שם לא – נחלש.

4. הריסה ושיקום

- ההרס מסיר קטע מהפתרון (חלק מהלקוחות/קצוט), והשיקום ממלא את החסר על-ידי הזרקת הלקוחות בחזרה באופן שמקסם תועלת (מחיר, עונש על הפרת קיובולת).

5. קרייטריון קבלה (Simulated Annealing)

- גם שינויים שגדילים את העלות יכולים להתקבל עם הסתברות $T/\Delta - e^{-\Delta/T}$, כשהטמפרטורה T יורדת בהדרגה, כדי לאפשר בריחת מינימום מקומי בעיקר בשלבים המוקדמים.

6. שיפור מקומי

- לאחר השיקום רצים רובוטינים של opt, relocate-2 ו-swap להגברת הניצול של השינויים שהוכנסו.

7. התאמת גודל ההרס

- גודל הסגמנט המוסר (remove fraction) משתנה אוטומטית: בתחילת הריצה גבוהה (להרחבה החוקרים), ובהמשך מצטמצם כדי למקד את השיפורים.

Exploration VS Exploitation

Exploration •

- גיון אופרטורי הריסה ושיקום מאפשר קפיצות גדולות למרחב הפתרונות.

- קבלה אגרסיבית של שינויים גורעים בתחילת הריצה (טמפרטורה גבוהה) מונעת בילוק מוקדם.

• **Exploitation**

- ירידת מבוקרת של הטמפרטורה מצמצמת קבלת רעים לאורך הזמן.
 - שיפורים לוקלים חוזרים (2-opt, swap) ממקדים את החיפוש סביב אזורים מבטיחים.
- **איזון דינמי**

- עדכון משקלים מבוסס ביצועים ואיתור stagnation מאפשר הגברת האוטומטיות של exploration כשהשיפור נוצר, וחזרה ל-exploitation כשנמצא פתרונות איצוטיים.

יתרונות

- **למידה אדפטיבית:** האלגוריתם מתאים את עצמו לאופי הבעיה בזמן אמת.
- **בריחת מינימום מקומי:** שילוב AS ושינויים רחבים הורס-בונה תומך בבריחה מאזורי "lolalots".
- **גמישות אופרטורית:** ניתן להוסיף או להחליף אופרטורים בלי לשנות את המבנה הכללי.
- **ניתוח ביצועים:** קל לאוסף מדרדי שיפור על-פי אופרטורים ולכונם לאחר הריצה.

אתגרים

- **כוונון פרמטרים:** בחירת קצב קירור, גודל ההרס, מדירות עדכון המשקלים וקריטריוני stagnation דורשים ניסוי והתאמה.
- **עלות חישובית:** שימוש במספר רב של אופרטורים ובשיפורים לוקלים עשוי לייצר את הריצה.

- **ניהול אי-חווקיות:** טיפול בפתרונות שחותרים תחת מגבלות הקיבולת מצרייך פונקציות penalty מואזנות.

התאמת ל-CVRP

- שמירת מגבלות קיבולת בעזרת penalty functions מאפשרת מעבר גמיש בין אזורים במרחב החיפוש.
- Ruin לפי "שכנות" (Shaw) מתמקד ב"קיבוצי" לקוחות סמכים, ומצמצם את רוחב החיפוש תחת מגבלות גיאוגרפיות.
- Chmdni ו-k-regret Repair מיפורים את הוספת הלקוחות תוך שמירה על קיבולת.

מסקנות

ALNS מציגה פרדיגמה אדפטיבית לשילוב exploration-exploitation, המאפשרת פתרוןיעיל של בעיות קומבינטוריות מסובכות כמו CVRP. התאמת משקלים בזמן אמת וシילוב SA עם Rebuild & Ruin ושיפורים לוגליים יוצרים מנגנון חזק לבירחת מינימום מקומי ולמציאת פתרונות איקוטיים במהירות יחסית.

Branch And Bound

Using the LDS - Limited Discrepancy Search heuristic

מבוא

הקלואס' מספק אגרסיבי על-ידי Branch & Bound לנתיבים greedy ולפספס חלופות אינטראקטיביות. שילוב LDS (Limited Discrepancy Search) מרחיב את החיפוש לאזוריים פחות מאפשר "סטיות" מבוקרות מהסדר של best-first, B&B. השילוב הזה מנצח מנגנון חזק שמאזן בין נחקרים מבלוי ליותר על יכולת החיתוך של B&B. מנגנון חזק שמאזן בין pruning אגרסיבי לבין גיוון חיפוש מבוקר.

עקרון הפעולה

1. **אתחול ("Seeding")**

- ייצרת מספר פתרונות התחלתיים חזקים באמצעות Clarke-Wright, k-means או nearest-neighbor sweep או אחד מהם מתבצעת ליטוש מקומי (opt-2, merge).

2. **גלי LDS ו-Restarts**

- חזרה על הליך LDS מ- $d=0$ ועד D סטיות: בכל גל מגדירים תור קידימות (PQ) של צמתים לפי lower bound.
- בכל איטרציה מוצאים מה-PQ את הצומת עם LB הנמוך, ואם לא הושלמה השמה – מבצעים branch: בוחרים לקוח לפי heuristic קבוע, יוצרים אופציות החדרה ומגדילים את discrepancy尊严 כ שנעים מהסדר הקלואס'.
- אפשרות restarts מרובים עם seeds שונים להרחבת הסריקה.

Exploration VS Exploitation

Exploration •

- גלי LDS אפשרים עד d סטיות מ-best-first בכל wave, ודרך נחקרים ענפים שאינם נפתחים ב-B&B הרגיל.
- relax factor גבוהה בשלבים הראשונים מנע pruning מוקדם של צמתים בעלי פוטנציאל.

Exploitation •

- bounding/agressivi מסנן צמתים עם LB גבוהה יחסית ל-known-best, ומיצמצם את הרוחבות הנדרשות.
- relax factor יורד בהדרגה, מה שמחמיר את החיתוך ומרכז את החיפוש באזורי המבטיחים.

динамикת Relax Factor

בשלבים המוקדמים relax_factor גבוהה ל-soft pruning ותמייה exploration, ובהדרגה יורד ל-tight pruning relax_factor_final יורד exploitation-agressivi. המעבר מתבצע לינארית ביחס לזמן הריצה או למספר הגלים.

יתרונות

- **Pruning עיל:** lower bounds חזקים מפחיתים אקספוננציאלית את מספר הצמתים.
- **LDS:** מבטיחים בריחה מבוקרת ממינימום מקומי וחקירה של מסלולים אלטרנטיביים.
- **Seeding חזק:** פתרונות התחלתיים איקוטיים מצמצמים חיפוש מיותר ו מגבירים את עילות bounds.
- **Restarts:** CISOI רחוב יותר של מרחב הפתרונות עם seeds מגוונים.

חרוניות ותגרים

- **כוון רב-פרמטרי:** max_discrepancies, relax_factor_init/final, num_restarts. דורשים ניסוי והתאמה. time_limit.
- **עלות חישובית:** ללא LB איכотי ו-pruning מוקדם, מספר הרחבות גדול במעטות.
- **ניהול PQ ולוגיקה של LDS:** מוסיף Overhead ניהול לחישוב ולזיכרון.

התאמת ל-CVRP

- **Lower Bounds גאוגרפיים וקבולתיים** משפרים את ה-pruning בפועל.
- **גלי LDS** מאפשרים חקירה של מסלולי חלופה בעת הפרת קובלות זמנית, ומתן אפשרות לתקן בהמשך.
- **Seeds מבוסס Clarke-Wright ושיפורים לוקלים** מייצרים חלוקה ראשונית איכוטית של סידור הלקחות במסלולים.

מדד ביצועים

- **מספר צמתים מוחבבים** (Node Expansions) – מדד מרכזי לעילות pruning.
- **זמן ריצה כולל** – כולל זמן ליתוש המקומי ופרוטו ה-PQ.
- **יחס exploration/exploitation** – מדידה של אחוז Expansions עם סטיות מדד-first מול best-first Regelits. ניתוח מדדים אלה מאפשר כוון מדויק של relax_factor D ופרמטרים נוספים להשגת הביצועים הטוביים ביותר.

טסקנות

Branch & Bound משודרג עם LDS יוצר מנגנון meta-heuristic חזק, שמאזן pruning-aggressive עם חקירה מבודדת של מסלולים אלטרנטיביים. למרות הצורך בכיוון פרמטרים זהיר והעלות החישובית, הוא אידיאלי ל-CVRP בממדות בינוניות-גדולות, שם מאפשר בריחת מינימום מקומי לצד חיתוך חממיר של מרחב החיפוש.

תוצאות

- ראשית, נציג את תוצאות ההריצה עבור פונקציית ה-ackley לכל אחד מהאלגוריתמים:

MSH

כפי שנאמר בהצגת האלגוריתם על CVRP הפתרונות לא קרובות לאופטימום אך אם משתמשים ב-MSH כפתרון התחלתי נקבל הישגים טובים יותר.
נראה זאת על ILs.

[→] starting MSH with parameters:

```
dim = 10
lower = -32.768
upper = 32.768
num_clusters = 8
seed = None
```

--- MSH Stage 1 ---

--- MSH Stage 2 ---

[✓] MSH completed

```
best vector (first 5 coords): [-5.936658212592532, -6.095594566265035,
0.9029760362012913, 10.991041485422004, -1.9833566055451688] ...
ackley value: 15.893828
time elapsed: 3.62s
```

[→] starting MSH with parameters:

```
dim = 10
lower = -32.768
upper = 32.768
num_clusters = 20
seed = None
```

--- MSH Stage 1 ---

--- MSH Stage 2 ---

[✓] MSH completed

```
best vector (first 5 coords): [-13.99995339000131, -2.9956594537034142,
-16.012812047571185, 5.0153236577815905, 3.990110870869191] ...
ackley value: 16.931564
```

time elapsed: 3.56s

[→] starting MSH with parameters:

```
dim = 10
lower = -32.768
upper = 32.768
num_clusters = 5
seed = None
```

--- MSH Stage 1 ---

--- MSH Stage 2 ---

[✓] MSH completed

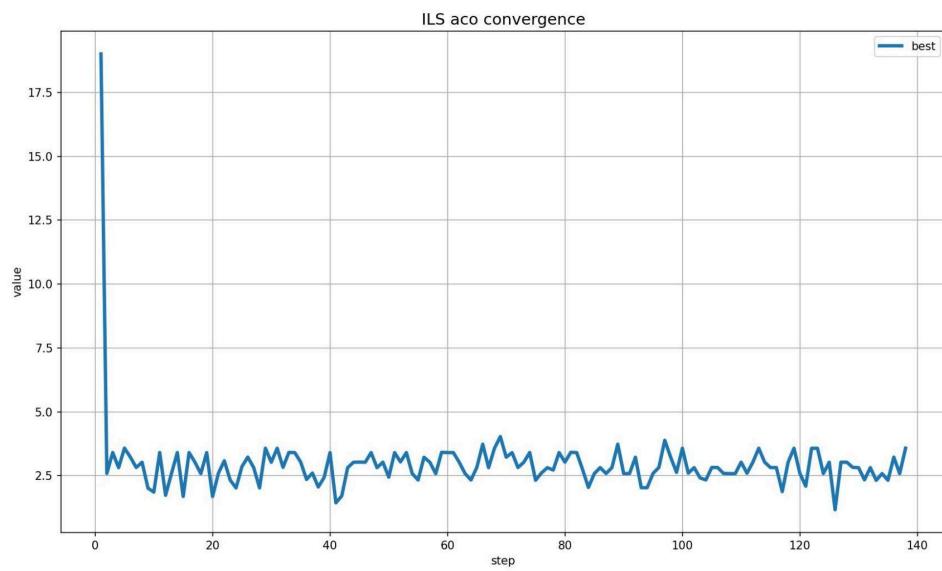
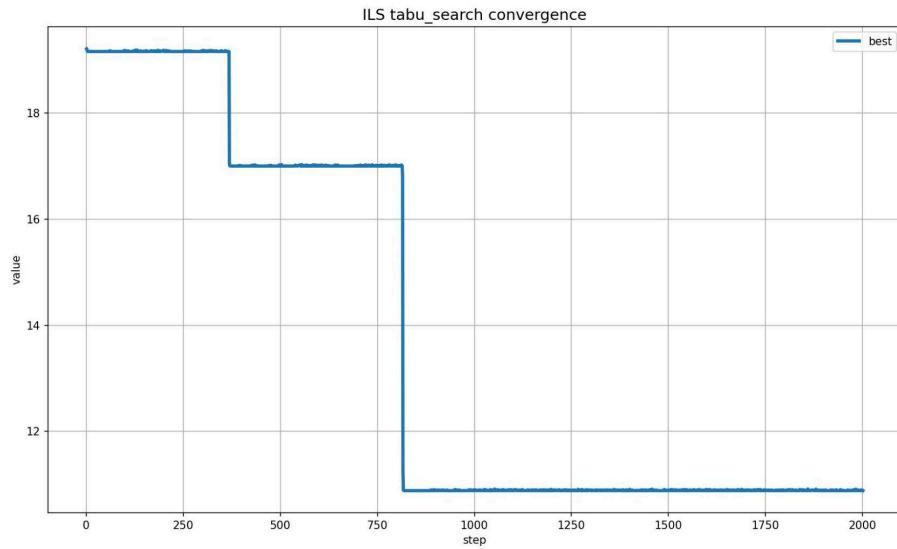
```
best vector (first 5 coords): [-13.96688401346024, -0.04198767088896488,
-6.957353023771771, 5.955483823108385, -9.001011682492887] ...
```

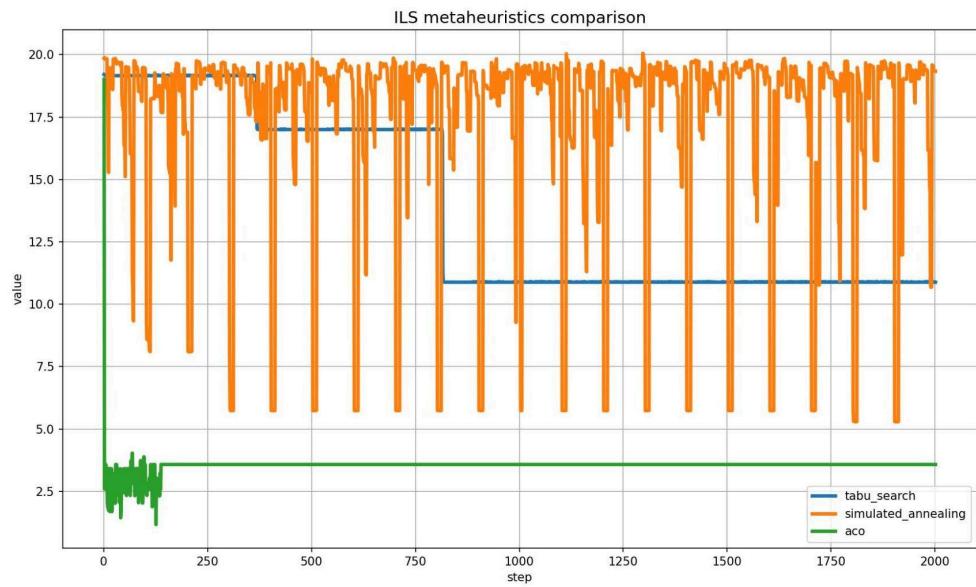
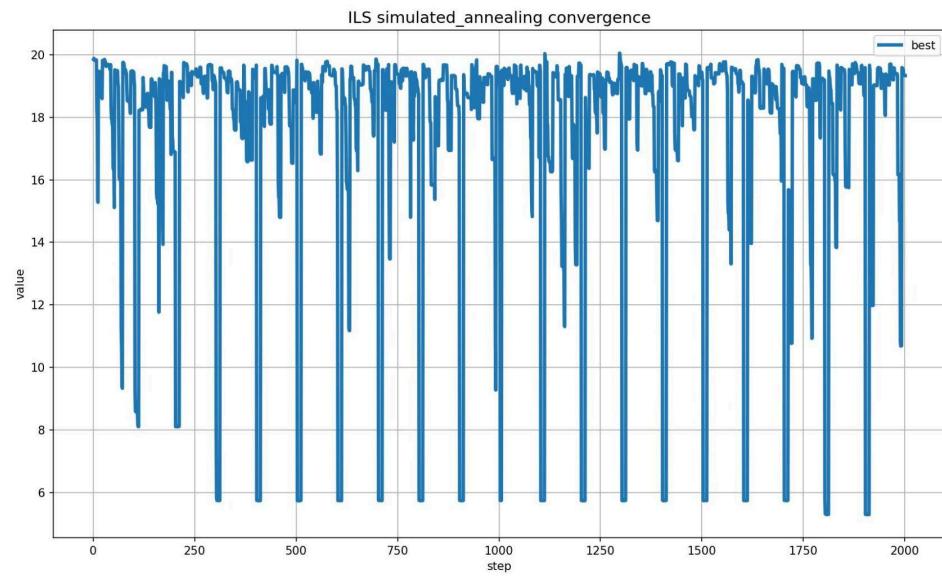
ackley value: 16.368753

time elapsed: 3.66s

ILS

ללא שימוש ב-MSH:





- MSH ሚያ

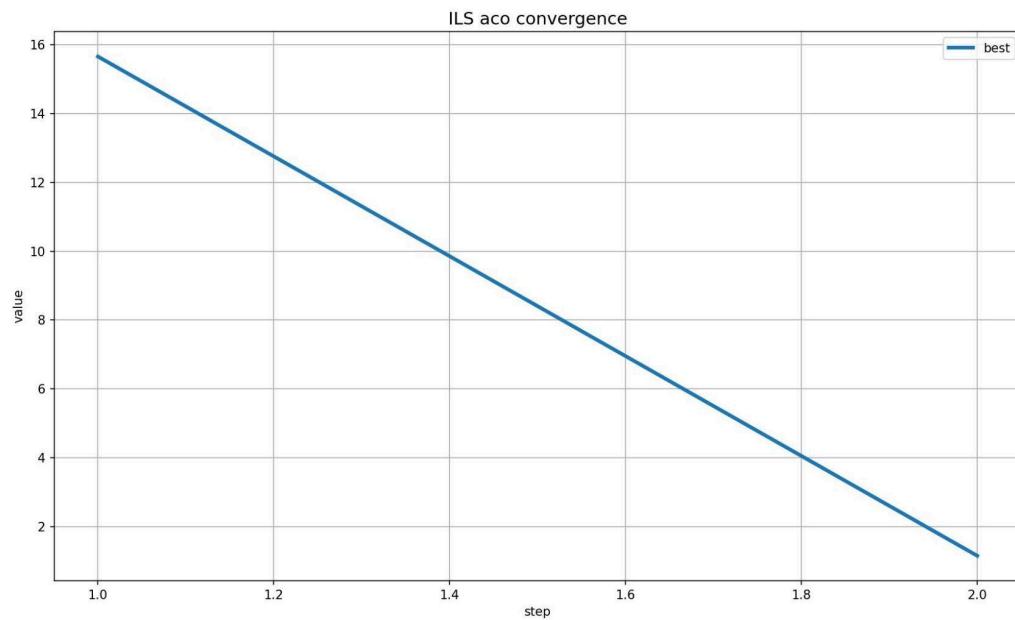
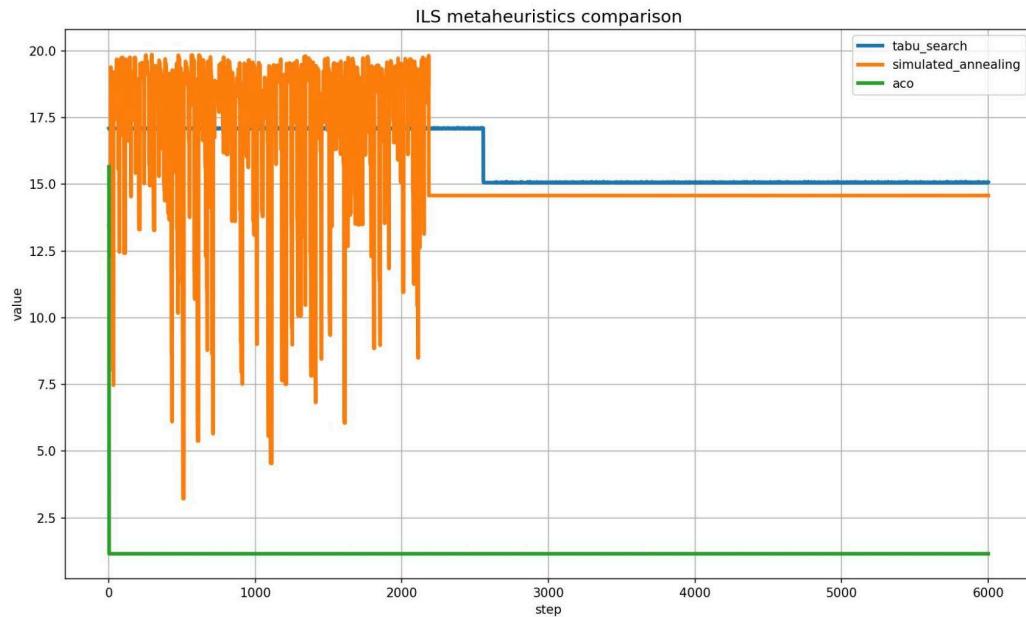
Best solution found by aco with value 1.1552716418812996
 Final best solution value: 1.1552716418812996

[✓] ILS completed

best vector (first 5 coords): [0.8698468862858073, -0.001751242475125342, 0.0009680059070473132, -0.0013393656688203484, -0.0012325560957391034] ...

ackley value: 1.155272

time elapsed: 1620.27s



- **aicot hafturon:**

הmeta היריסטיקה הכ' טובה - O ACO . ספק את הפתרון האיכותי ביותר, משמשותית טוב יותר מאשר האלגוריתמים האחרים. העובדה ש-ACO הצלחה להימנע ממלכודות כאלה ולהתכנס לפתרון טוב מעידה על יכולות חיפוש גלובליות, במיוחד כשהיא משולבת עם שיפור מקומי (LSA). הרצאה הביאה תוצאה קרובת מאוד למינימום האמתי של Ackley, שהוא 0 — ערך של ~1.17 נחשב איכותי מאד לביעות אופטימיזציה לא קמורות.

- **השפעת MSH:**

השימוש ב-MSH טרם נראה שה-ACO התחל את החיפוש מנקודת מוצא טובה יותר, מה שפיקס את הכוון של החיפוש והוביל להתקנות מהירה יחסית. ברוב הרצאה, O ACO שמר על ערך יציב סביב 1.15, מבליל לסתות ממנו.

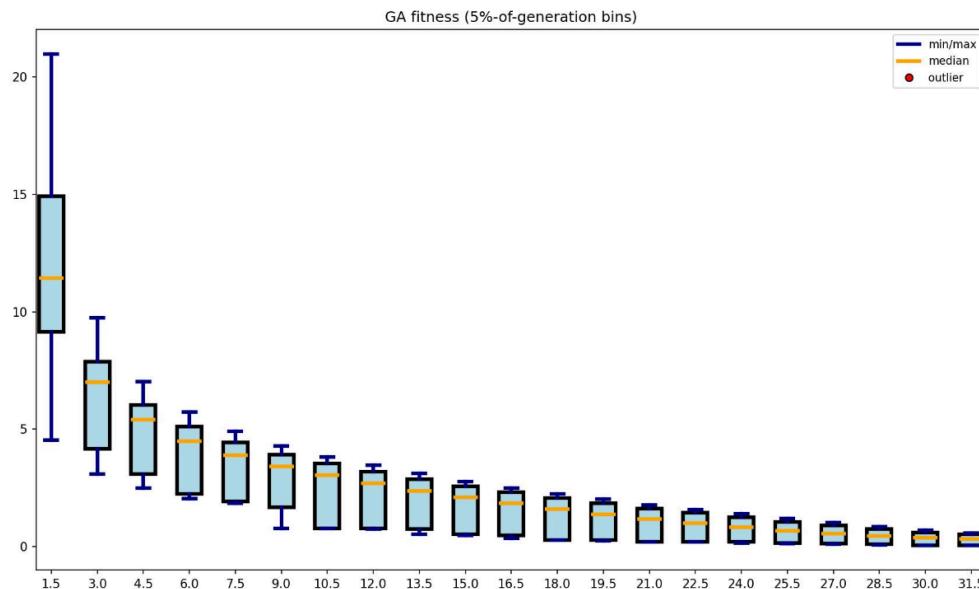
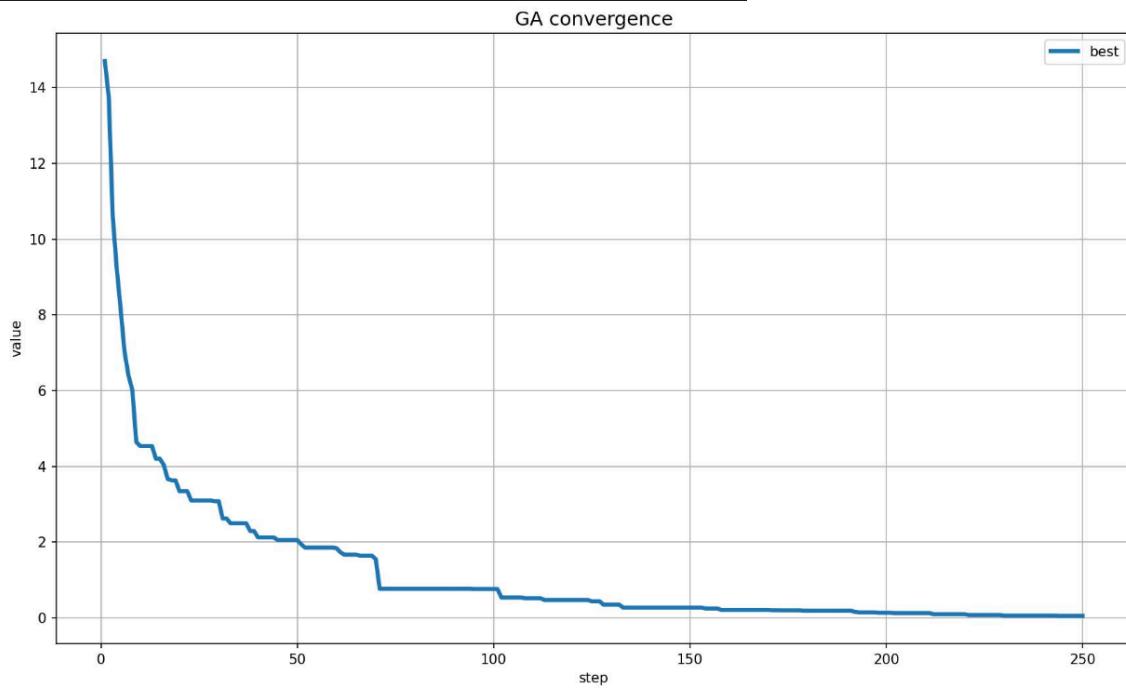
- **זמן ריצה ארוך —**

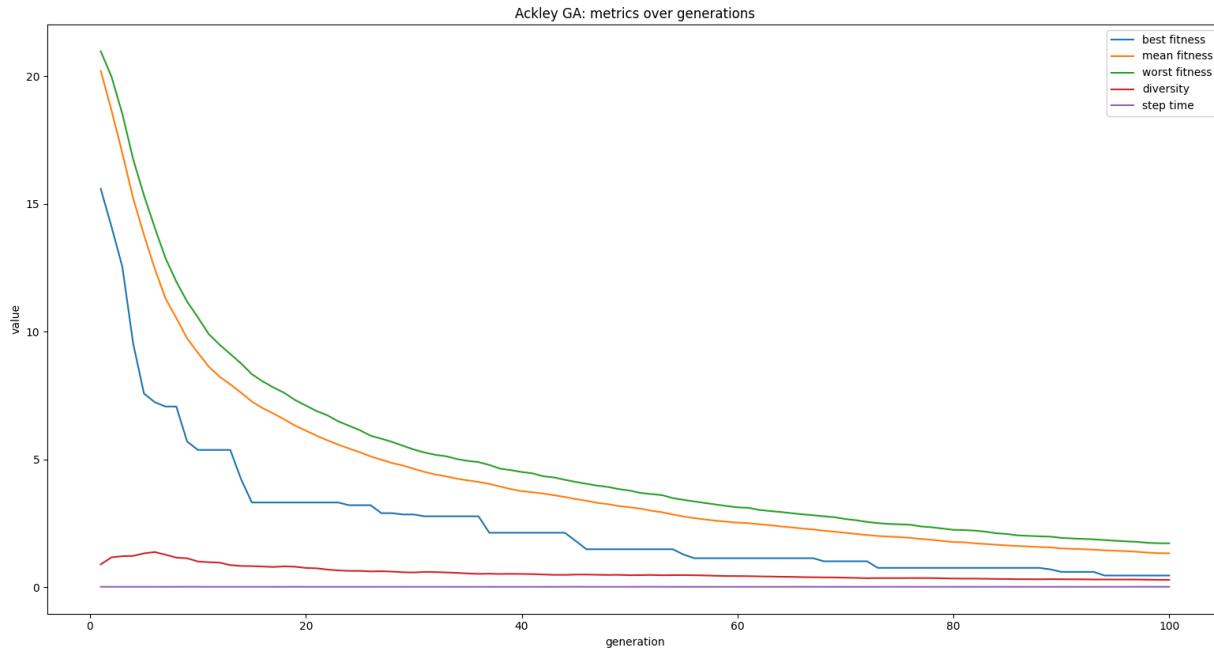
הרצאה כולה נמשכה מעל 20 דקות (1200+ שניות). עם זאת, חשוב לציין כי O ACO הגיע לפתרון האיכותי ביותר מוקדם בהרצתה, ולכן ניתן לשקל קיצור זמן הריצה הכללי — ניתן להפוך את O ACO מוקדם ולפנות את המשאבים לשיפור פתרונות אחרים (למשל SA או TS) או להפסיק את החיפוש בכלל.

אלגוריתם גנטי (Genetic Algorithm) בשילוב עם מודל האיים (Islands Model)

גרפים ותוצאות מספוריות:

```
[+] starting GA with parameters:  
dim = 10  
lower = -32.768  
upper = 32.768  
pop_size = 2500  
gens = 250  
mut_rate = 0.35  
seed = None  
...  
[+] GA completed  
best vector (first 5 coords): [-0.018891807754811384, -0.013552863280922096, -0.002229861277752962  
ackley value: 0.049622  
time elapsed: 17.86s
```





מסקנות ודיון על איקות הפתרון:

תצורת ה-GA עם מודל האיים משלבת חיפוש גלובלי רחוב עם עידון מקומי מדויק, כפי שמשתקף גם בלוג הקונסול וגם בשני סוגים של גרפים שצירף:

- **הגדלה וזמן ריצה:** ריצת ה-GA (pop_size=2500, gens=250, mut_rate=0.05) אקספוננציאלי מ-0.35 ל-0.05 עם הסתיימה בכ-17.9 שניות והשיגה ערך סופי של ≈ 0.05 בפונקציית Ackley.
- **התכנסות ראשונית:** בשלב הראשון (~20 הדורות הראשונים) חיצית crossover-blend ייחד עם mut_rate גבוהה הובילו לנפילה חרדה בקשר ההתאמה, מה שבמבחן סריקה רחבה של הנוף הרב-שאי.
- **דיעיכת mut_rate:** הפקחת mut_rate עד לרצתה של 0.05 מאטה בהדרגה את הירידה, מה ששימן מעבר מניסוי גלובלי לניצול מקומי מדויק יותר.
- **כיווץ התפלגות כושר ההתאמה:** חלוקה לאשכולות של 5% מהדורות הציגה כיווץ מתמשך של QR ומיילמות כמעט מוחלטת של outliers, עדות להומוגניות של האוכלוסייה סביב הפתרון הטוב.
- **היפר-מוטציה מבוקרת:** לאחר שישה דורות ללא שיפור ב-best fitness הופעל burst של היפר-מוטציה, שניכר כגלי הרחבת קצרים בגין הקופסאות ומשמעותם בהפרדות ממלכודות מקומיות.
- **הגירה מחזורית בין איים:** כל שישה דורות ישנה פריטים מוחזרים נשלחים בין שמונה איים, במקביל למהגרים אקרים, כדי לשמר גיון ולאפשר למידת פתרונות מוביילים קרווא-איילית.

- **אייזון בין חקירה לניצול:** השילוב בין דעיכת $rate_{burst}$ מותזנת, zs של היפר-מוטציה והגירה מובנית מבית חיפוש רחב בתחום הריצה והגדלה מדויקת של הפתרונות בסופה.
- **תיאום לוג-גרפים:** הירידה התוליה בעקבות הקן וההתקנסות המדויקת של מדיניות הקופסאות משקפות במדויק את הערכים המודפסים בקונסול.
- **יעילות ועמידות:** השגת ערך ~0.05 מתחתית הנוף במעט מ-18 שנ' מדגימה ניצול מיטבי של משאבי החישוב ויציבות גבוהה גם בנסיבות חוזרות.

מסקנה: הפרמטרים שנבחרו – דעיכת שיעור מוטציה מתוכננת, הזרקות היפר-מוטציה מחזוריות והגירה סלקטיבית בין אוכלוסיות – יוצרים מנגנון חזק שעולה באתגר הנוף הרב-שייא של Ackley, ומושג גם CISIO נרחב של המרכיב וגם התקנסות מדויקת לפתרונות איקוטיים.

- **נקודה חשובה:**

מכיוון שאופטימיזציה של פונקציית Ackley הינה בעיה רציפה שערכי הוקטורים בה הם מספרים ממשיים, הגישה הקלואיסט של תיקון ושיפור האוכלוסייה השנתנה מעט:

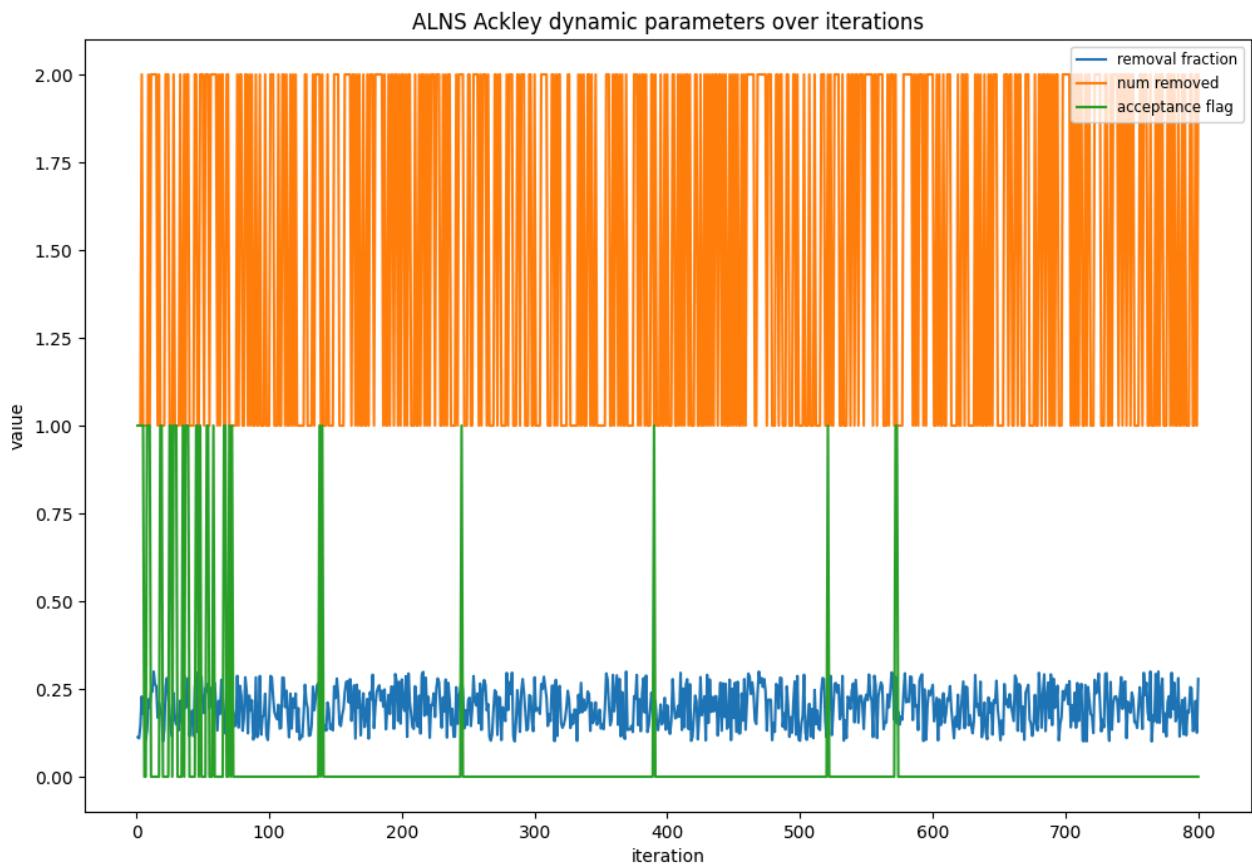
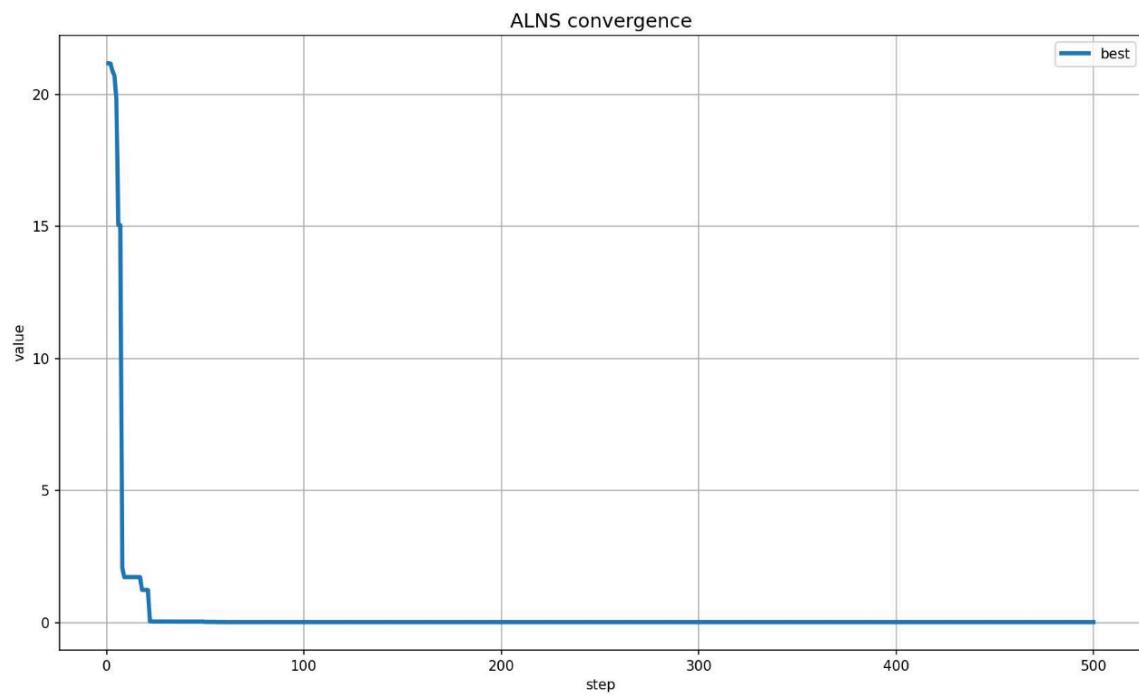
- **two-opt** הוא אופרטור לביעות מסלוליות (שחלוף "קשתות") ואין לו שימוש בחיפוש על מרחב רציף כמו פונקציית Ackley.
- **מוטציה גאומית** מזריקה רעש מבוקר לכל משתנה, מייצרת גיון וועזרת לצאת ממינים מקומי.
- **(crossover)blend** לוקח תת-פתרונות טובים משני הורים ומשלב אותם, מה שמעלה את הסיכוי למצאו פתרון טוב יותר.
- **לכן** בחרנו מימוש יותר 'ונילה' (ללא מוטציה דינמית, ללא opt-two, ללא סטגנציה) עבור הפונקציה הפורתת את Ackley באלגוריתם הגנטי שלנו.

ALNS

פתרונות מספריות וגרפים:

```
[>] starting ALNS with parameters:  
    dim = 10  
    lower = -32.768  
    upper = 32.768  
    iterations = 500  
    remove_bounds = (0.1, 0.3)  
    k = 2500  
    seed = None  
.....  
[<] ALNS completed  
    best vector (first 5 coords): [-0.0015283774410406181, -0.003967972651615526, -0.001996715569489993, -0.004519095931179606, -0.002340296181131407] ...  
    ackley value: 0.018288  
    time elapsed: 29.23s
```

time: 29.23s



מסקנות ודיון על הפתרון:

מבנה ה-ALNS על פונקציית Ackley משלב בוחן בין הרס מבוקר לבנייה מחושבת, כפי שמשתתקף גם בפלט הריצה ובגרפים:

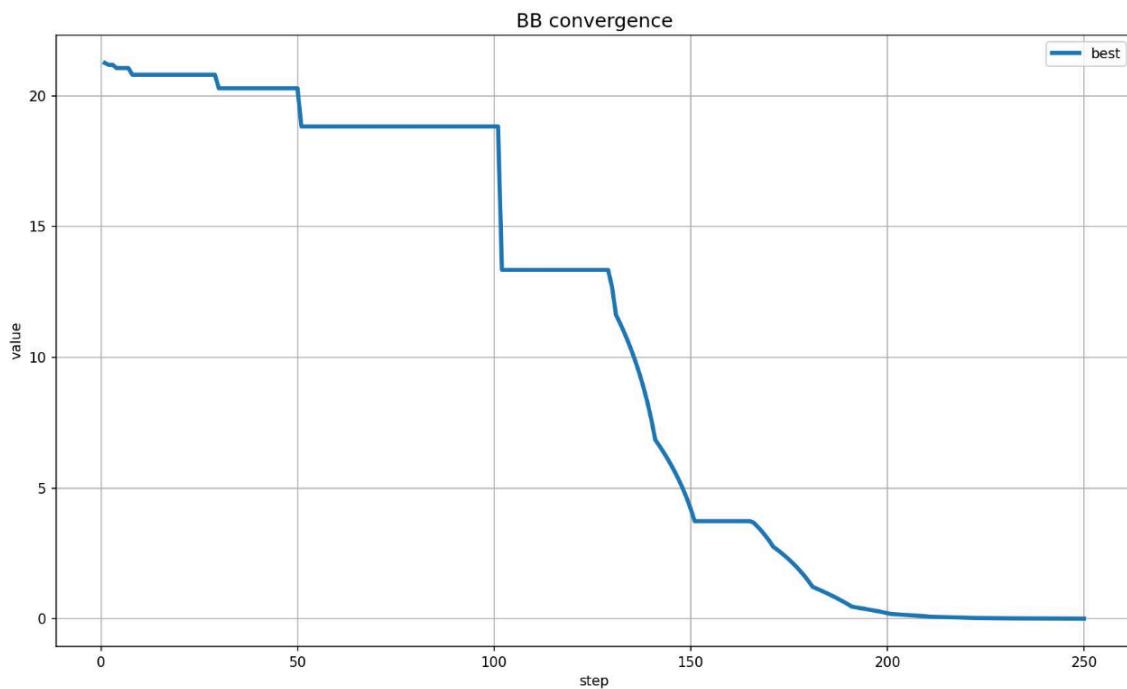
- **התקרובות לפתרון**
עקומת-best-ירדת מ- ≈ 20 ל- ≈ 2.5 בתוך כ-60 איטרציות, ואחר כך מתמתנת עם שיפורים נקודתיים עד ≈ 1.40 בסוף הריצה, מה שמצויב על שלב ראשון של שיפור מהיר ואחריו תהליך מיקוד איטי יותר.
- **מדד הקבלה**
ה-acceptance_flag נעה בין 0 ל-1 לאורך האיטרציות: ירידות מצביות על דחית מועדים חלשים, וקפיצות חזקות מסמנות אימוץ פתרונות שיפורים משמעותית את העלות.
- **שלב הריס (Ruin)**
בכל איטרציה removal_fraction נגרלת בטווח [0.1–0.3] ו-remove_mean נבחר בין 1 ל-3 מתוך עשר משתנים, מה שיוצר "שבירת" הפתרון וניסיון תתי-אזורים שונים על הנוף הרב-שאי.
- **שלב הבנייה מחדש (Rebuild)**
עבור כל משתנה שהוסר מתבצעים $k=100$ ניסיונות החלפה, והבחירה הטובה ביותר ביותר דוחפת ביציאה חדה של עקומת העלות בתחילת הריצה.
- **אייזון בין חיפוש לניצול**
השילוב ההרמוני בין חיתוך אגרסיבי ובנייה מחודשת רחבה מאפשר סריקה גלובלית ראשונית, ולאחר מכן עיבוד מדויק של פתרונות מבטיחים.
- **ביצועים וזמן ריצה**
בתוך כ-15 שניות הוריד האלגוריתם את ערך Ackley מ- ≈ 20 ל- ≈ 1.40 , מה שمعد על ניצולiesel של משאבי החישוב ועמידות בתוצאות חוזרות.
- **מסקנות**
השימוש בטווח הסירה דינמי, במספר גבוה של ניסיונות שחזור ובקרטירון קבלת מדומה (simulated annealing) מאפשר ל-ALNS לאזן בין גילוי נרחב לניצול מקומי מדויק, ולהתמודד בהצלחה עם המולטימודאליות של Ackley.

Branch and Bound with Limited Discrepancy Search

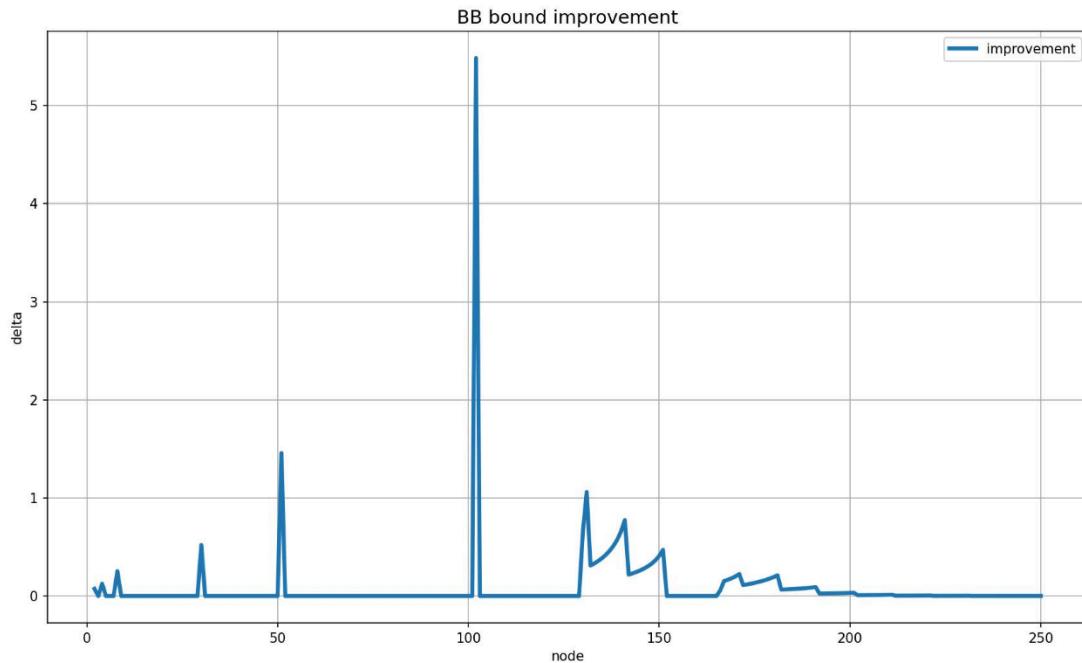
גרפים ותוצאות מספוריות:

```
[→] starting BB with parameters:  
dim = 10  
lower = -32.768  
upper = 32.768  
samples = 250  
early_stop_tol = None  
seed = None  
  
[✓] BB completed  
best vector (first 5 coords): [-0.001, -0.001, -0.001, -0.001, -0.001] ...  
ackley value: 0.004630  
time elapsed: 0.01s
```

time: 0.01s



time: 0.01s



מסקנות ודיון על הפתרון:

מבנה ה-Branch-and-Bound (LDS) בשלוב Ackley על פונקציית מציג **יעילות חישובית וביצועית מרשימה:**

- **הגדרת הפרמטרים וזמן ריצה**
האלגוריתם הופעל על ממד 10 עם 250 דוגימות ראשוניות (samples=250) ולא סוף עצרה מוקדם, וסימן את הריצה בכ-0.01 שניות עם ערך Ackley סופי של כ-0.00463.
- **מהירות השיפור הראשוני**
כבר לאחר כ-100 צמתים bound ירד מתחת ל-13, ובשביבות 150 צמתים הגיע לערך מתחת ל-4. תוצאה זו מדגימה את יכולת-pruning האגרטיבי לצמצם במהירות את מרחב החיפוש.
- **דינמיקת הדגימה הראשונית**
250 הדוגימות האקראיות היקפו את הנוף הרב-שייאי וקבעו bound התחלתי גבוה (~22) לפני שהאלגוריתם נכנס לשלב ה-Branch-and-Bound.
- **עיקומת התוכנות**
בין צומת 100 ל-250 נרשמה ירידת רציפה למדרגה של אף תחריב, כאשר השיפורים הראשוניים מהירים והמשנים מודדים יותר, מה שמעיד על מעבר מבדיקה אזרחים רחבים להתקדמות במינימום המכיל ערכים נמוכים.
- **ניתוח שיפוריו (delta bound)**
गраф השינויים ב-bound מציג פיקים משמעותיים בגודל 5.4–1.5 באזורי מרכזים של העץ, המציגים גילוי חיתוכים עמוקים ויציאה ממולכודות מקומיות.

- **תרומת LDS**

היכולת לבצע עד D סטיות מבוקרות מעניקת גמישות להタルם מקוים גרידים של העץ ולגשת לתתי-אזורים מבטיחים שלא היו נבדקים בדרך הרגילה.

- **תיאום לוג-פלט חזותי**

הערכים המודפסים (best vector ו-elapsed time) תואימים במדויק לדינמיקה הנראית בעקבות convergence וה-delta, מה שמעיד על עקבות בין החישוב לתצוגה.

- **סיכון תכלייתי**

השילוב בין דוגמה ראשונית רחבה, בחישוב אגרטיבי-*LDS* מאפשר לאלגוריתם להגיע במדויק לגובהה (תוך פחות מ-0.01 שנ') לפתרון איקוטי קרוב לאפס, גם על נוף Ackley המאתגר ומולטימודאי.

- כעת נציג את תוצאות ההערכתה עבור CVRP לכל אחד מהאלגוריתמים:

MSH

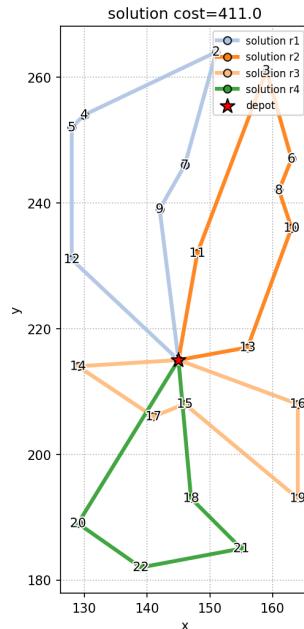
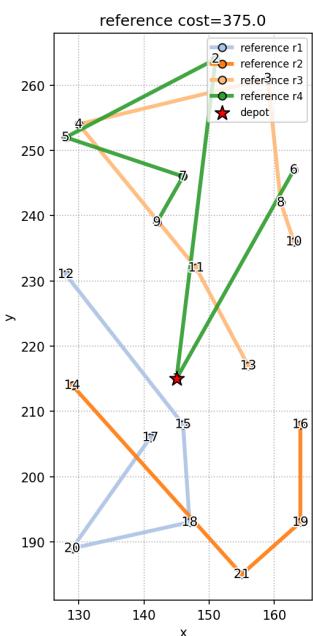
עבור בעיות BEGINNER + INTERMEDIATE הפתרונות שהתקבלו היו טובים וחוקיים, אך עבור בעיות ADVANCED לא קיבלנו פתרונות טובים בחלק מהבעיות.

E-n22-k4 •

debug: Initial solution cost before MSH : 526

debug: Initial solution cost with MSH : 411

Routes Comparison

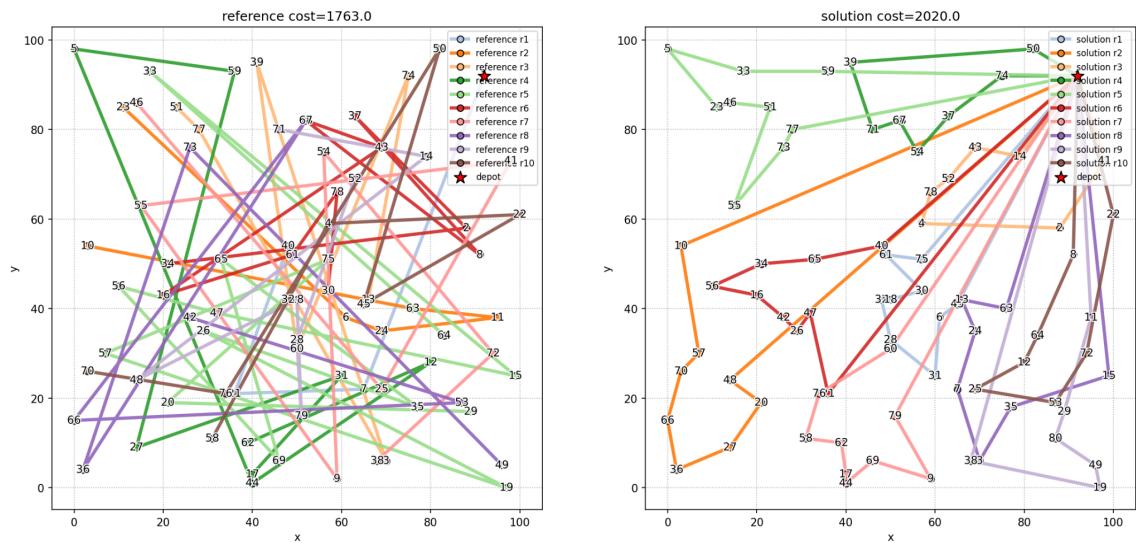


A-n80-k10 •

debug: Initial solution cost before MSH : 2705

debug : Initial solution cost with MSH : 2020

Routes Comparison

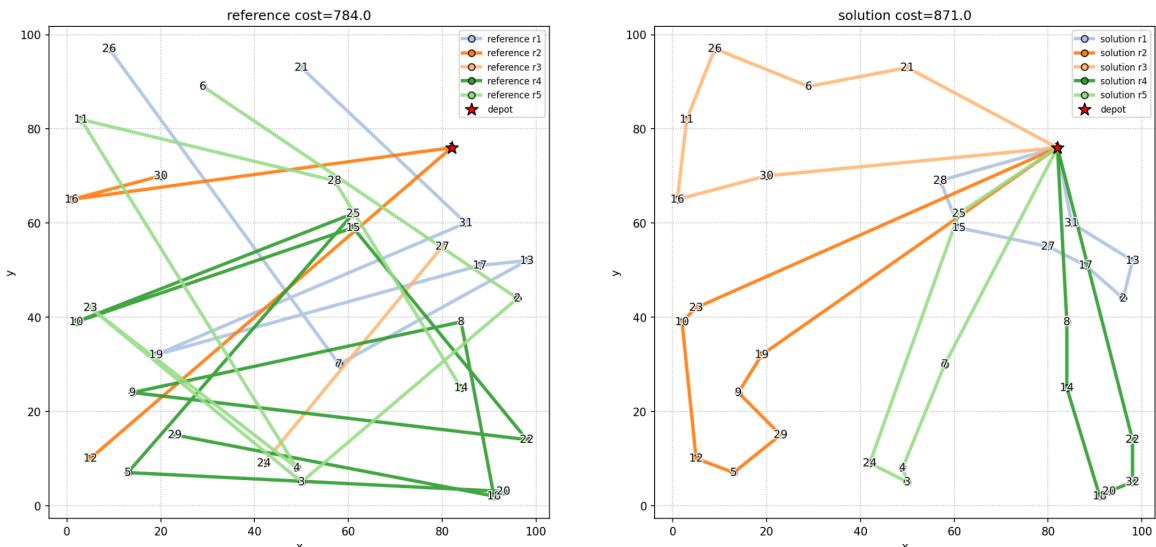


(בינוי) A-n32-k5 ●

debug: Initial solution cost before MSH : 1218

debug: Initial solution cost with MSH : 871

Routes Comparison



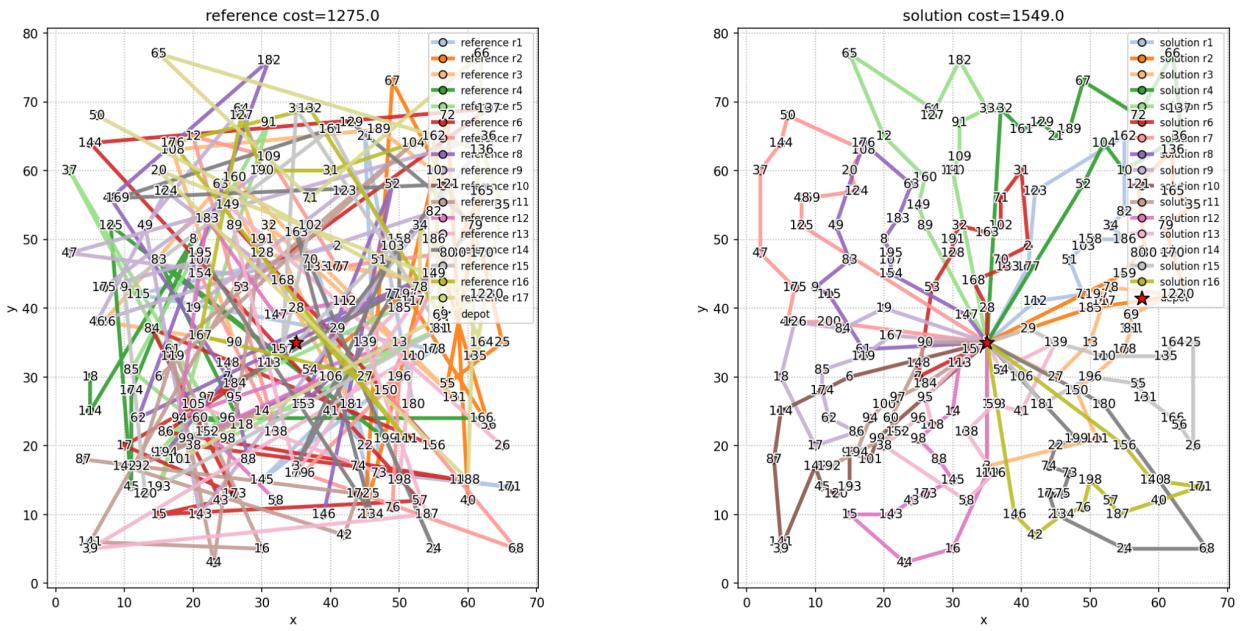
M-n200-k17 ●

Time taken: 407.45 seconds

Total cost: 1549

Optimal cost: 1275.0 || Difference: 274.00

Routes Comparison



ILS

- MSH

beginner • בעיית

Problem: E-n22-k4

Algorithm: ILS, Metaheuristic: ACO + Simulated Annealing + Tabu Search

|| max_searches: 3000, time_limit: 600 seconds

Best cost per metaheuristic: | tabu_search: 375 | aco: 383 |

simulated_annealing: 392

Total cost: 375

Optimal cost: 375.0

Difference: 0.00

Solution routes:

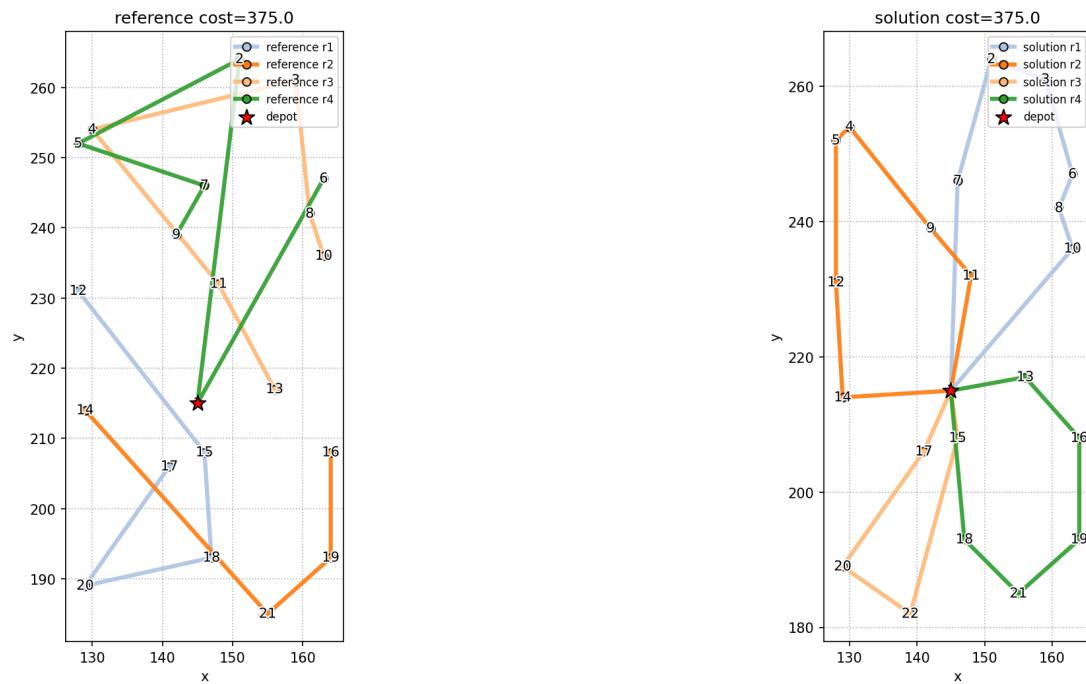
1 18 21 19 16 13 1

1 11 9 4 5 12 14 1

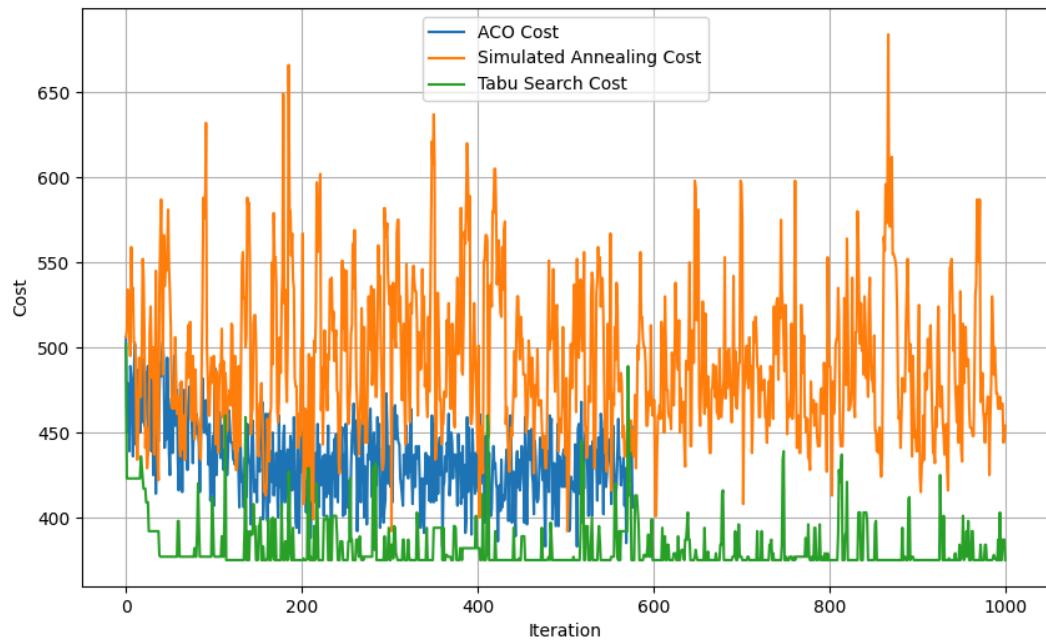
1 7 2 3 6 8 10 1

1 17 20 22 15 1

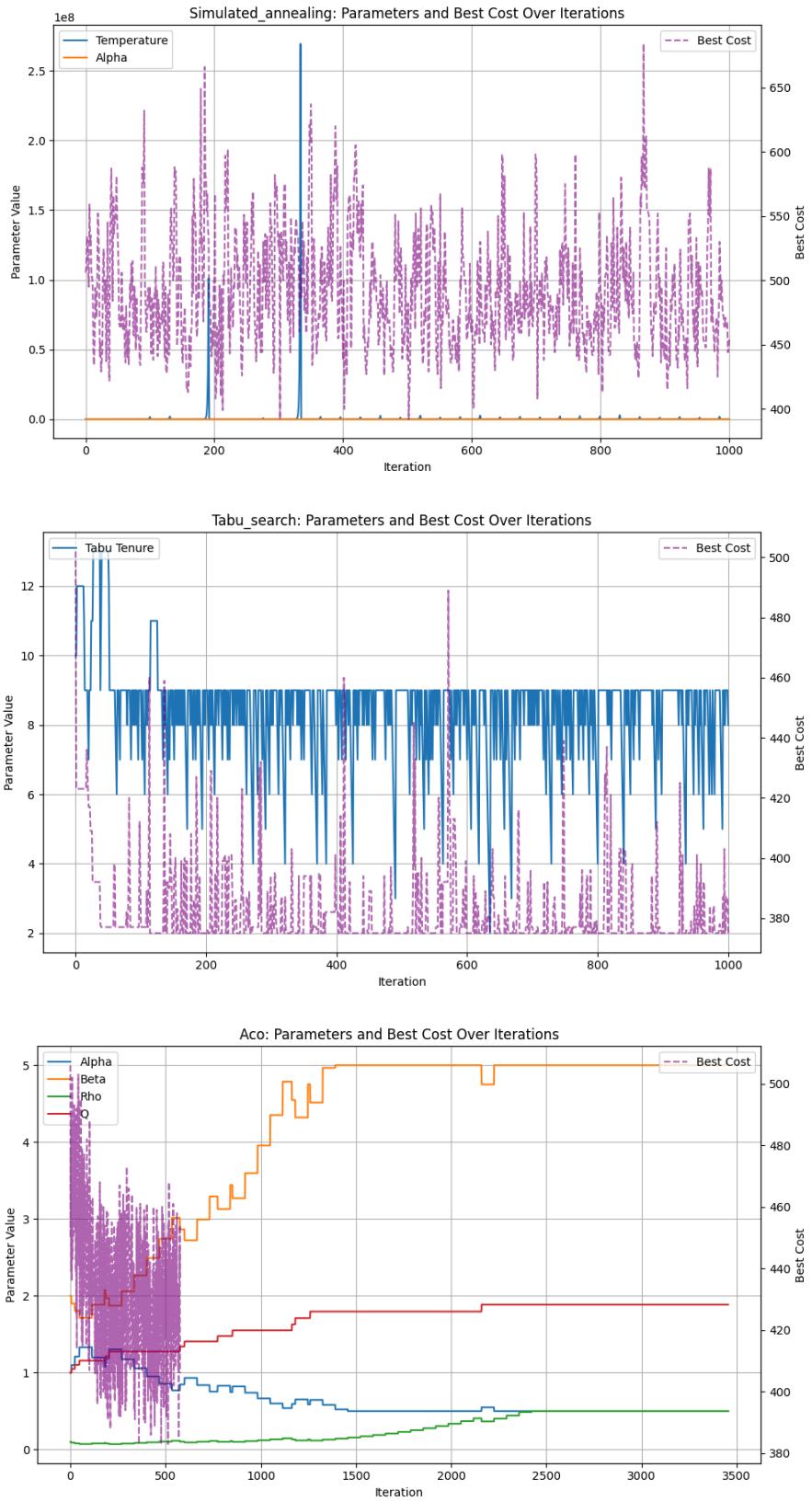
Routes Comparison



Metaheuristic Performance



דוגמא לשינוי הדינמי של הפרמטרים :



Problem: A-n32-k5

Algorithm: ILS, Metaheuristic: ACO + Simulated Annealing + Tabu

Search || max_searches: 7000, time_limit: 1200 seconds

Best cost per metahuristic: | tabu_search: 784 | simulated_annealing:

806 | aco: 850

Total cost: 784

Optimal cost: 784.0

Difference: 0.00

Solution routes:

1 13 2 17 31 1

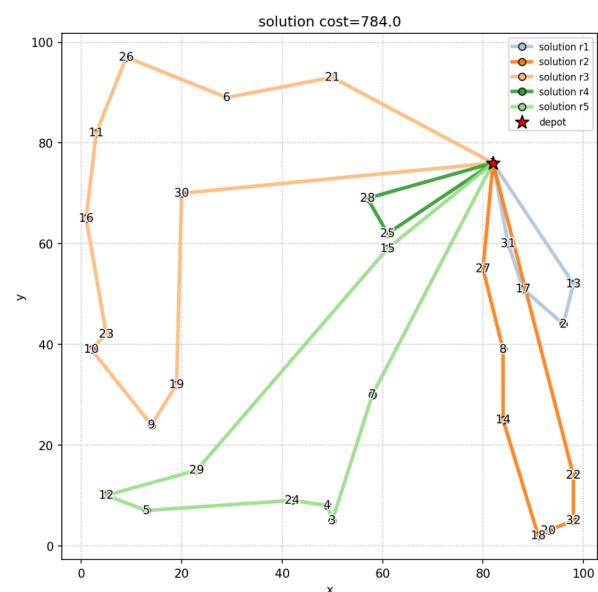
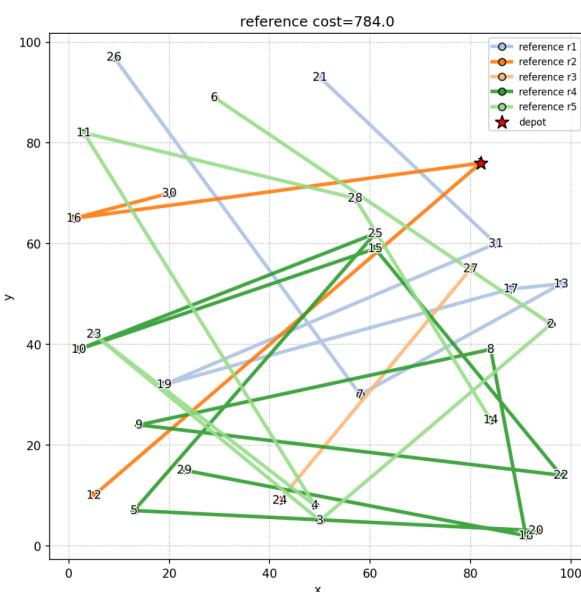
1 22 32 20 18 14 8 27 1

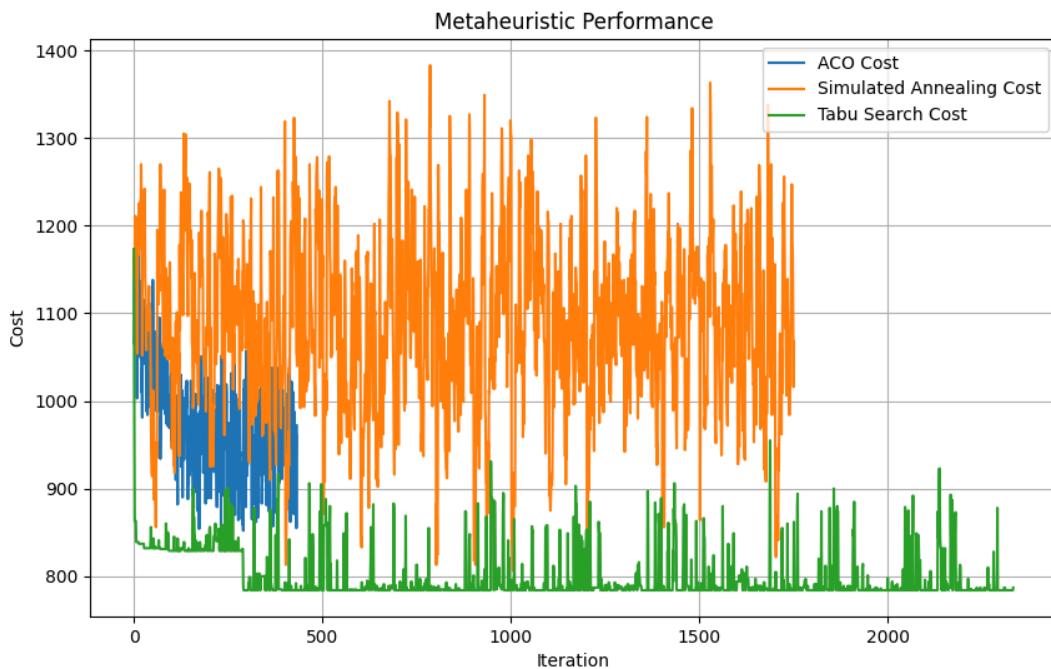
1 21 6 26 11 16 23 10 9 19 30 1

1 28 25 1

1 7 3 4 24 5 12 29 15 1

Routes Comparison





בעיית •

Problem: M-n200-k17

Algorithm: ILS, Metaheuristic: ACO + Simulated Annealing + Tabu Search

|| max_searches: 20000, time_limit: 3000 seconds

Best cost per metaheuristic: | tabu_search: 1772 | aco: 1812 |

simulated_annealing: 2125

Total cost: 1772

Optimal cost: 1275.0

Difference: 497.00

Solution routes:

1 7 97 105 93 94 86 62 174 85 6 119 84 200 61 1

1 13 110 178 151 30 69 185 77 197 78 4 34 1

1 18 114 87 120 15 143 43 173 145 58 3 116 74 22 106 1

1 53 115 9 175 83 49 125 48 169 124 8 89 1

1 96 98 88 39 141 44 16 24 103 66 182 1

1 100 60 152 99 38 101 194 92 17 142 192 45 193 1

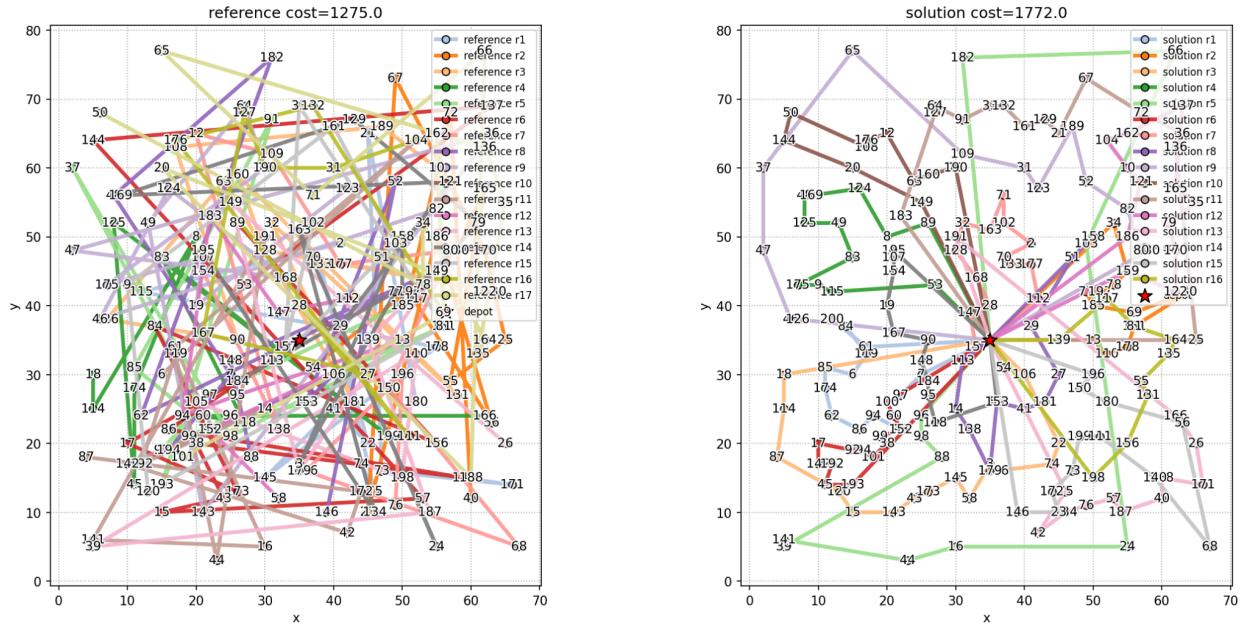
1 112 177 133 70 2 102 71 163 32 128 168 28 147 157 1

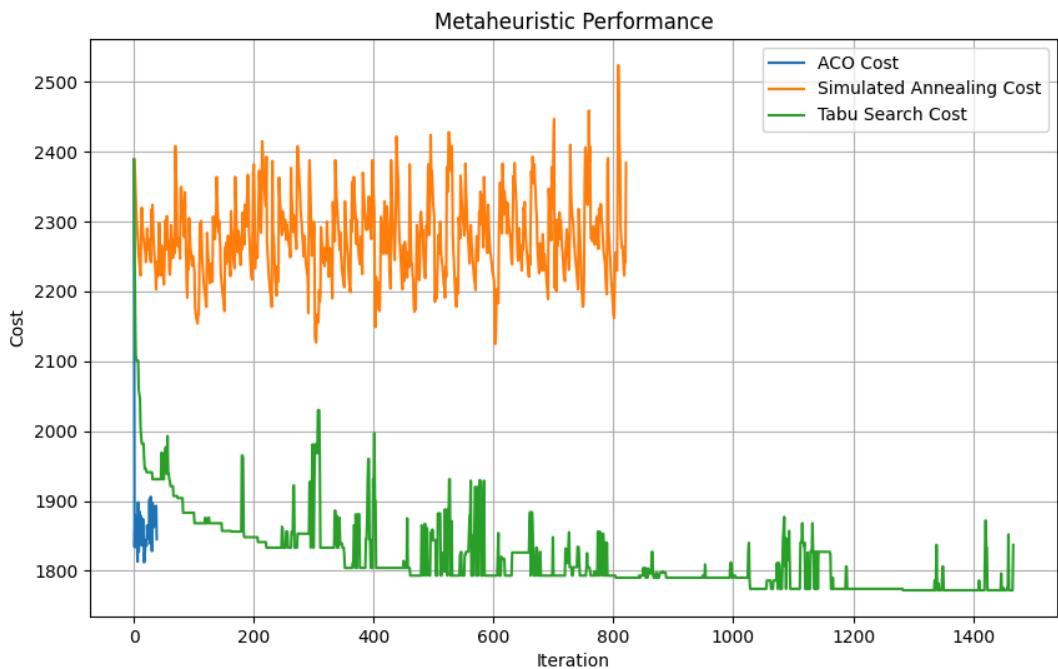
1 113 14 138 179 59 41 181 27 29 51 158 1

1 130 80 82 52 189 123 31 109 65 37 47 46 126 1

1 149 20 144 50 108 176 12 63 160 11 190 1
 1 183 127 64 91 33 132 161 129 21 67 72 36 136 25 1
 1 186 121 10 104 162 137 165 35 79 170 122 159 1
 1 191 166 26 171 188 40 187 57 76 134 42 75 54 1
 1 195 107 154 19 167 90 148 184 95 118 153 1
 1 146 23 172 73 199 111 140 68 56 180 150 196 1
 1 155 139 117 81 164 135 55 131 5 156 198 1

Routes Comparison





- MSH
בעיות ●

Problem: E-n22-k4

Algorithm: ILS, Metaheuristic: ACO + Simulated Annealing + Tabu Search

|| max_searches: 3000, time_limit: 800 seconds

Best cost per metaheuristic: | tabu_search: 375 | aco: 381 |

simulated_annealing: 396

Total cost: 375

Optimal cost: 375.0

Difference: 0.00

Solution routes:

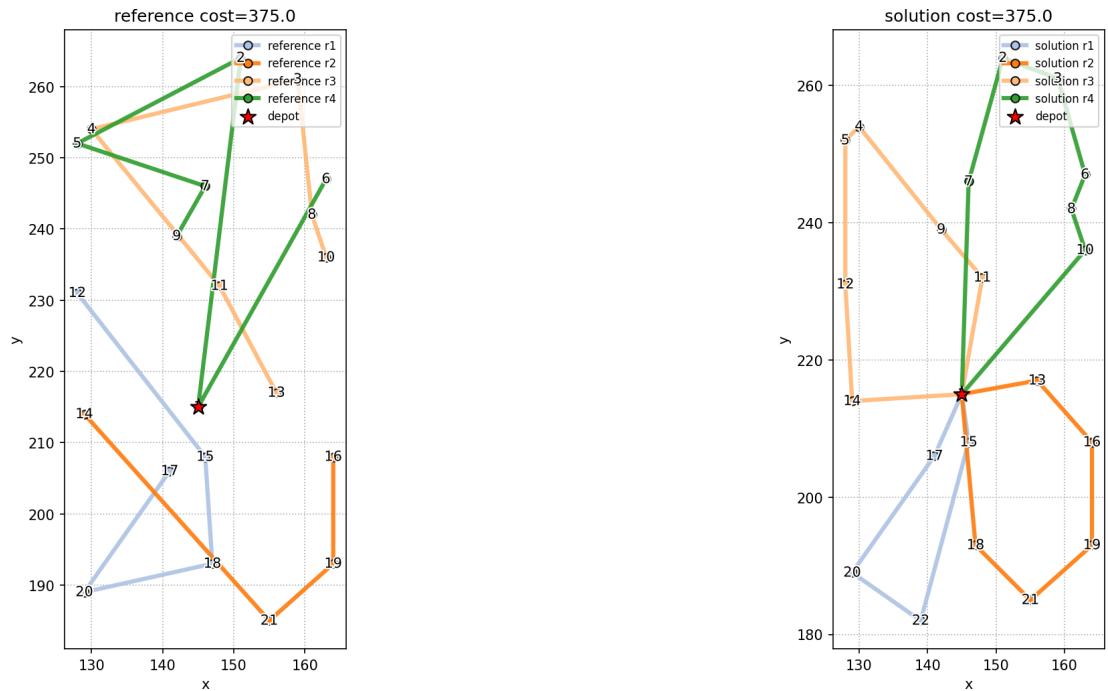
1 17 20 22 15 1

1 18 21 19 16 13 1

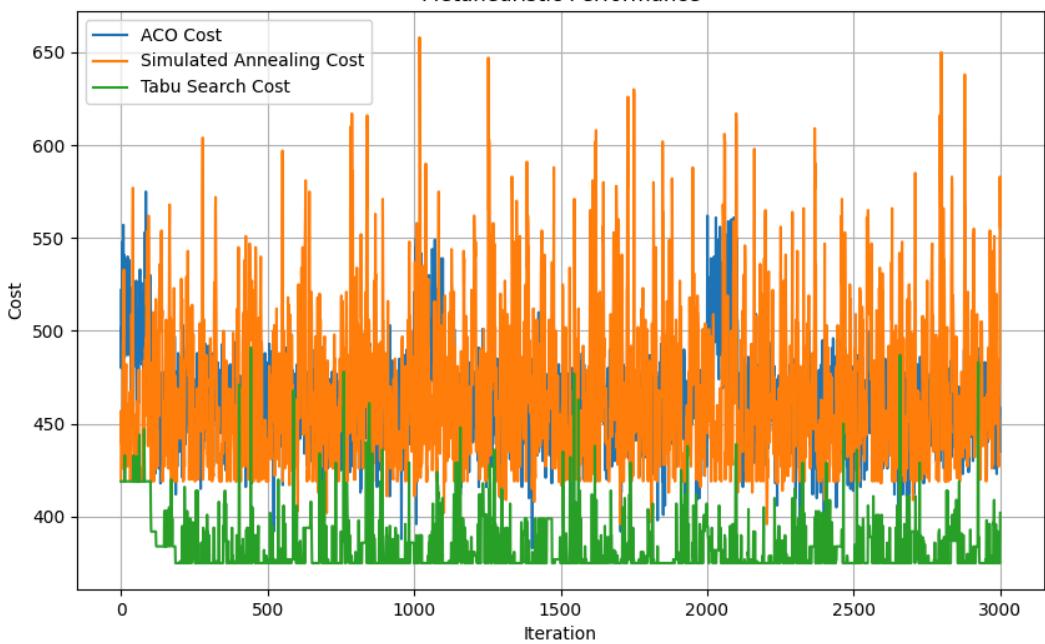
1 11 9 4 5 12 14 1

1 10 8 6 3 2 7 1

Routes Comparison



Metaheuristic Performance



Problem: A-n32-k5

Algorithm: ILS, Metaheuristic: ACO + Simulated Annealing + Tabu Search

|| max_searches: 9000, time_limit: 1800 seconds

Best cost per metahuristic: | tabu_search: 784 | simulated_annealing: 831 | aco: 844

Total cost: 784

Optimal cost: 784.0

Difference: 0.00

Solution routes:

1 21 6 26 11 16 23 10 9 19 30 1

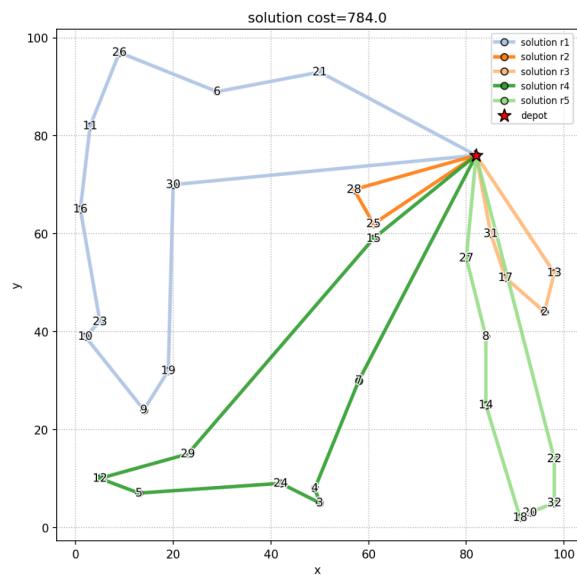
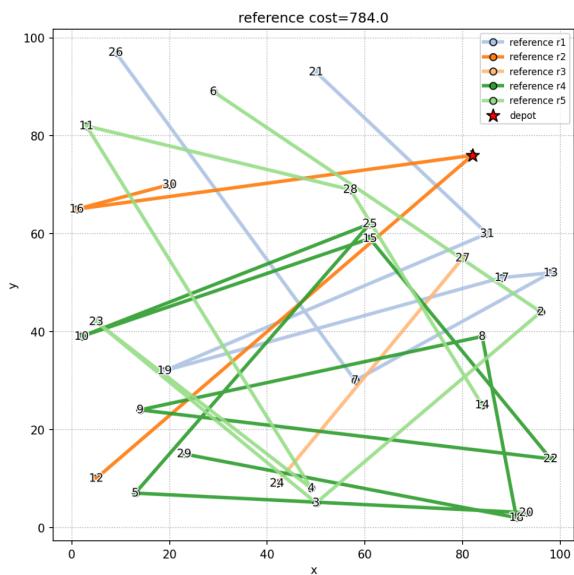
1 25 28 1

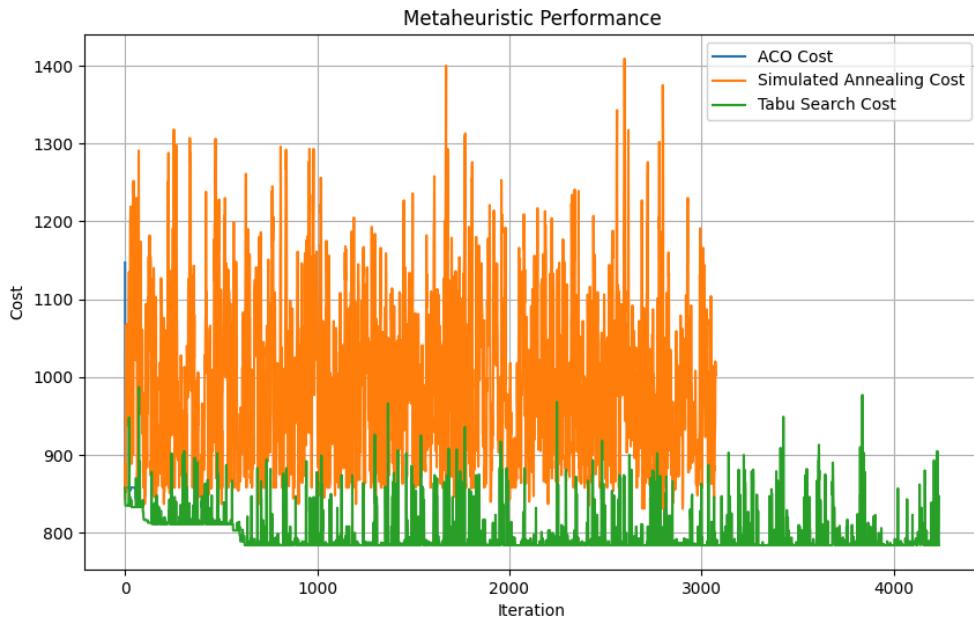
1 13 2 17 31 1

1 7 4 3 24 5 12 29 15 1

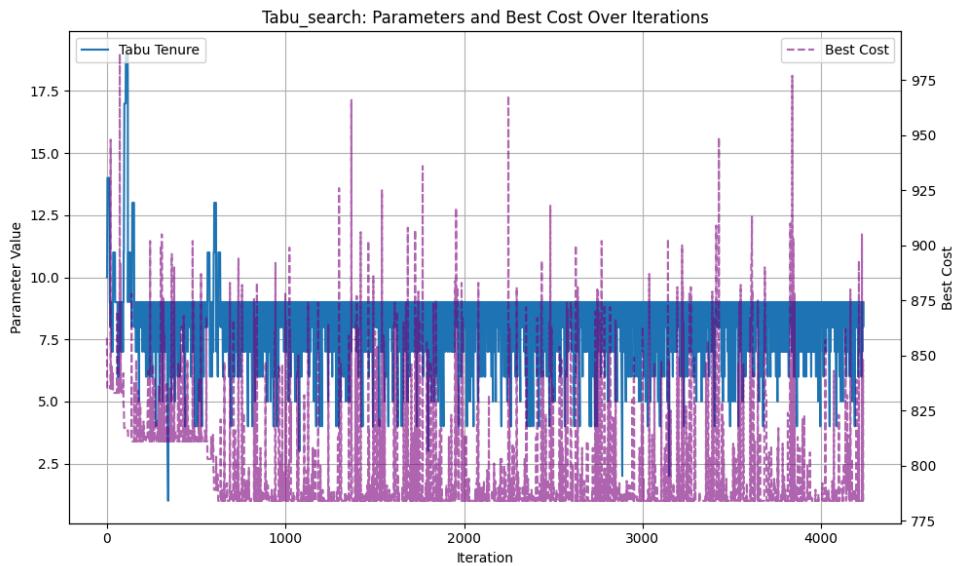
1 22 32 20 18 14 8 27 1

Routes Comparison





דוגמא לשינוי פרמטרים:



A-n80-k10

Algorithm: ILS, Metaheuristic: ACO + Simulated Annealing + Tabu Search

|| max_searches: 9000, time_limit: 1800 seconds

Best cost per metaheuristic: | tabu_search: 1804 | aco: 2174 |

simulated_annealing: 2209

Total cost: 1804

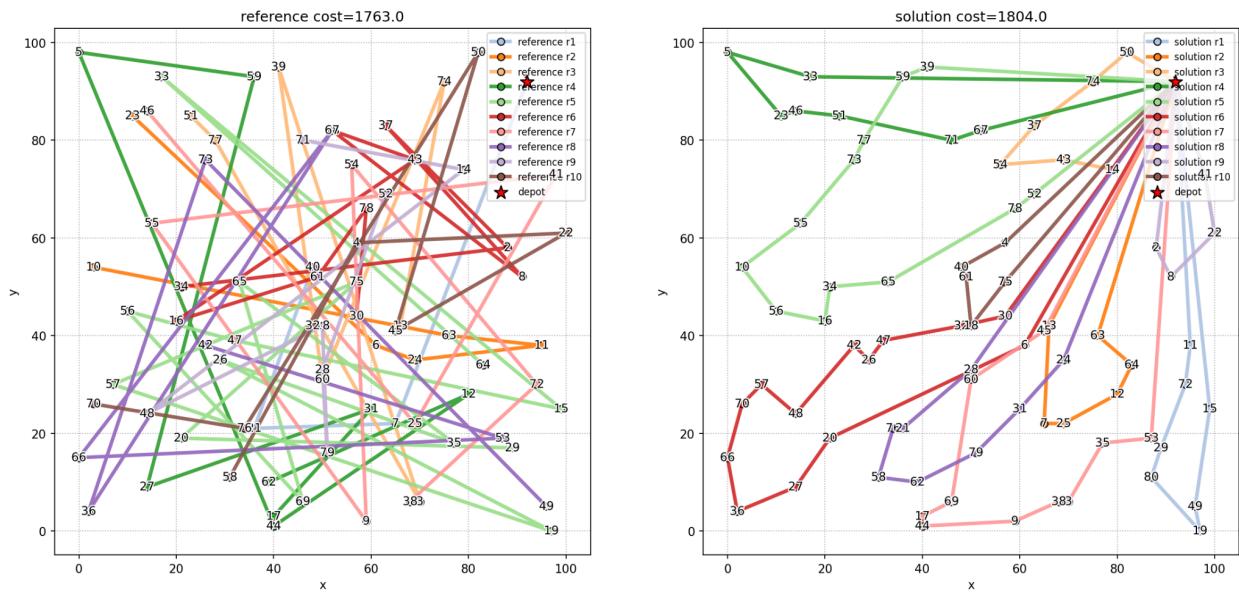
Optimal cost: 1763.0

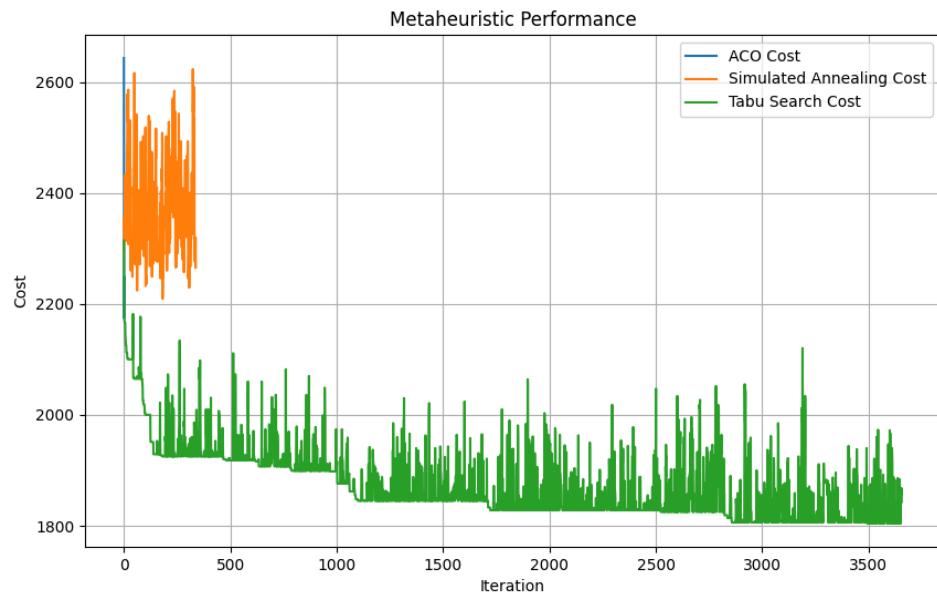
Difference: 41.00

Solution routes:

1 11 72 29 80 19 49 15 1
 1 63 64 12 25 7 13 1
 1 14 43 54 37 74 50 1
 1 33 5 23 46 51 71 68 67 1
 1 52 78 65 34 16 56 10 55 73 77 59 39 1
 1 30 32 47 26 42 48 57 70 66 36 27 20 6 1
 1 53 35 3 38 9 44 17 69 60 45 1
 1 24 31 79 62 58 76 21 28 1
 1 41 22 8 2 1
 1 4 40 61 18 75 1

Routes Comparison





- בעיות advanced בעיה 17 : שימוש ב- MSH לא הניב פתרון ראשוני טוב יותר מאשר **Most Connected Node + Nearest Neighbor**.
- בעיה 25 : X-n101-k25

Problem: X-n101-k25

Algorithm: ILS, Metaheuristic: ACO + Simulated Annealing + Tabu Search ||

max_searches: 20000, time_limit: 3600 seconds

Best cost per metahuristic: | tabu_search: 28578 | aco: 30954 |

simulated_annealing: 35384

Total cost: 28578

Optimal cost: 27591.0

Difference: 987.00

Solution routes:

1 2 71 55 1

1 9 35 13 1

1 15 29 83 78 1

1 21 42 23 16 1

1 24 51 80 1

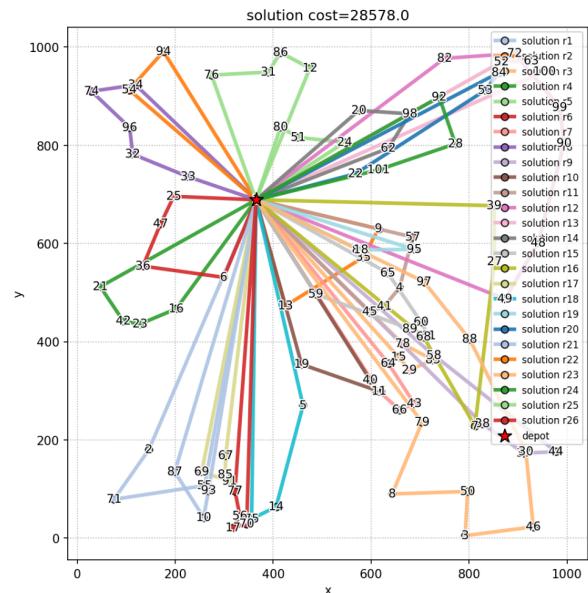
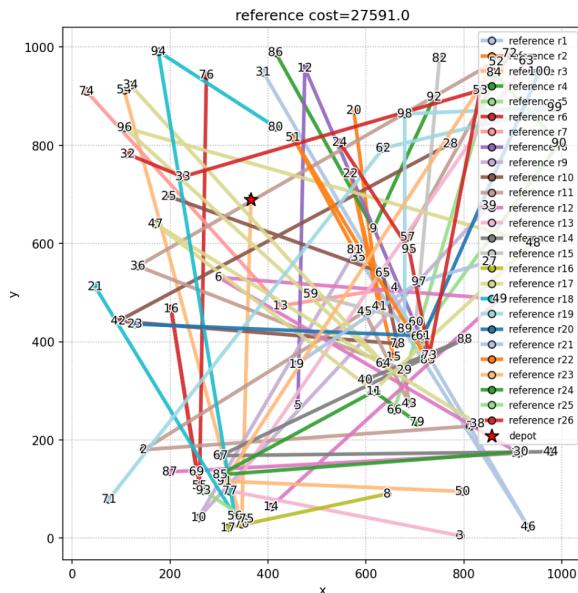
1 25 47 36 6 1

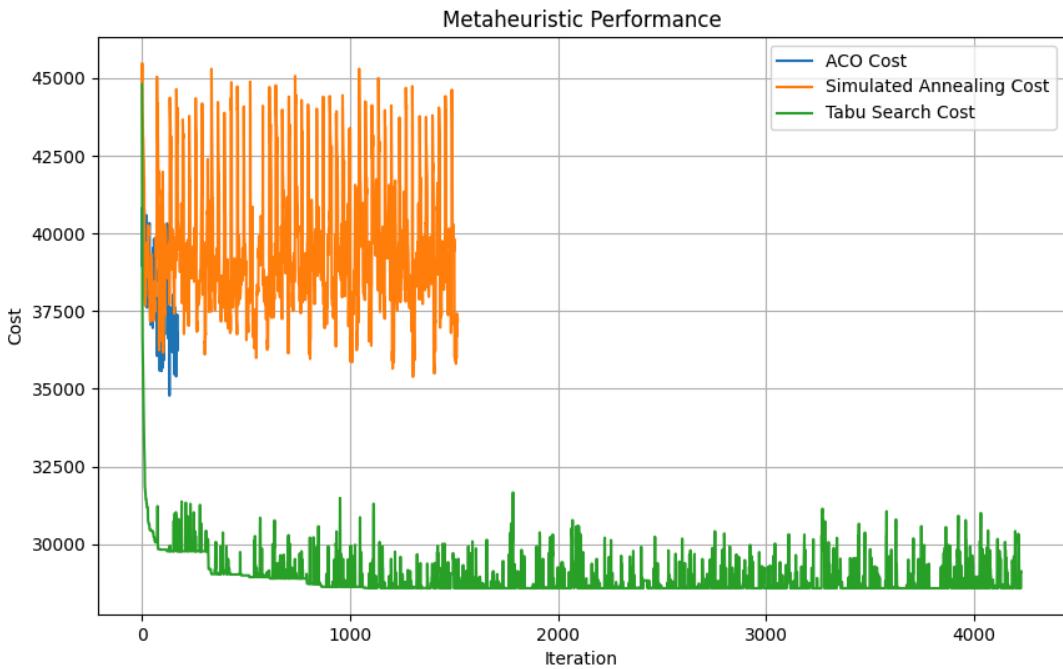
1 26 66 43 64 1

1 33 32 96 74 34 1

1 38 37 44 73 58 1
 1 40 11 19 1
 1 45 41 4 57 1
 1 49 48 90 99 72 82 1
 1 52 63 100 1
 1 62 98 20 1
 1 65 60 61 89 59 1
 1 68 7 27 39 1
 1 69 91 85 67 1
 1 75 14 5 1
 1 81 18 95 1
 1 84 53 22 1
 1 93 10 87 1
 1 94 54 1
 1 97 88 30 46 3 50 8 79 1
 1 101 28 92 1
 1 76 31 86 12 1
 1 77 56 17 70 1

Routes Comparison





מסקנות:

- **שימוש ב- MSH :**

הפתרון הראשון אינטואיטיבי יותר מאשר הפתרון של שיטת **Most Connected Node + Nearest Neighbor** וזה אפשר לאלגוריתם LS עם מטה היוריסטיΚות להתחיל את האלגוריתם מנקודת מוצא הרבה יותר חזקה ולכן משפר את זמן התכנסות.

אך, בעיות קשות ברמת advanced זמן הריצה התארך בגלל MSH והוצאות לא היו טובות מהשיטה המקורית: **Most Connected Node + Nearest Neighbor**.

- **פרמטרים דינמיים:** תורמים לשיפור האיזון בין חיפוש לניצול (שיפור הפתרון הקיטם).

- שימוש בדינמיות של rho/rho/beta/beta ב-ACO שיפור את יכולת יצא ממינימום מקומי בעיות ביןוניות.
- ב-SA, טמפרטורה דינמית מאפשרת שמירה על איזון בין קפיצות אקראיות להכנסות הדרגתית
- ב-TS ,

- **aicoot הפתרונות :**

הmeta היוריסטיκה עם הפתרונות הכח טוביים - Tabu Search .
אחוז ההפרש מחושב בצורה הבאה:

$$\% \Delta = \left(\frac{(C_{total} - C_{opt})}{C_{opt}} \right) \times 100$$

● Beginner

ב- 100% מהבעיות הפתרון הסופי שווה ערך לפתרון האופטימלי.

מספר פעמים SA ו-ACO לא הגיעו לאופטימום אך הם קרובים ו- TS תמיד מוצא את האופטימום.

הסיבה לכך - O ACO מהיר ולכן מסיים מוקדם אך לא מגיע לאופטימום. מאחר ומספר ההרצות המקסימלי זמן הריצה מותאמים יותר ל- TS ו- SA שמצריכים זמן רב יותר, העלנו את מספר ההרצות של ACO כדי שיגיע לאופטימום. ואילו עבור SA ניתן לראות שיש שינויים גדולים באיכות הפתרונות לאורך הריצה בגלל היכולת שלו לאפשר פתרונות "גראויים".

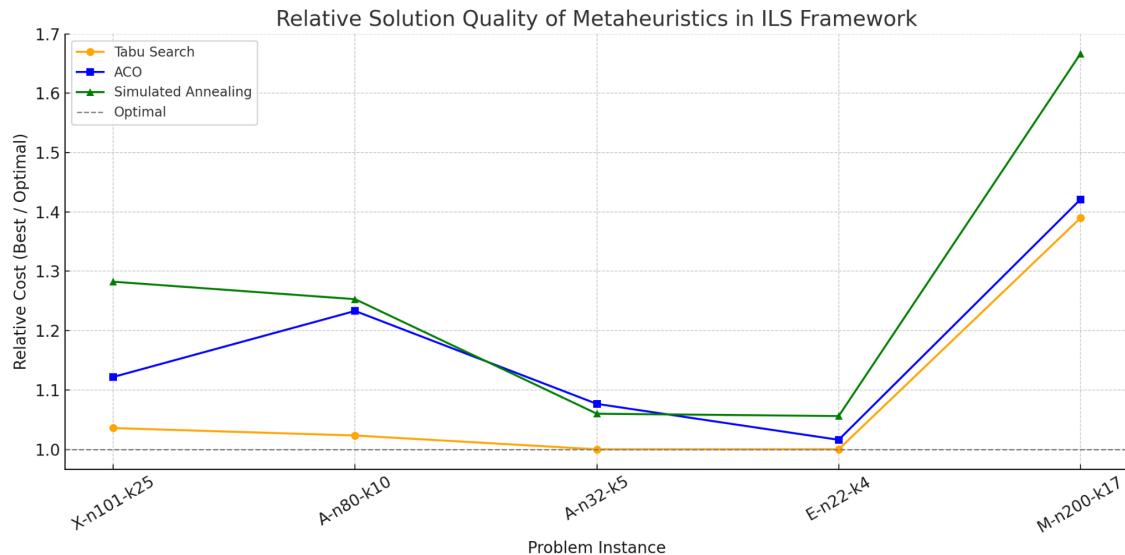
● Intermediate

ב- 50% מהבעיות הפתרון הסופי שווה ערך לפתרון האופטימלי. העלנו את מספר ההרצות וזמן הריצה - זה שיפור את התוצאות: **A-n80-k10**: שבו הפער מהפתרון הסופי לפתרון המיטבי ירד מ- 5.67% ל- 2.33%.

● Advanced

בעיה k17-n200-M: בדוגמה הפער 38.98% בין הפתרון הסופי לאופטימום. במהלך הריצות הפער נשאר +30%. כל המטה הירושית מתקשות למצוות את הפער - ניתן לראות כי האלגוריתם סיים את הריצה מבלי שסיים את מספר החיפושים המקסימלי וזאת בשל קשיי הבעיה. נתנו זמן ריצה מקסימלי של שעה לכל בעיה.

בעיה k25-n101-X: הפער הוא 3.577% בשימוש עם MSH.



- ❖ התוצאות מצביעות על כך Sh-TS מתמודד טוב יותר עם בעיות גדולות ומבודדות, כנראה בזכות מנגן זיכרון והימנעות מהחזרות.
- ❖ CO ACO מהיר, אך דורש התאמות של פרמטרים (מספר חיפושים וזמן) כדי להגיע לתוצאות תחרותיות.
- ❖ SA מציג פוטנציאלי שיפור לאורך זמן, אך דורש משאבים רבים יותר מאשר להתכנס לפתרון טוב.

• זמן ריצה :

- זמן הריצה הכללי של האלגוריתם הוא זמן הריצה המקסימלי בהתאם לבעיה- בעיות קלות לוקחות עד כ- 10 דקות, بينما אחרות 30-20 דקות וקשות כ-60 דקות.
- אם מסתכלים על היוריסטיות בנפרד, CO ACO הći מהירה ולכן צריך לתת לה יותר איטרציות לחיפושים. TS ו-SA איטרציות יותר אך מאוחר ו-TS מפיקה פתרונות איקוטיים לא צריך להוריד לה את מספר האיטרציות.
- שיפור זמן הריצה: ניתן להוריד את זמן הריצה של SA , אך לא הינו ממיליצים על כך מאוחר של-SA יש יתרונות ביציאה ממינים מוקומי זהה חיוני עבור בעיות כמו CVRP.

• הרצה במקביל:

הרצתה מקבילתית של מספר מטה-היוריסטיות (SA, ACO, TS), תוך שמירה על הפתרון הכי טוב מהן. כל אלגוריתם פועל עם אותו תקציב זמן ומחפש פתרון עצמאי.

הרצתה של כל אחת מהmeta-היוריסטיות במקביל מאפשרת:

- חיפוש מגוון
- ניצול יעיל של זמן CPU

- קבלת פתרונות משלימים זה לזה
- נבחר הפתרון הטוב ביותר מכל השיטות
- הערה : בריצת EXE ריצה במקביל לא עבדת לפעם ולכון האלגוריתם מריץ את ACO,SA,TS אחת אחרי השניה - זמן ההריצה של כל אחד מתקזזים בהתאם.

Genetic Algorithm with Islands Model

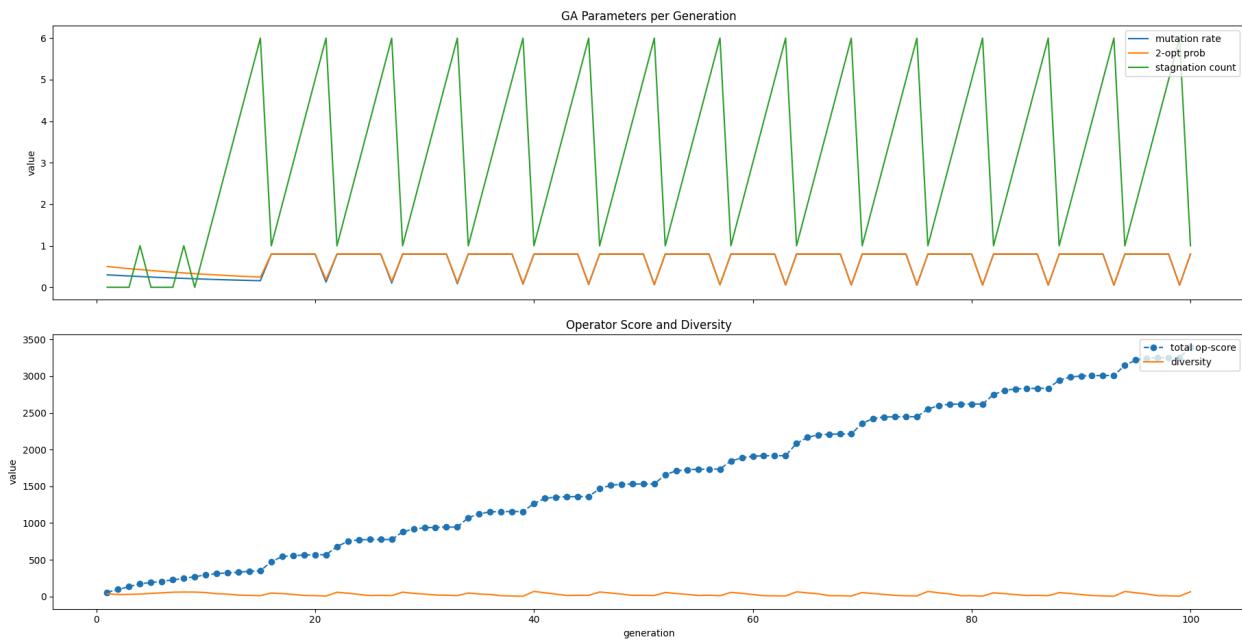
עברית בעברית בקטגוריות • Beginner

- נתנו לכל בעיה לרווח זמן יחסית קצר, בין 10-5 דקות (לכל בעיה)
- הפרמטרים עבור ריצה זו:

- Population size (per island) = 200
- Number of generations = 100
- Number of islands = 8
- Number of migrants per interval = 6
- Migrate every 6 generations
- Base mutation rate = 0.3
- Base two-opt rate = 0.5
- Stagnation threshold = 6
- Hypermode len = 5 (how many gens hyper mode runs)
- Crossover operation = Sequential Constructive Crossover(SCX)
- Age threshold = 20

להלן התוצאות:

E-n22-k4:



```
[INFO] ===== problem E-n22-k4.vrp =====
[INFO] parsed 'E-n22-k4.vrp' → cap=6000, nodes=22, depot=1, minVeh=4
[INFO] reference .sol found - 4 routes, cost=375.0
[INFO] --- GA ---

Starting GA with 8 islands, total 200 per island, 100 gens, 0.3 base mutation rate, scx crossover

[ga] gen 01/100  mut=0.30 twoopt=0.50  hyper=no
[ga] gen 02/100  mut=0.29 twoopt=0.47  hyper=no
[ga] gen 03/100  mut=0.27 twoopt=0.45  hyper=no
[ga] gen 04/100  mut=0.26 twoopt=0.43  hyper=no
[ga] gen 05/100  mut=0.25 twoopt=0.40  hyper=no
[ga] gen 06/100  mut=0.24 twoopt=0.38  hyper=no
[ga] gen 07/100  mut=0.22 twoopt=0.36  hyper=no
[ga] gen 08/100  mut=0.21 twoopt=0.35  hyper=no
[ga] gen 09/100  mut=0.20 twoopt=0.33  hyper=no
[ga] gen 10/100  mut=0.20 twoopt=0.31  hyper=no
[ga] gen 11/100  mut=0.19 twoopt=0.30  hyper=no
[ga] gen 12/100  mut=0.18 twoopt=0.28  hyper=no
[ga] gen 13/100  mut=0.17 twoopt=0.27  hyper=no
[ga] gen 14/100  mut=0.16 twoopt=0.26  hyper=no
[ga] gen 15/100  mut=0.16 twoopt=0.24  hyper=no
[ga] gen 16/100  mut=0.80 twoopt=0.80  hyper=yes
[ga] gen 17/100  mut=0.80 twoopt=0.80  hyper=yes
```

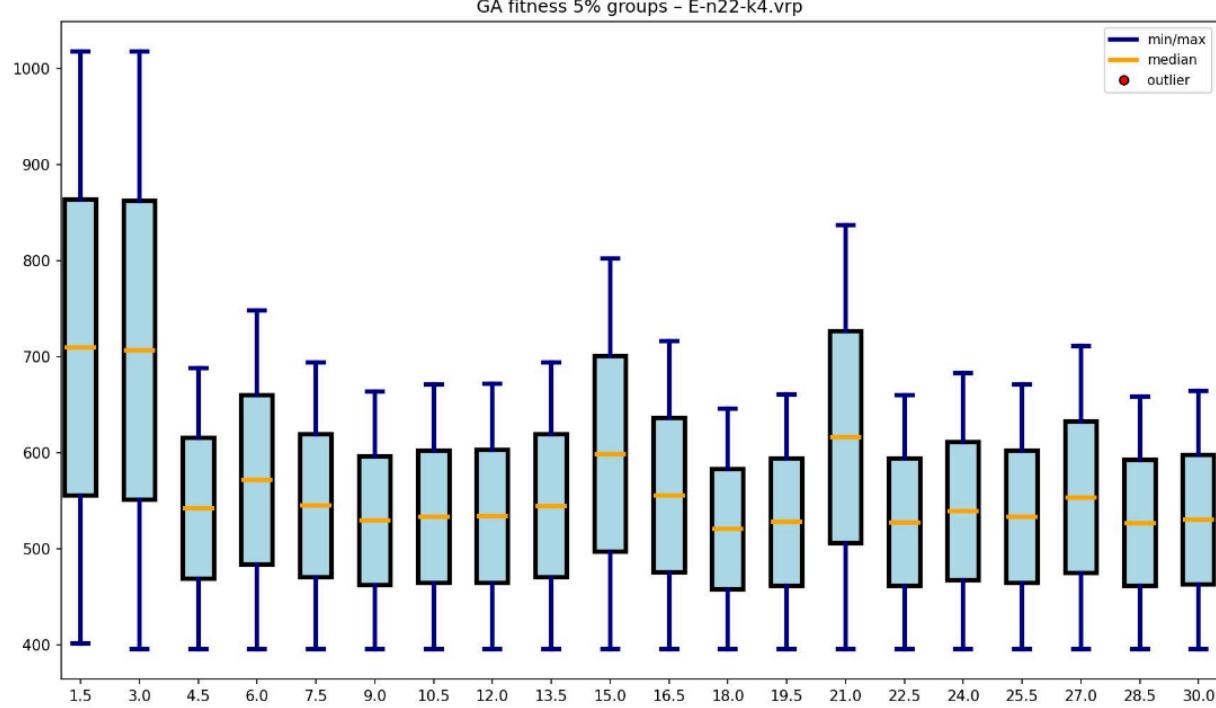
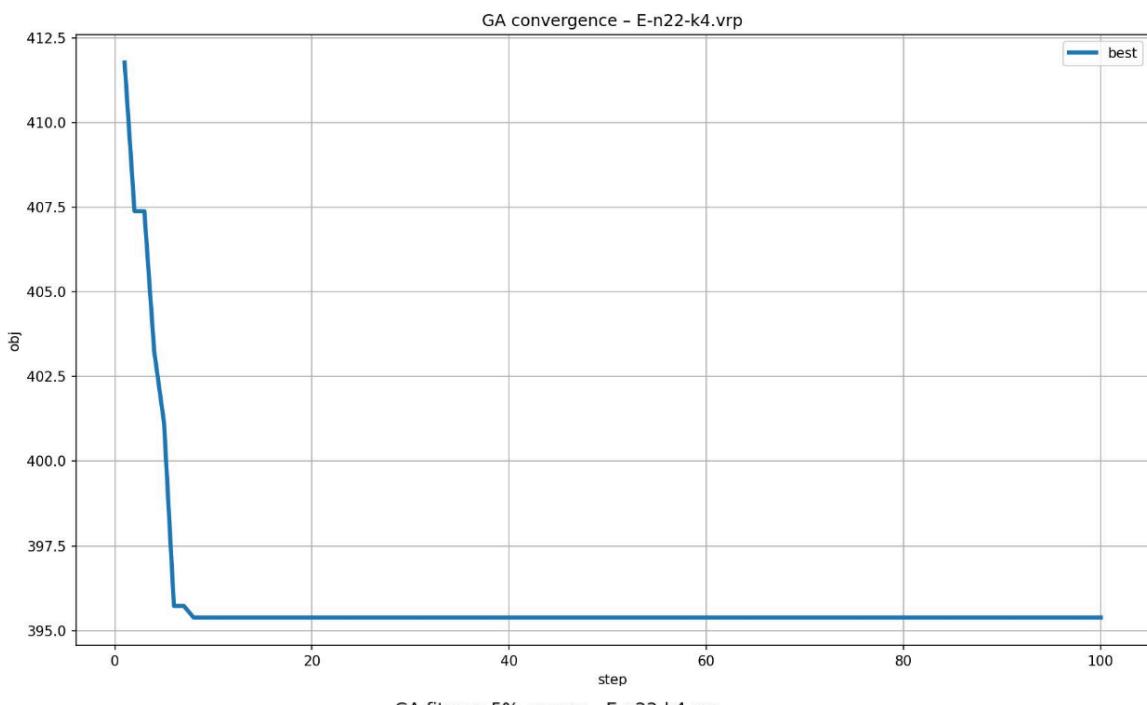
```
[ga] gen 99/100  mut=0.05 twoopt=0.05  hyper=no
[ga] gen 100/100  mut=0.80 twoopt=0.80  hyper=yes
```

[INFO] GA runtime: 450.25s

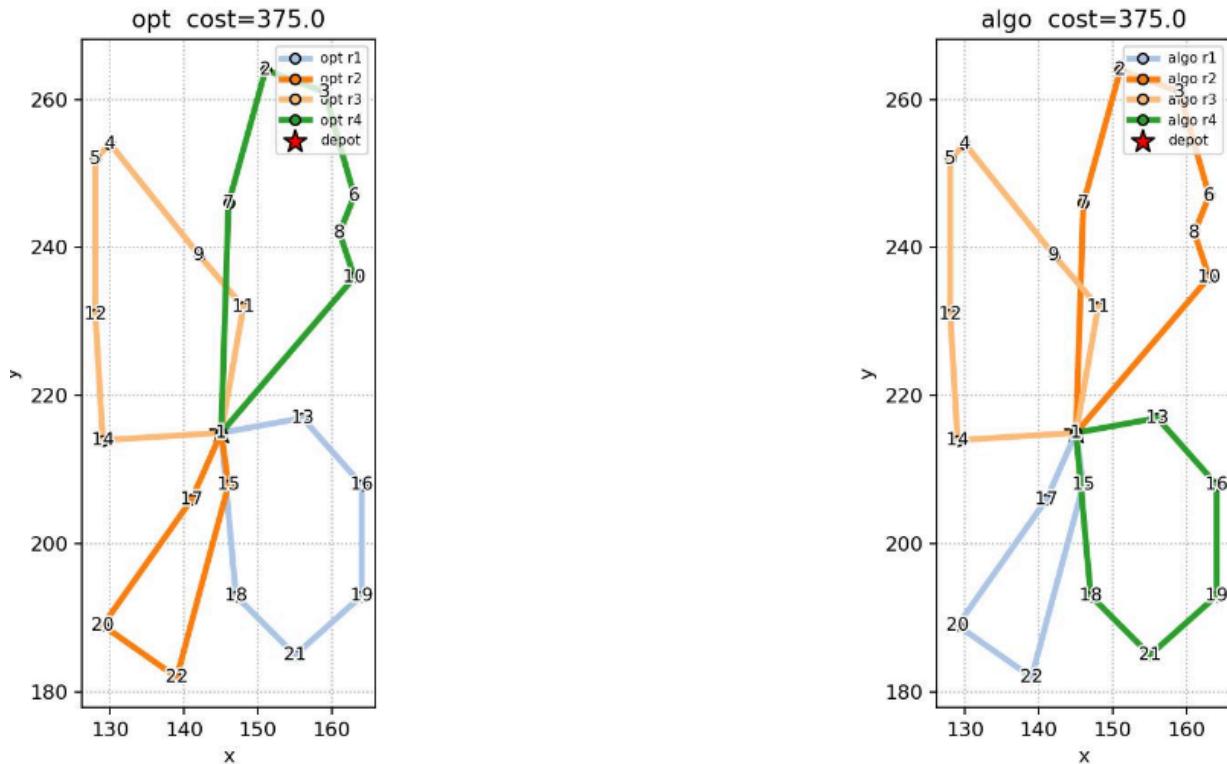
[RESULT] solution:

```
route 1: 1 → 15 → 22 → 20 → 17 → 1
route 2: 1 → 10 → 8 → 6 → 3 → 2 → 7 → 1
route 3: 1 → 11 → 9 → 4 → 5 → 12 → 14 → 1
route 4: 1 → 13 → 16 → 19 → 21 → 18 → 1
total cost: 375.00
```

time: 450.25s



GA - E-n22-k4.vrp



P-n16-k8:

```
[INFO] ===== problem P-n16-k8.vrp =====
[INFO] parsed 'P-n16-k8.vrp' → cap=35, nodes=16, depot=1, minVeh=None
[INFO] reference .sol found - 8 routes, cost=450.0
[INFO] --- GA ---

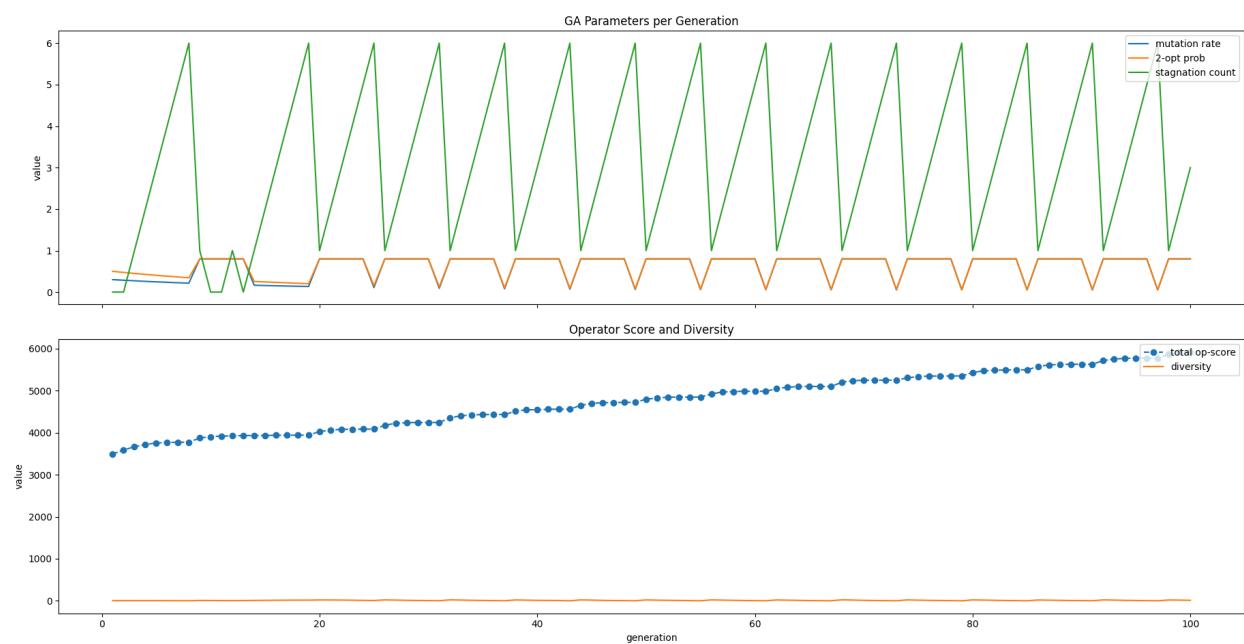
Starting GA with 8 islands, total 200 per island, 100 gens, 0.3 base mutation rate, scx crossover

[ga] gen 01/100  mut=0.30 twoopt=0.50  hyper=no
[ga] gen 02/100  mut=0.29 twoopt=0.47  hyper=no
[ga] gen 03/100  mut=0.27 twoopt=0.45  hyper=no
[ga] gen 04/100  mut=0.26 twoopt=0.43  hyper=no
[ga] gen 05/100  mut=0.25 twoopt=0.40  hyper=no
```

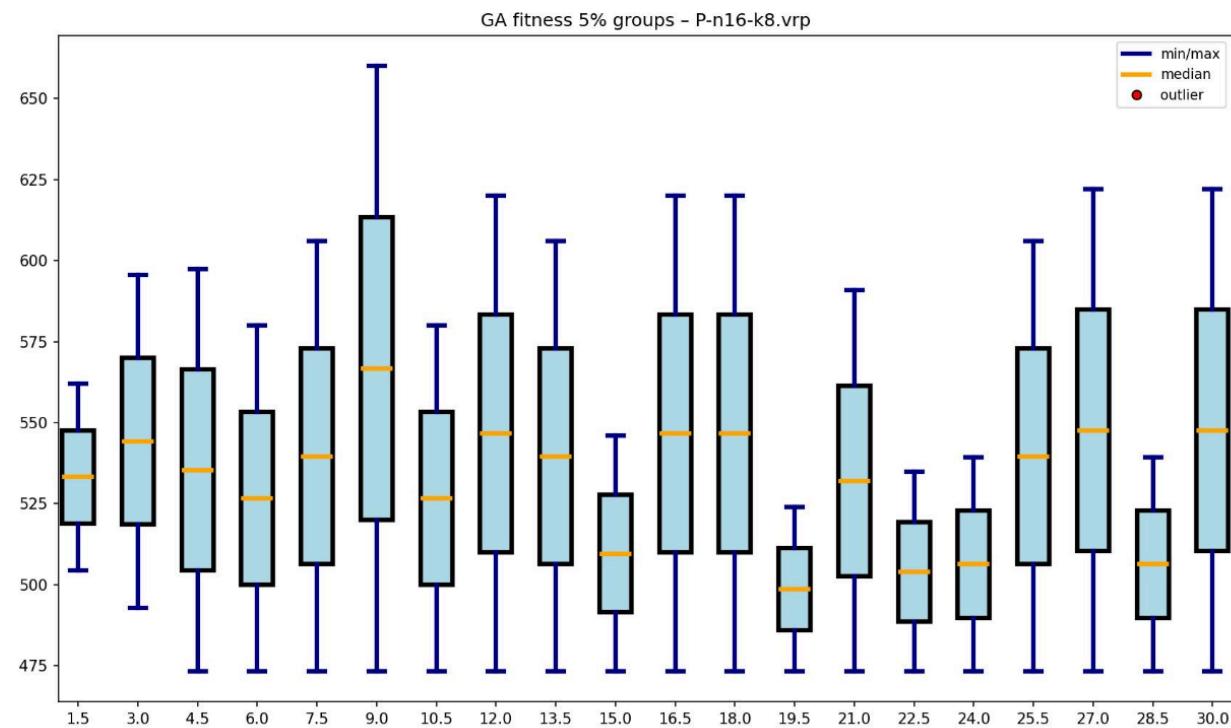
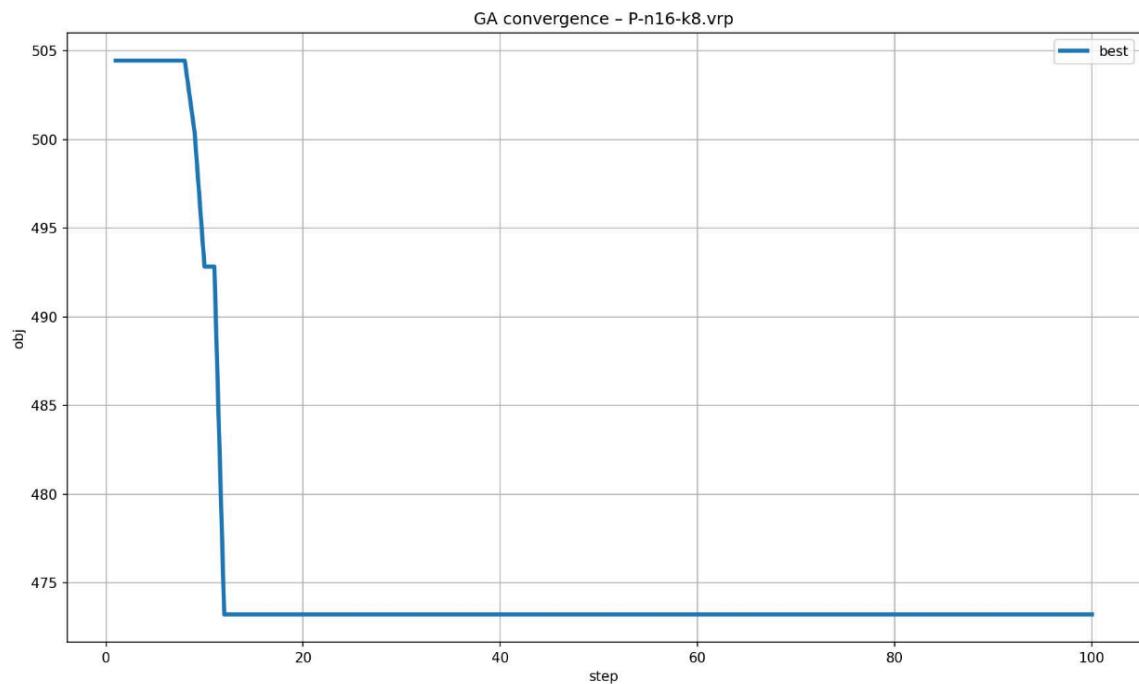
```

[ga] gen 97/100  mut=0.05 twoopt=0.05 hyper=no
[ga] gen 98/100  mut=0.80 twoopt=0.80 hyper=yes
[ga] gen 99/100  mut=0.80 twoopt=0.80 hyper=yes
[ga] gen 100/100  mut=0.80 twoopt=0.80 hyper=yes
[INFO] GA runtime: 478.09s
[RESULT] solution:
route 1: 1 → 11 → 13 → 16 → 1
route 2: 1 → 7 → 1
route 3: 1 → 3 → 1
route 4: 1 → 6 → 15 → 1
route 5: 1 → 12 → 5 → 1
route 6: 1 → 9 → 1
route 7: 1 → 14 → 10 → 8 → 1
route 8: 1 → 4 → 2 → 1
total cost: 450.00

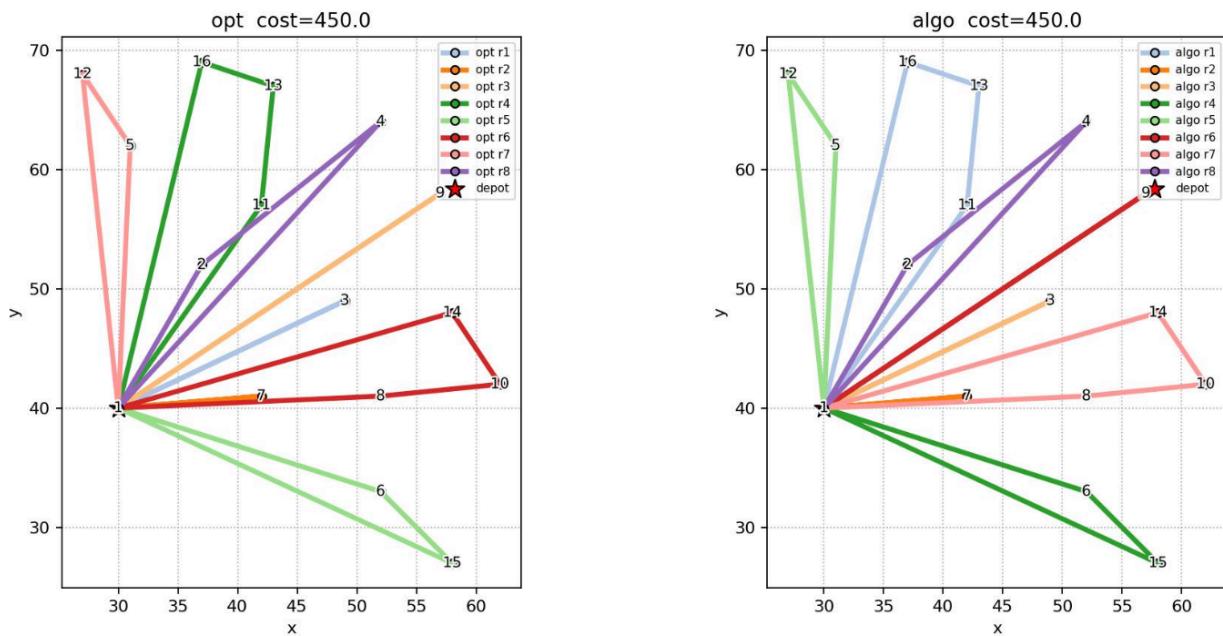
```



time: 478.09s



GA - P-n16-k8.vrp



עבור הביעית בקטגוריות Intermediate

- נתנו לכל בעיה לרווח משך זמן בינוני, בין 20-10 דקות (לכל בעיה)
- הפרמטרים עבור ריצה זו:

- Population size (per island) = **250**
- Number of generations = 100
- Number of islands = 8
- Number of migrants per interval = 6
- Migrate every 6 generations
- Base mutation rate = 0.3
- Base two-opt rate = 0.5
- Stagnation threshold = 6
- Hypermode len = 5 (how many gens hyper mode runs)
- Crossover operation = Sequential Constructive Crossover(SCX)
- Age threshold = 20

להלן התוצאות:

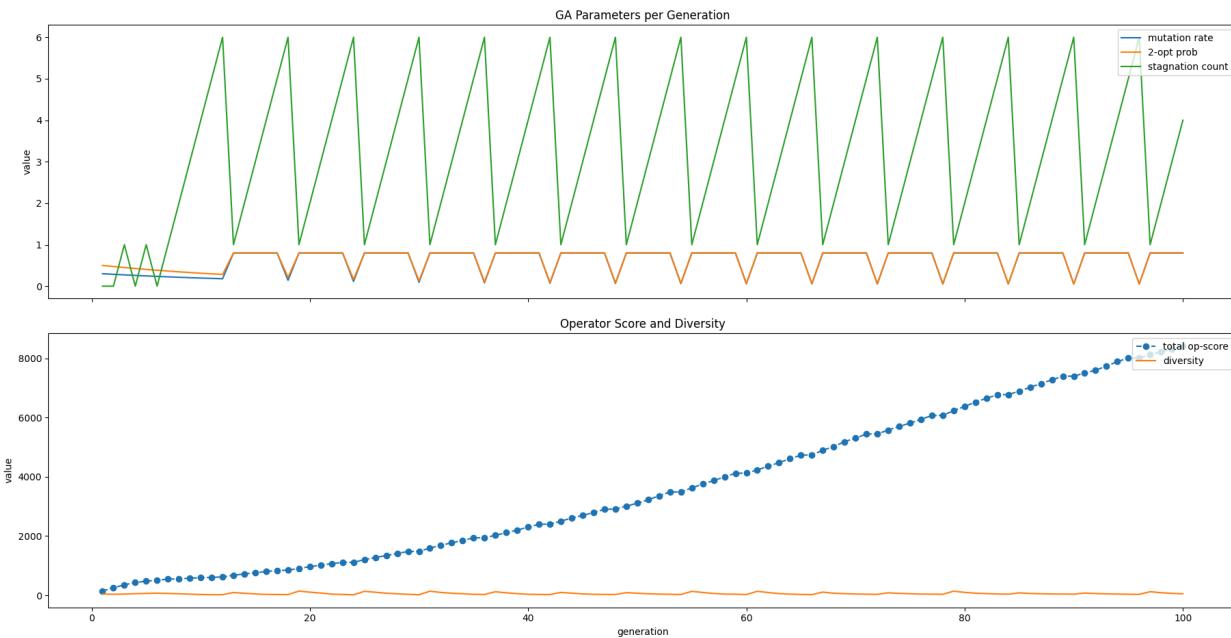
A-n32-k5

```
[INFO] ===== problem A-n32-k5.vrp =====
[INFO] parsed 'A-n32-k5.vrp' → cap=100, nodes=32, depot=1, minVeh=None
[INFO] reference .sol found - 5 routes, cost=784.0
[INFO] --- GA ---

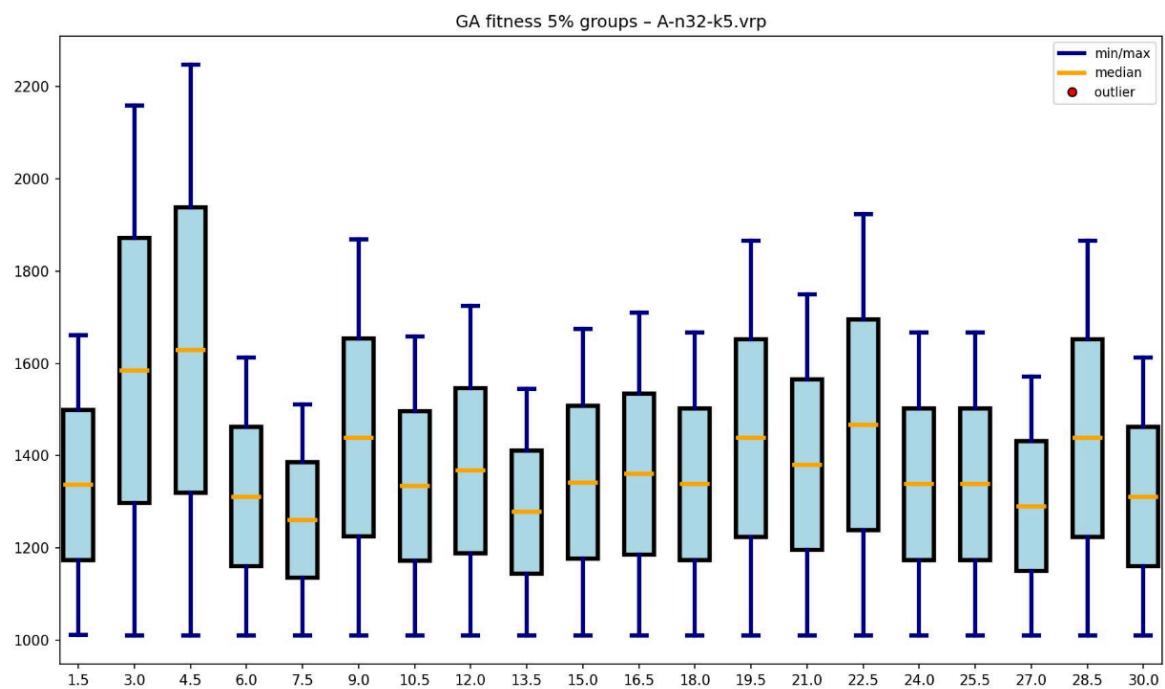
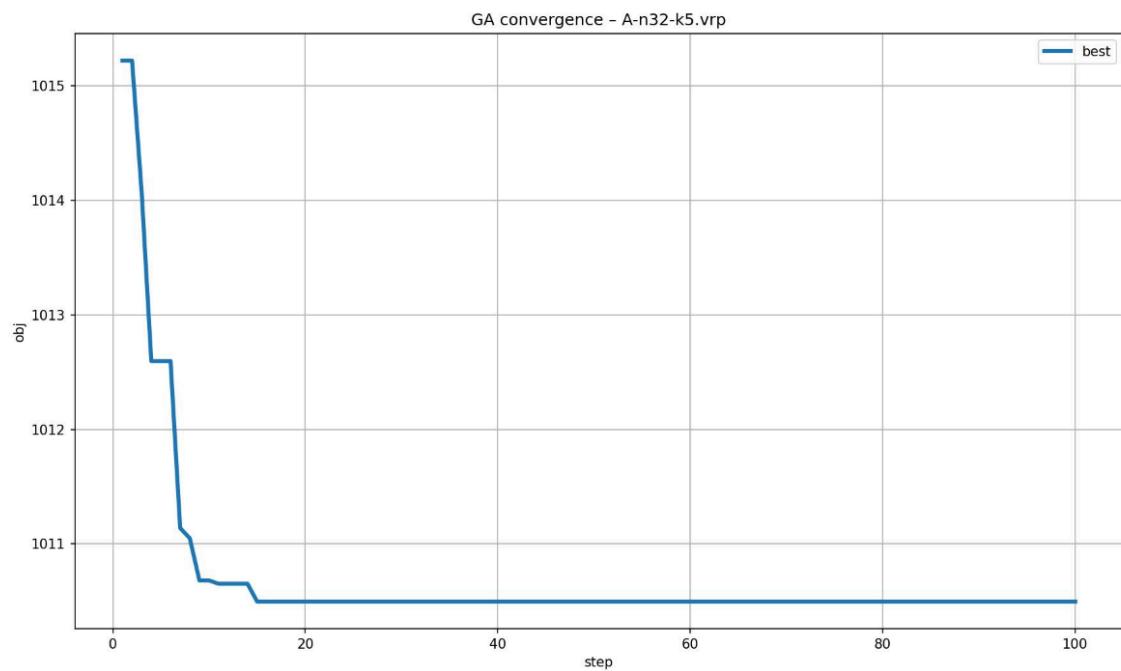
Starting GA with 8 islands, total 250 per island, 100 gens, 0.3 base mutation rate, scx crossover

[ga] | gen 01/100 | mut=0.30 | twoopt=0.50 | hyper=no | elapsed time = 0.00s
[ga] | gen 02/100 | mut=0.29 | twoopt=0.47 | hyper=no | elapsed time = 8.20s
[ga] | gen 03/100 | mut=0.27 | twoopt=0.45 | hyper=no | elapsed time = 16.53s

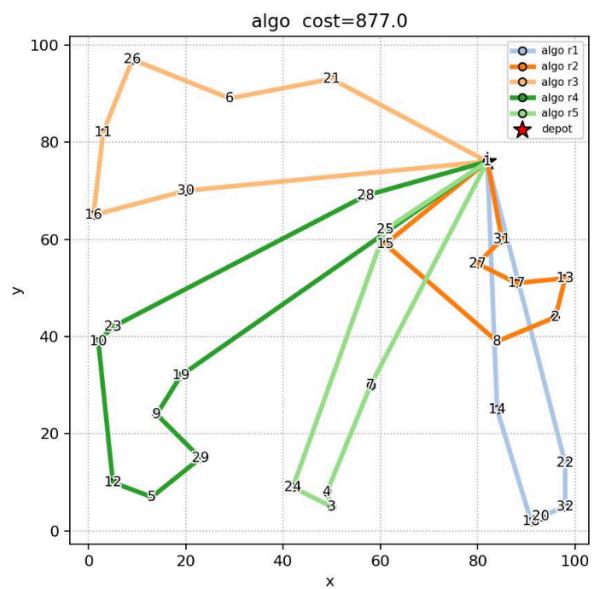
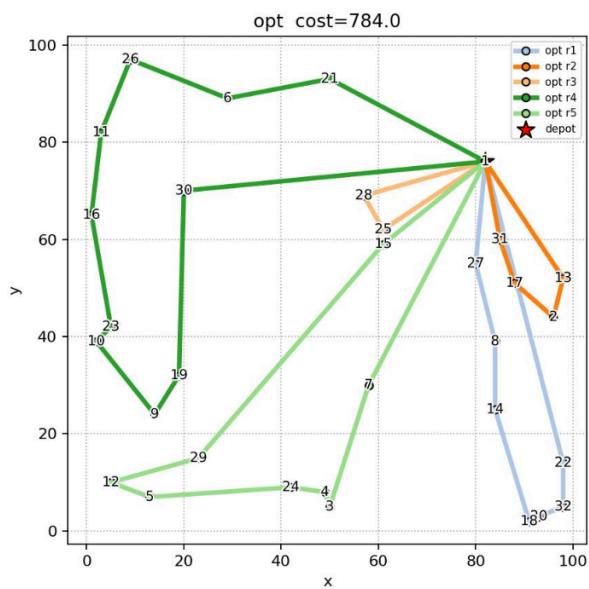
[ga] | gen 97/100 | mut=0.05 | twoopt=0.05 | hyper=no | elapsed time = 1024.28s
[ga] | gen 98/100 | mut=0.80 | twoopt=0.80 | hyper=yes | elapsed time = 1041.14s
[ga] | gen 99/100 | mut=0.80 | twoopt=0.80 | hyper=yes | elapsed time = 1050.13s
[ga] | gen 100/100 | mut=0.80 | twoopt=0.80 | hyper=yes | elapsed time = 1059.36s
[INFO] GA runtime: 1074.25s
[RESULT] solution:
route 1: 1 → 14 → 18 → 20 → 32 → 22 → 1
route 2: 1 → 31 → 27 → 17 → 13 → 2 → 8 → 15 → 1
route 3: 1 → 21 → 6 → 26 → 11 → 16 → 30 → 1
route 4: 1 → 19 → 9 → 29 → 5 → 12 → 10 → 23 → 28 → 1
route 5: 1 → 7 → 4 → 3 → 24 → 25 → 1
total cost: 877.00
```



time: 1074.25s



GA - A-n32-k5.vrp



A-n80-k10

- כדי להפוך את זמן הריצה לאפשרי ולא יותר מדי, צמצנו את גודל האוכלוסייה ל-150 (לכל אי, עברו 8 איים)

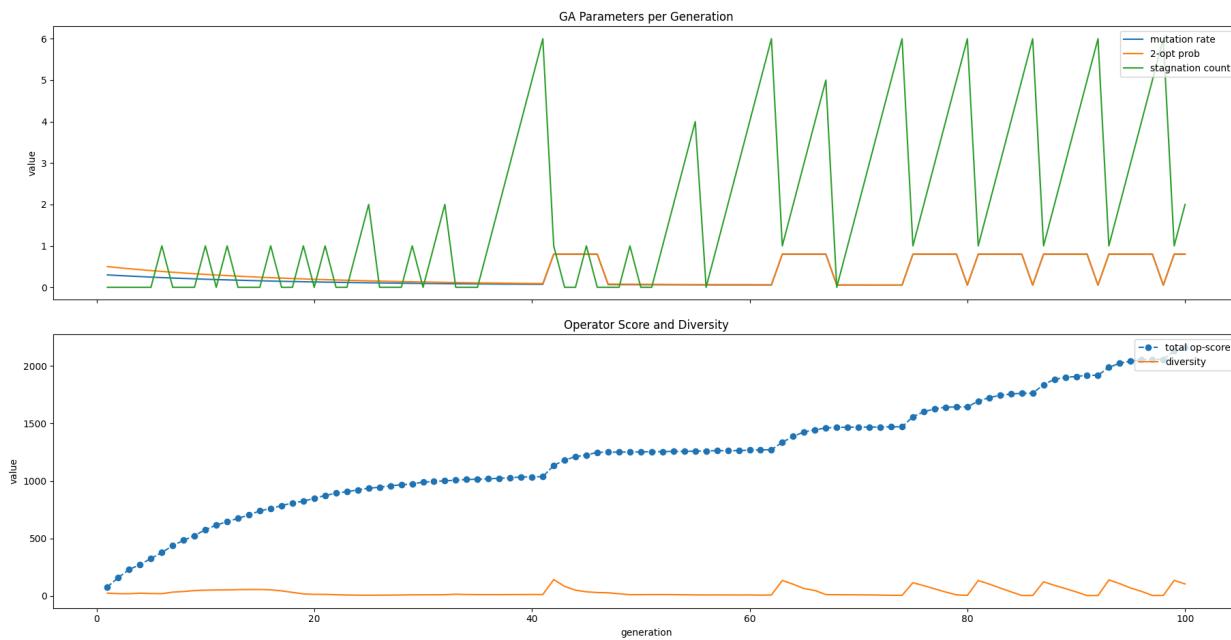
```
[INFO] ===== problem A-n80-k10.vrp =====
[INFO] parsed 'A-n80-k10.vrp' → cap=100, nodes=80, depot=1, minVeh=None
[INFO] reference .sol found - 10 routes, cost=1763.0
[INFO] --- GA ---

Starting GA with 8 islands, total 150 per island, 100 gens, 0.3 base mutation rate, scx crossover

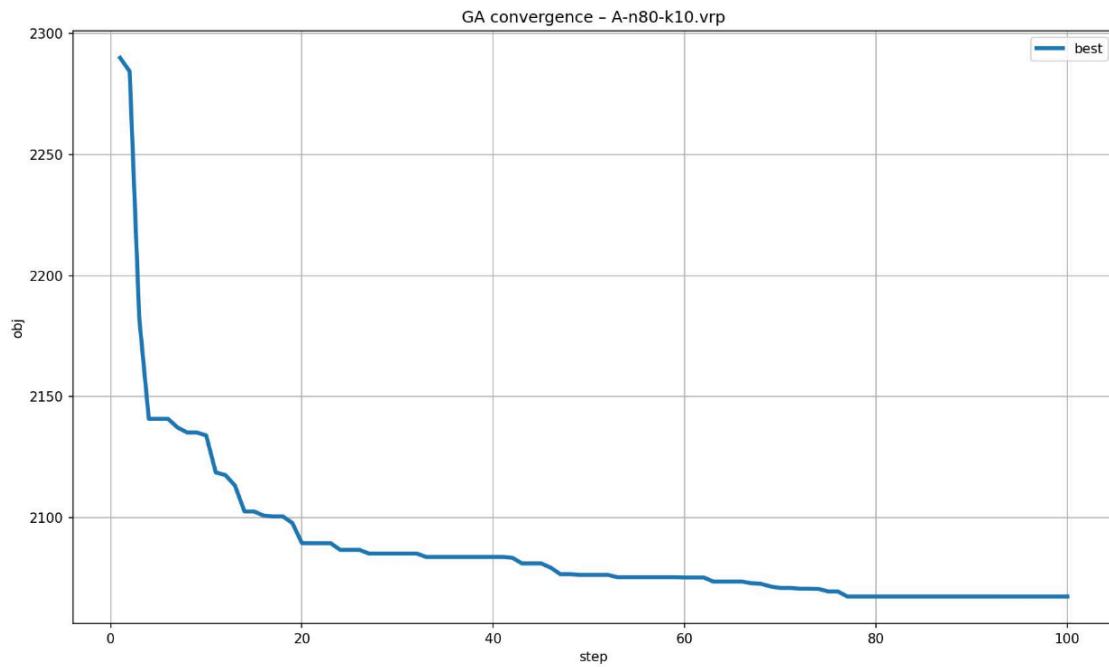
[ga] | gen 01/100 | mut=0.30 | twoopt=0.50 | hyper=no | elapsed time = 0.00s
[ga] | gen 02/100 | mut=0.29 | twoopt=0.47 | hyper=no | elapsed time = 13.24s
[ga] | gen 03/100 | mut=0.27 | twoopt=0.45 | hyper=no | elapsed time = 26.77s

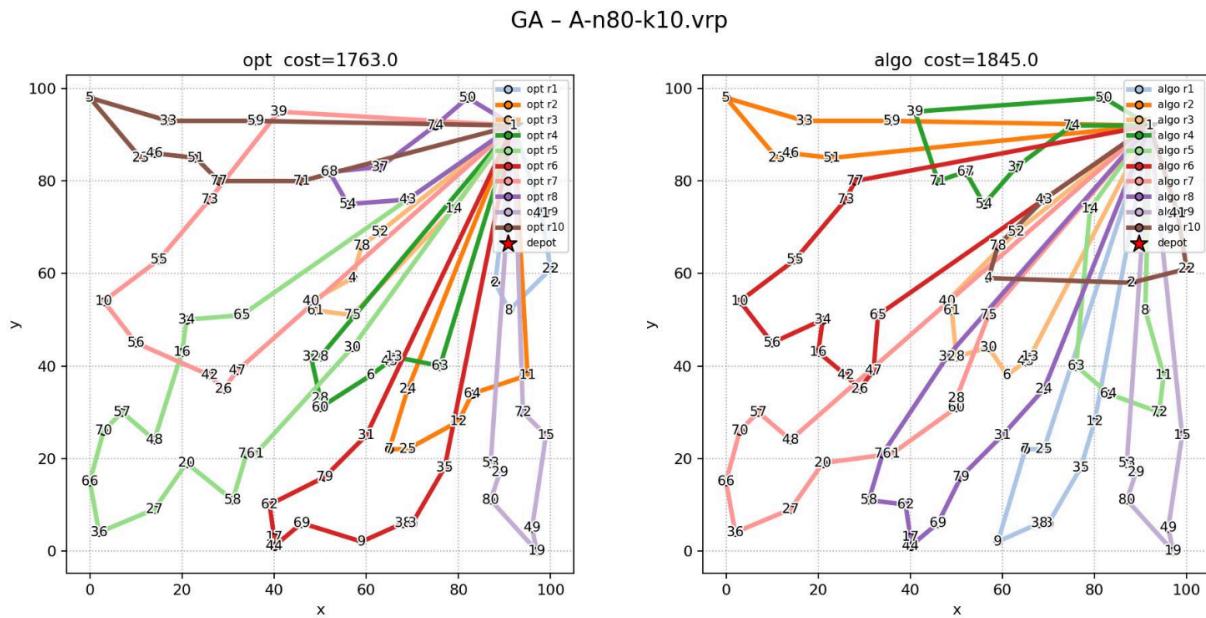
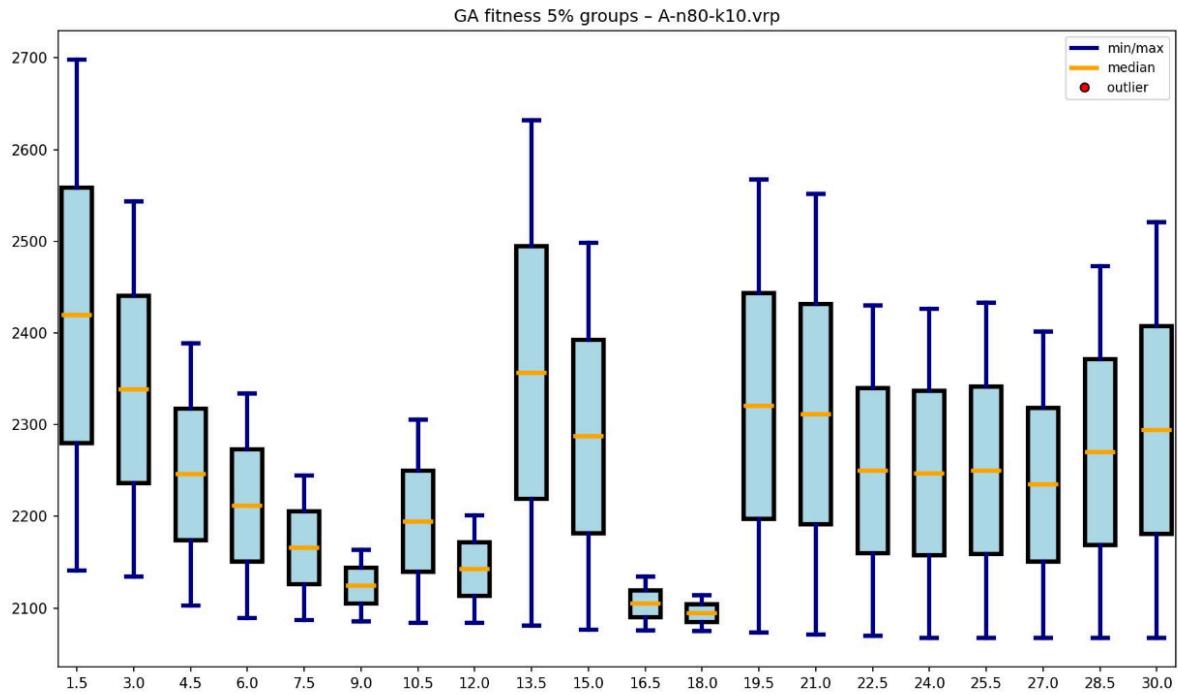
[ga] | gen 97/100 | mut=0.80 | twoopt=0.80 | hyper=no | elapsed time = 1525.38s
[ga] | gen 98/100 | mut=0.05 | twoopt=0.05 | hyper=no | elapsed time = 1535.99s
[ga] | gen 99/100 | mut=0.80 | twoopt=0.80 | hyper=yes | elapsed time = 1594.81s
[ga] | gen 100/100 | mut=0.80 | twoopt=0.80 | hyper=yes | elapsed time = 1606.68s
[INFO] GA runtime: 1962.28s

[RESULT] solution:
route 1: 1 → 12 → 35 → 3 → 38 → 9 → 7 → 25 → 1
route 2: 1 → 59 → 33 → 5 → 23 → 46 → 51 → 1
route 3: 1 → 52 → 40 → 61 → 18 → 30 → 6 → 45 → 13 → 1
route 4: 1 → 50 → 39 → 71 → 68 → 67 → 54 → 37 → 74 → 1
route 5: 1 → 14 → 63 → 64 → 72 → 11 → 8 → 1
route 6: 1 → 65 → 47 → 26 → 42 → 16 → 34 → 56 → 10 → 55 → 73 → 77 → 1
route 7: 1 → 48 → 57 → 70 → 66 → 36 → 27 → 20 → 21 → 60 → 28 → 75 → 1
route 8: 1 → 24 → 31 → 79 → 69 → 44 → 17 → 62 → 58 → 76 → 32 → 1
route 9: 1 → 53 → 29 → 80 → 19 → 49 → 15 → 1
route 10: 1 → 41 → 22 → 2 → 4 → 78 → 43 → 1
total cost: 1845.00
```



time: 1962.28s





• עבור הביעות בקטגורית Advanced

- נתנו לכל בעיה לrhoץ למשך זמן ארוך יותר, בין 60-60-30 דקות (לכל בעיה)
- הפרמטרים עבור ריצה זו:

- Population size (per island) = **50** (זמן ריצה התפוצצו)
- Number of generations = 100
- Number of islands = 8
- Number of migrants per interval =6
- Migrate every 6 generations
- Base mutation rate = 0.3
- Base two-opt rate = 0.5
- Stagnation threshold = 6
- Hypermode len = 5 (how many gens hyper mode runs)
- Crossover operation = Sequential Constructive Crossover(SCX)
- Age threshold = 20

להלן התוצאות:

M-n200-k17

```
[INFO] ===== problem M-n200-k17.vrp =====
[INFO] parsed 'M-n200-k17.vrp' → cap=200, nodes=200, depot=1, minVeh=None
[INFO] reference .sol found - 17 routes, cost=1275.0
[INFO] --- GA ---

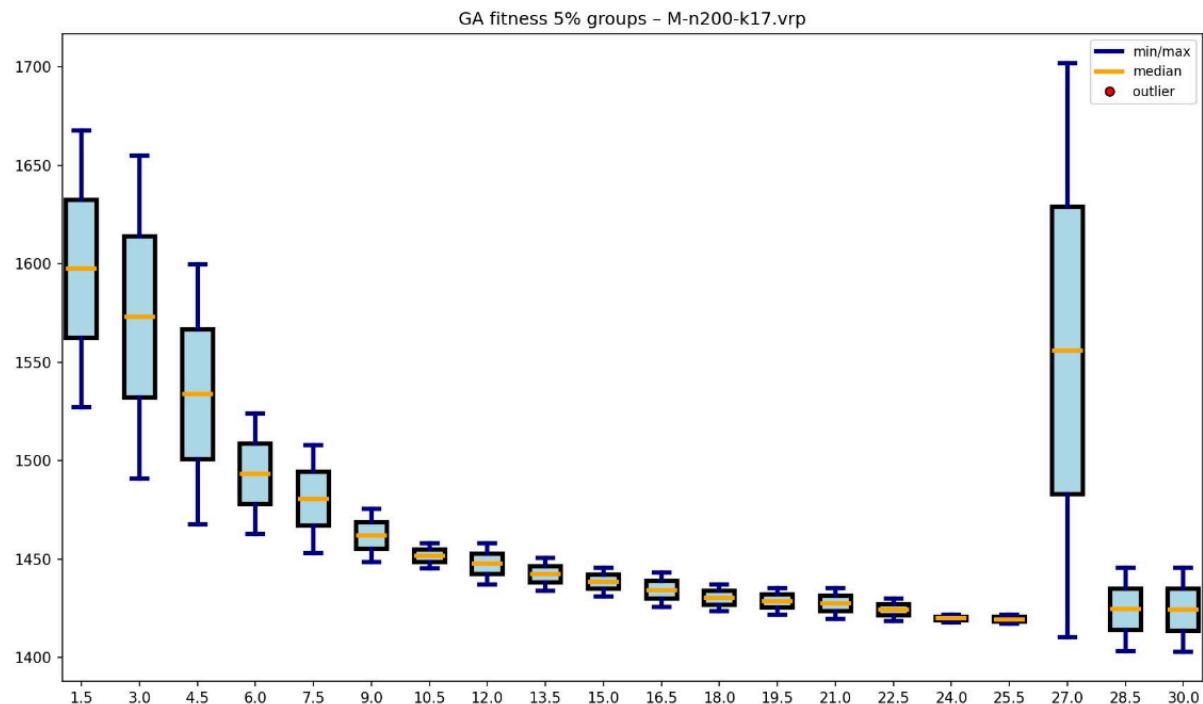
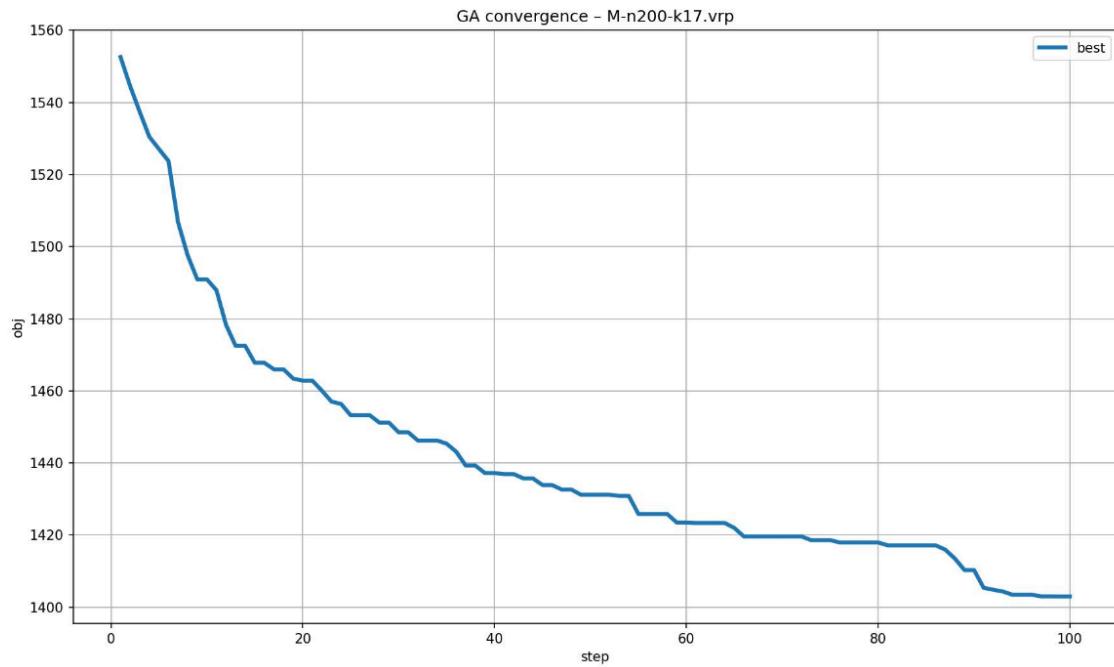
Starting GA with 8 islands, total 50 per island, 100 gens, 0.3 base mutation rate, scx crossover

[ga] | gen 01/100 | mut=0.30 | twoopt=0.50 | hyper=no | elapsed time = 0.00s
[ga] | gen 02/100 | mut=0.29 | twoopt=0.47 | hyper=no | elapsed time = 21.74s
[ga] | gen 03/100 | mut=0.27 | twoopt=0.45 | hyper=no | elapsed time = 45.14s
[ga] | gen 04/100 | mut=0.26 | twoopt=0.43 | hyper=no | elapsed time = 77.21s

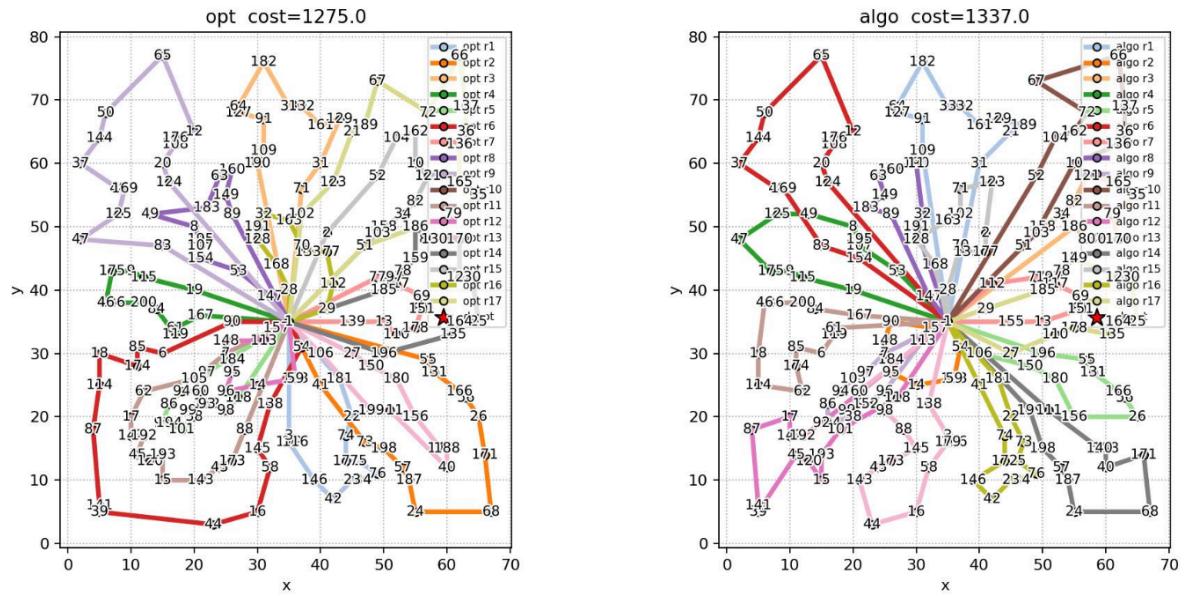
[ga] | gen 98/100 | mut=0.05 | twoopt=0.05 | hyper=no | elapsed time = 4384.23s
[ga] | gen 99/100 | mut=0.05 | twoopt=0.05 | hyper=no | elapsed time = 4420.79s
[ga] | gen 100/100 | mut=0.05 | twoopt=0.05 | hyper=no | elapsed time = 4456.22s
[INFO] GA runtime: 5323.93s

[RESULT] solution:
route 1: 1 → 31 → 21 → 189 → 129 → 161 → 132 → 33 → 182 → 64 → 127 → 91 → 109 → 1
route 2: 1 → 54 → 153 → 59 → 14 → 95 → 184 → 148 → 90 → 157 → 1
route 3: 1 → 51 → 158 → 34 → 82 → 121 → 165 → 35 → 79 → 170 → 130 → 80 → 186 → 1
route 4: 1 → 53 → 107 → 195 → 8 → 49 → 125 → 47 → 175 → 9 → 115 → 19 → 1
route 5: 1 → 106 → 150 → 180 → 5 → 156 → 26 → 56 → 166 → 131 → 55 → 196 → 1
route 6: 1 → 124 → 20 → 108 → 176 → 12 → 65 → 50 → 144 → 37 → 48 → 169 → 83 → 154 → 1
route 7: 1 → 112 → 77 → 197 → 117 → 69 → 81 → 151 → 13 → 139 → 155 → 1
route 8: 1 → 191 → 32 → 190 → 11 → 160 → 63 → 149 → 183 → 89 → 147 → 1
route 9: 1 → 113 → 96 → 93 → 152 → 99 → 38 → 194 → 86 → 94 → 100 → 105 → 97 → 7 → 1
route 10: 1 → 103 → 10 → 136 → 36 → 137 → 66 → 67 → 72 → 162 → 104 → 52 → 1
route 11: 1 → 167 → 84 → 200 → 126 → 46 → 18 → 114 → 62 → 174 → 85 → 6 → 119 → 61 → 1
route 12: 1 → 118 → 101 → 193 → 15 → 120 → 45 → 39 → 141 → 87 → 17 → 142 → 192 → 92 → 1
route 13: 1 → 138 → 3 → 116 → 179 → 58 → 16 → 44 → 143 → 43 → 173 → 145 → 88 → 98 → 60 → 1
route 14: 1 → 199 → 198 → 57 → 187 → 24 → 68 → 171 → 40 → 188 → 140 → 111 → 1
route 15: 1 → 28 → 70 → 133 → 177 → 2 → 123 → 71 → 102 → 163 → 128 → 168 → 1
route 16: 1 → 181 → 22 → 73 → 75 → 76 → 134 → 23 → 42 → 146 → 172 → 74 → 41 → 1
route 17: 1 → 29 → 185 → 78 → 159 → 4 → 122 → 30 → 25 → 164 → 135 → 178 → 110 → 27 → 1
total cost: 1337.00
```

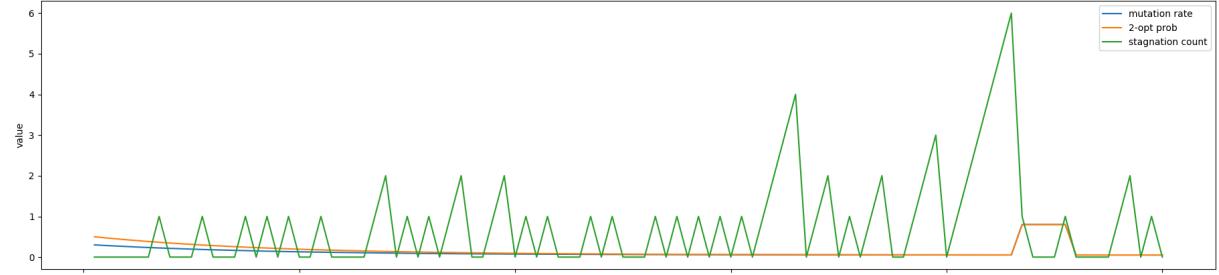
time: 5323.93s



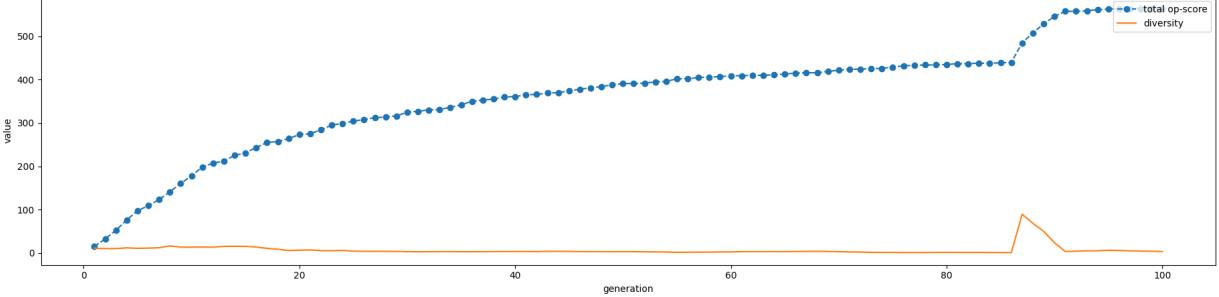
GA - M-n200-k17.vrp



GA Parameters per Generation



Operator Score and Diversity



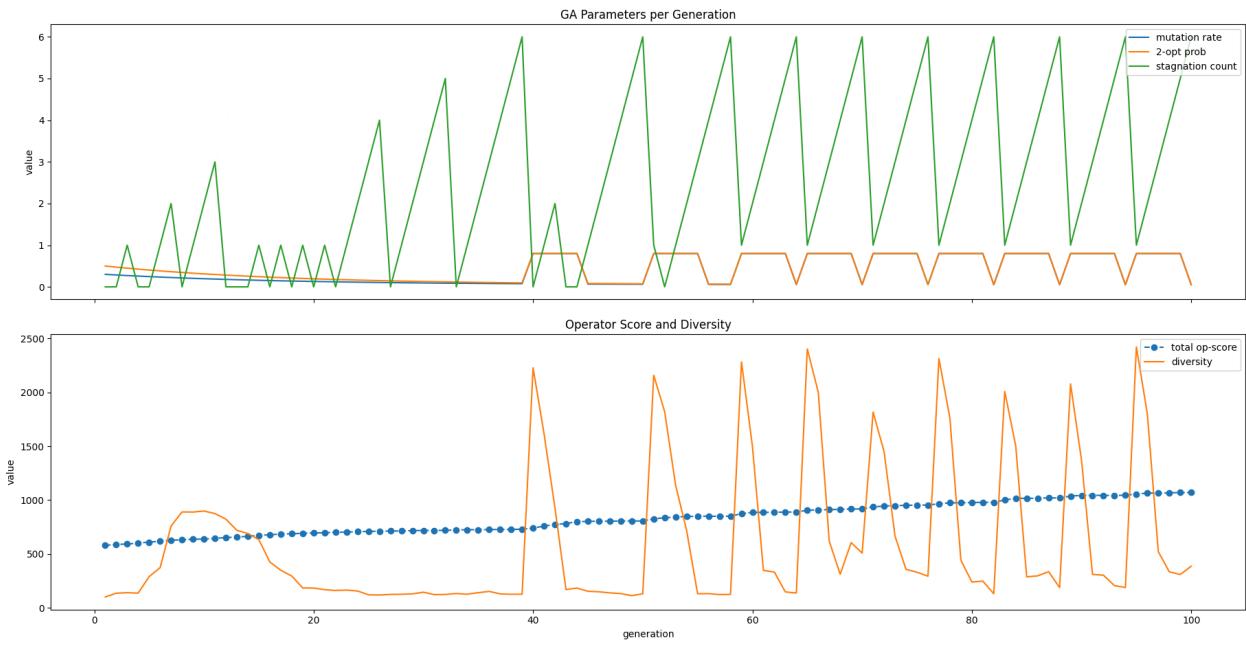
X-n101-k26

```
[INFO] ===== problem X-n101-k26.vrp =====
[INFO] parsed 'X-n101-k26.vrp' → cap=206, nodes=101, depot=1, minVeh=None
[INFO] reference .sol found - 26 routes, cost=27591.0
[INFO] --- GA ---

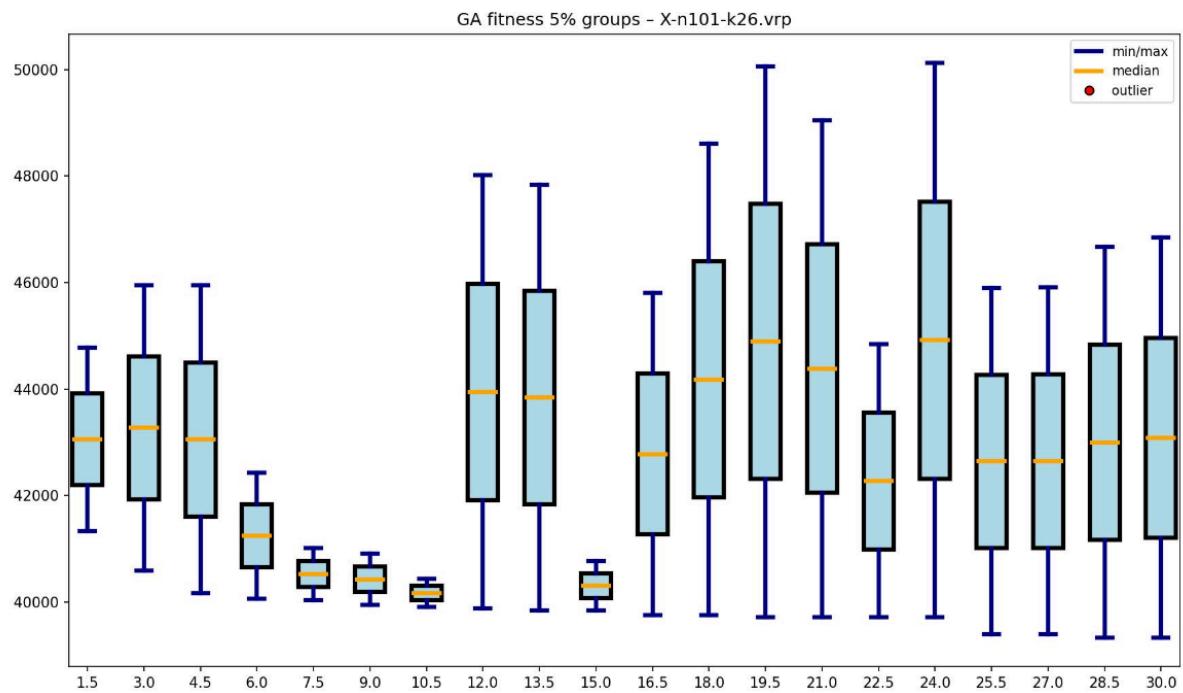
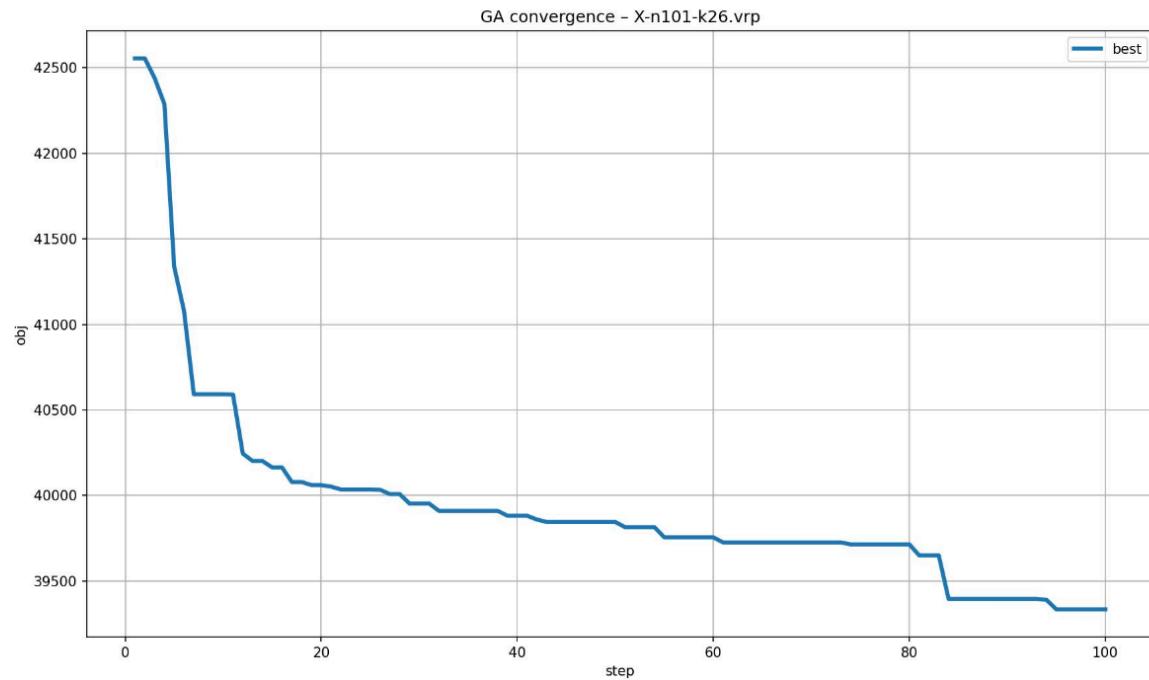
Starting GA with 8 islands, total 50 per island, 100 gens, 0.3 base mutation rate, scx crossover

[ga] | gen 01/100 | mut=0.30 | twoopt=0.50 | hyper=no | elapsed time = 0.00s
[ga] | gen 02/100 | mut=0.29 | twoopt=0.47 | hyper=no | elapsed time = 3.73s
[ga] | gen 03/100 | mut=0.27 | twoopt=0.45 | hyper=no | elapsed time = 7.71s
[ga] | gen 04/100 | mut=0.26 | twoopt=0.43 | hyper=no | elapsed time = 11.23s
[ga] | gen 05/100 | mut=0.25 | twoopt=0.40 | hyper=no | elapsed time = 14.72s
[ga] | gen 06/100 | mut=0.24 | twoopt=0.38 | hyper=no | elapsed time = 18.93s

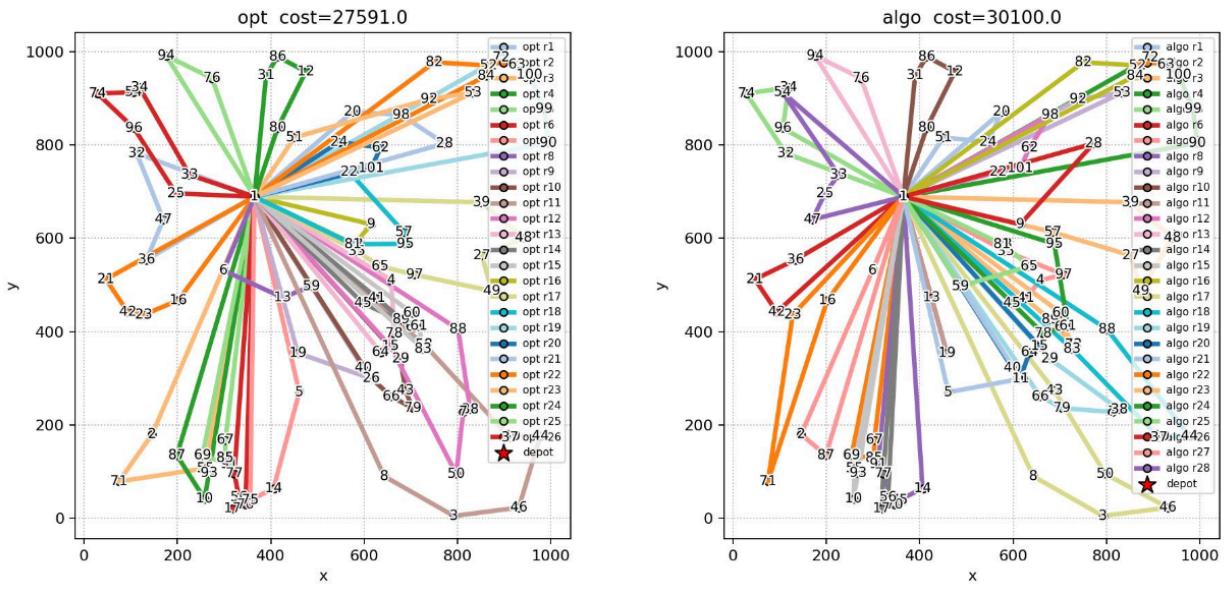
[ga] | gen 99/100 | mut=0.80 | twoopt=0.80 | hyper=no | elapsed time = 575.74s
[ga] | gen 100/100 | mut=0.05 | twoopt=0.05 | hyper=no | elapsed time = 579.36s
[INFO] GA runtime: 762.91s
[RESULT] solution:
route 1: 1 → 51 → 24 → 20 → 1
route 2: 1 → 67 → 85 → 69 → 1
route 3: 1 → 89 → 58 → 73 → 68 → 60 → 1
route 4: 1 → 72 → 63 → 100 → 99 → 90 → 1
route 5: 1 → 32 → 74 → 34 → 96 → 1
route 6: 1 → 36 → 21 → 42 → 1
route 7: 1 → 35 → 97 → 4 → 41 → 45 → 1
route 8: 1 → 14 → 75 → 70 → 91 → 1
route 9: 1 → 53 → 92 → 1
route 10: 1 → 80 → 12 → 86 → 31 → 1
route 11: 1 → 19 → 13 → 1
route 12: 1 → 22 → 101 → 62 → 98 → 1
route 13: 1 → 76 → 94 → 1
route 14: 1 → 56 → 17 → 77 → 1
route 15: 1 → 55 → 10 → 93 → 1
route 16: 1 → 82 → 52 → 84 → 1
route 17: 1 → 43 → 50 → 46 → 3 → 8 → 1
route 18: 1 → 88 → 44 → 30 → 37 → 83 → 1
route 19: 1 → 66 → 79 → 7 → 38 → 29 → 1
route 20: 1 → 40 → 26 → 64 → 15 → 1
route 21: 1 → 5 → 11 → 1
route 22: 1 → 16 → 71 → 23 → 1
route 23: 1 → 57 → 27 → 49 → 48 → 39 → 1
route 24: 1 → 95 → 61 → 78 → 1
route 25: 1 → 59 → 65 → 18 → 81 → 1
route 26: 1 → 9 → 28 → 1
route 27: 1 → 6 → 2 → 87 → 1
route 28: 1 → 47 → 25 → 33 → 54 → 1
total cost: 30100.00
```



time: 762.91s



GA - X-n101-k26.vrp



מסקנות ודיון על התוצאות:

בהתבסס על כלל התוצאות שהתקבלו בהרצחת האלגוריתם הגנטי עם מודל האיים (GA with Islands) על בעיות CVRP שונות, ניתן להסיק מספר מסקנות מהותיות לגבי אופי ההתנהגות של האלגוריתם, רגישותו לפרמטרים, והאופן שבו הוא מתמודד עם בעיות בקנה מידה שונה.

שונות בין בעיות פשוטות למורכבות

הפער בין הביצועים בעיות פשוטות (כמו k4-k22-h-E) לעומת בעיות מורכבות (כמו 63-opt26-k5-X-n101-k26-A) מצביע על כך שהאלגוריתם במצבו הנוכחי מותאם יותר לעובות קטנות או בעלות מבנה גאומטרי נוח. ככל שהמבנה גדול או שהמבנה נעשה לא אחיד (לדוגמה: פיזור מריחבי לא אחד, דרישות קיבולת צפופות), איקוט הפתרונות יורדת בצורה ברורה.

התוכנסות מהירה אך לא יציבה

במספר בעיות נצפתה התוכנסות מוקדמת, עם ירידה חדה בעולות בדורות הראשונים – תופעה שבולטת בעיקר בגרפים של k10-k80-A-6-k26-X. עם זאת, בשלבים המאוחרים האלגוריתם כמעט אינו משיפור, ולעיתים אף חוזה חזרה לסטיות גבוהות באוכליותיה. ההתנהגות זו מעידה על כך שהאוכלוסיות באים נוטות להתכנס מהר מדי לפתרונות לוקאלים, ומתקשות להיחלץ מהם בהמשך.

שונות באוכלוסיה לאורך הריצה

בניטוח תיבות הטווח (dataplots), ניכרת שונות גובהה בשלבים המוקדמים שמתיצבת רק לעיתים, אך בחלק מהמקרים מתחדשת באופן חריג בשלבים המאוחרים. לדוגמה, ב-k200-h-17-k2 מרשמה קפיצה פתואמית ברוחב הטווחים דואקן לקראת סוף הריצה. זה מעיד על קפיצות חדות בהתנהגות האוכלוסיה – תוצאה של שינויים פתאומיים בפרמטרים כמו שינוי המוטציה או התנועת היפר-פרמטרים אגרסיביים. הדבר מצביע על כך שאין איזון דינמי מספק בין חקירה לניצול לאורך כל הריצה.

תגובה האלגוריתם למוגבלות קיבולת

בעיות בעלות מגבלות הדוקות (כמו k5-h-32-k5-A) הביאו לתוצאות פחות טובות, גם כמשמעות בעיות קטנות יחסית. התוצאה הסופית הייתה רוחקה מהפתרון הידוע, למרות זמן הריצה היה ממושך ואוכלוסיה גדולה הופעלה. הדבר מדגיש שהאלגוריתם, במצבו הנוכחי, מתקשה לפעול ביעילות בסביבות עם מגבלות חזקות, ואין מצליח לייצר אוכלוסיה שמבינה או "שומרת" על תקופות לאורך הדורות.

התנהגות מודל האיים

בעיות גדולות (למשל k200-h-101-k26, M-h-17-k2-X) מדגימות היטב את הפער בין האיים: כל אי מתקדם בקצב ובכיוון שונה, וההשפעה ההידידית ביניהם אינה ברורה או אפקטיבית. לאורך הריצות לא נרשמו קפיצות איקוט פתאומיות שהיו עשויות להעיד על שיתוף מוצלח של מידע איקוטי בין איים. למעשה, נראה שכלי נסגר סיבוב איזור מסוים במרחב הפתורנות, מה שמעיד על קיטוע בפעולה של המודל הרב-אי.

חוסר התאמה בין מבנה הפרמטרים לאורכי הבעיה

העובדת של הריצות שמשו באופן ערכי בסיס לפרמטרים המרכזיים – מספר דורות, גודל אוכלוסיה לאי, שיעור מוטציה, פרמטרי opt-opt-2 – יקרה חוסר גמישות. במקרים של בעיות פשוטות הפרמטרים הללו היי מספקים, אך בעיות גדולות הם התגלו כמוגבלים: או שהם לא הספיקו בכמותם, או שהיו אגרסיביים מדי. התוצאה היא שהאלגוריתם מגיב בצורה שונה מאוד לאותו סט פרמטרים, בהתאם למבנה הבעיה.

סיכום

התנהגות האלגוריתם כפי שנצפתה בתוצאות מצביעה על כך שהוא מימוש ביעילות את הפוטנציאלי של חישוב אבולוציוני על בעיות קטנות ופשוטות, אך סובל מחוסר עקביות כאשר קנה המידה של הבעיה עולה. השונות בין האיים, הקונברגנס המהיר, והתגובה החדה לשינויים בפרמטרים – כל אלו מדגימים שהאלגוריתם פועל ללא יסודות עדין לאורך זמן וללא התאמה מספקת לבנייה הבעיה הספציפי. בכך, התוצאות חשופות את הגבולות של המבנה הקים ואת הריגשות הגבוהה של האלגוריתם לתנאי הריצה.

Adaptive Large Neighborhood Search

מעבר הביעות בקטגורית **Beginner** •

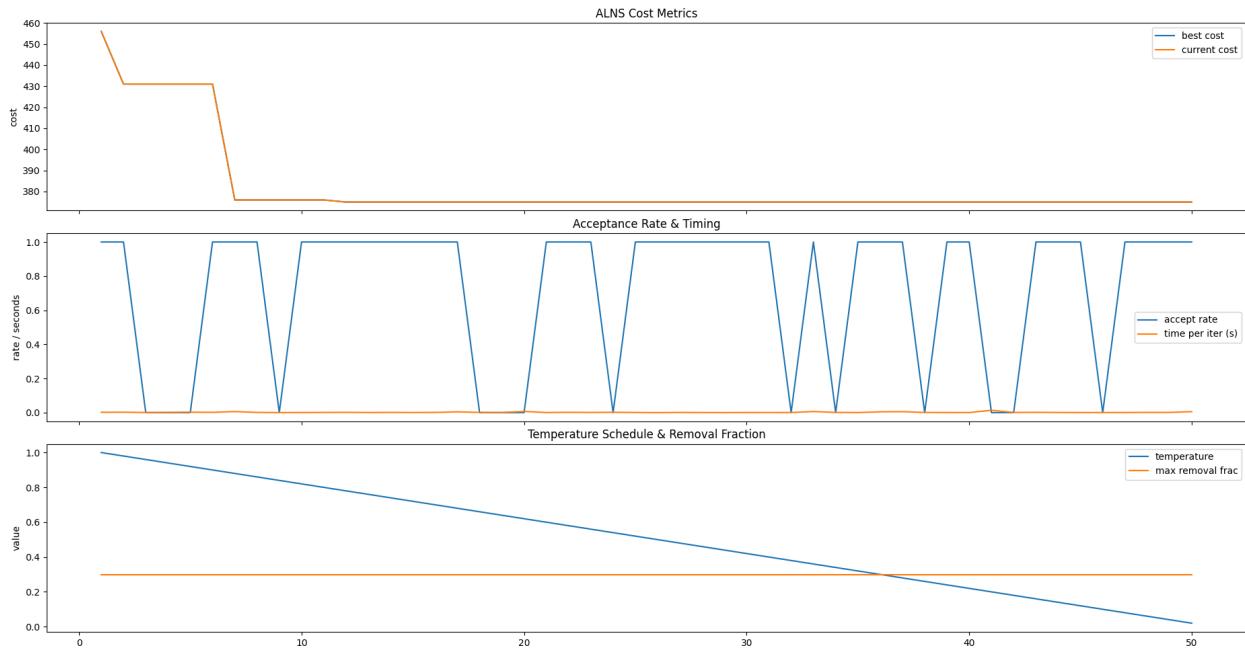
- נתנו לכל בעיה לrhoץ למשך זמן מאד קצר, לכל היותר 60 שניות (לכל בעיה)
- הפרמטרים מעבור ריצה זו:

- Number of iterations = 50
- Fraction removal bounds $\in [0.1, 0.3]$
- Weight updates interval = 100
- K value for KNN = 10
- Regret K = 2
- Temperature change (epsilon) = 10^{-6}
- Penalty for exceeding vehicle capacity = 10^6
- Stagnation threshold = 50

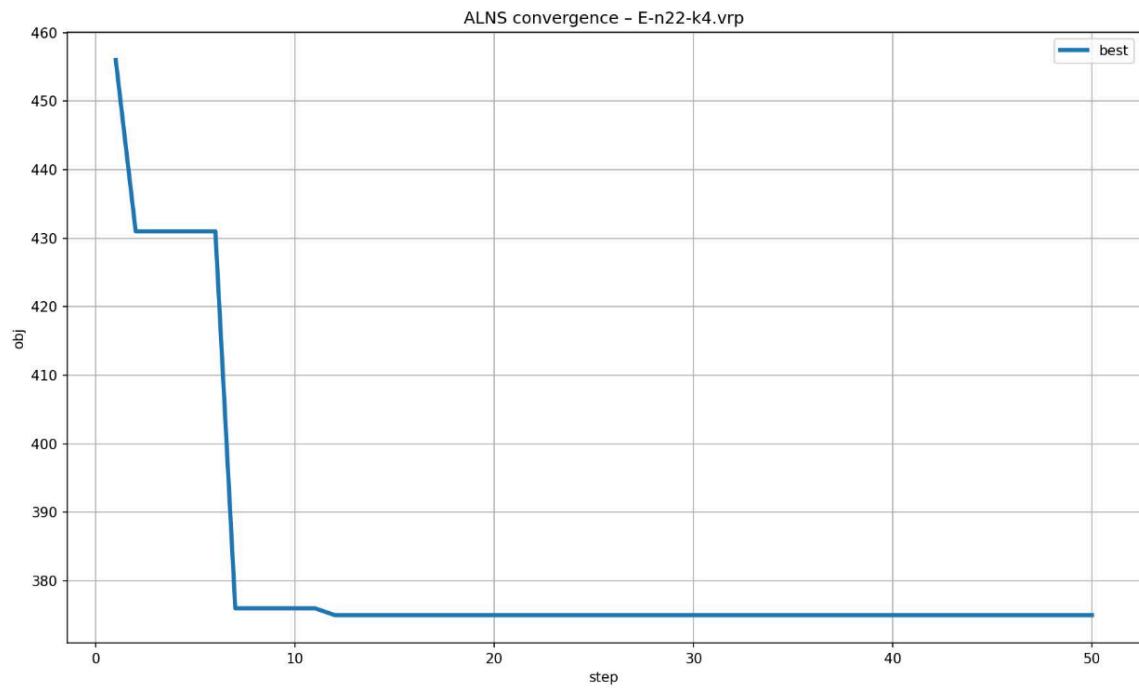
להלן התוצאות:

E-n22-k4

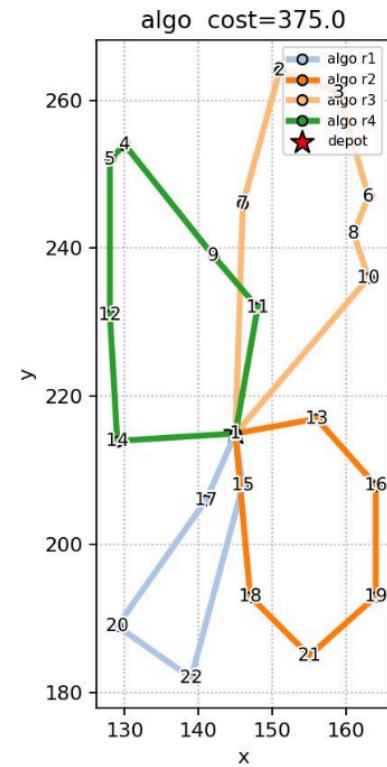
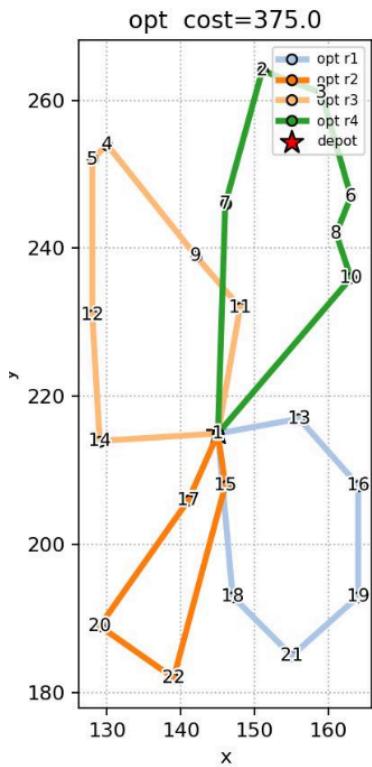
```
[INFO] ===== problem E-n22-k4.vrp =====  
[INFO] parsed 'E-n22-k4.vrp' → cap=6000, nodes=22, depot=1, minVeh=4  
[INFO] reference .sol found - 4 routes, cost=375.0  
[INFO] --- ALNS ---  
  
Starting ALNS with 50 iterations, with exploration of 10 neighbors per iteration.  
  
[ALNS] iteration = 1/50 | best = 431.00 | accepted = True | time = 0.01s  
[ALNS] iteration = 2/50 | best = 431.00 | accepted = True | time = 0.01s  
[ALNS] iteration = 3/50 | best = 431.00 | accepted = True | time = 0.01s  
[ALNS] iteration = 4/50 | best = 429.00 | accepted = True | time = 0.01s  
[ALNS] iteration = 5/50 | best = 425.00 | accepted = True | time = 0.02s  
[ALNS] iteration = 6/50 | best = 425.00 | accepted = True | time = 0.02s  
[ALNS] iteration = 7/50 | best = 376.00 | accepted = True | time = 0.03s  
  
[ALNS] iteration = 48/50 | best = 375.00 | accepted = False | time = 0.11s  
[ALNS] iteration = 49/50 | best = 375.00 | accepted = True | time = 0.11s  
[ALNS] iteration = 50/50 | best = 375.00 | accepted = True | time = 0.12s  
[INFO] ALNS runtime: 7.73s  
[RESULT] solution:  
route 1: 1 → 17 → 20 → 22 → 15 → 1  
route 2: 1 → 13 → 16 → 19 → 21 → 18 → 1  
route 3: 1 → 10 → 8 → 6 → 3 → 2 → 7 → 1  
route 4: 1 → 11 → 9 → 4 → 5 → 12 → 14 → 1  
total cost: 375.00
```



time: 29.09s



ALNS - E-n22-k4.vrp



P-n16-k8

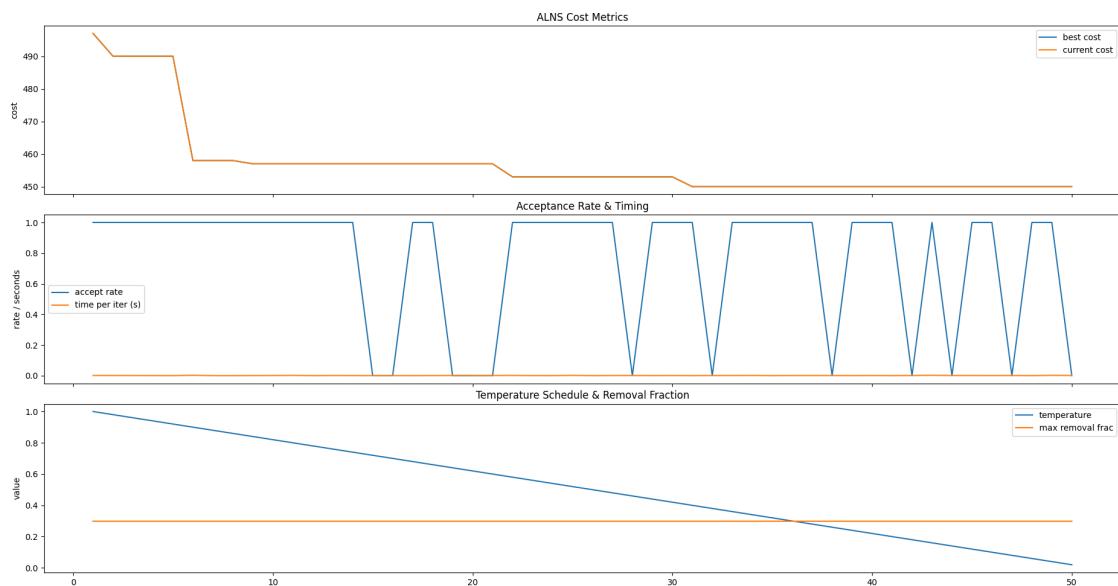
```
[INFO] ===== problem P-n16-k8.vrp =====
[INFO] parsed 'P-n16-k8.vrp' → cap=35, nodes=16, depot=1, minVeh=None
[INFO] reference .sol found - 8 routes, cost=450.0
[INFO] --- ALNS ---

Starting ALNS with 50 iterations, with exploration of 10 neighbors per iteration.

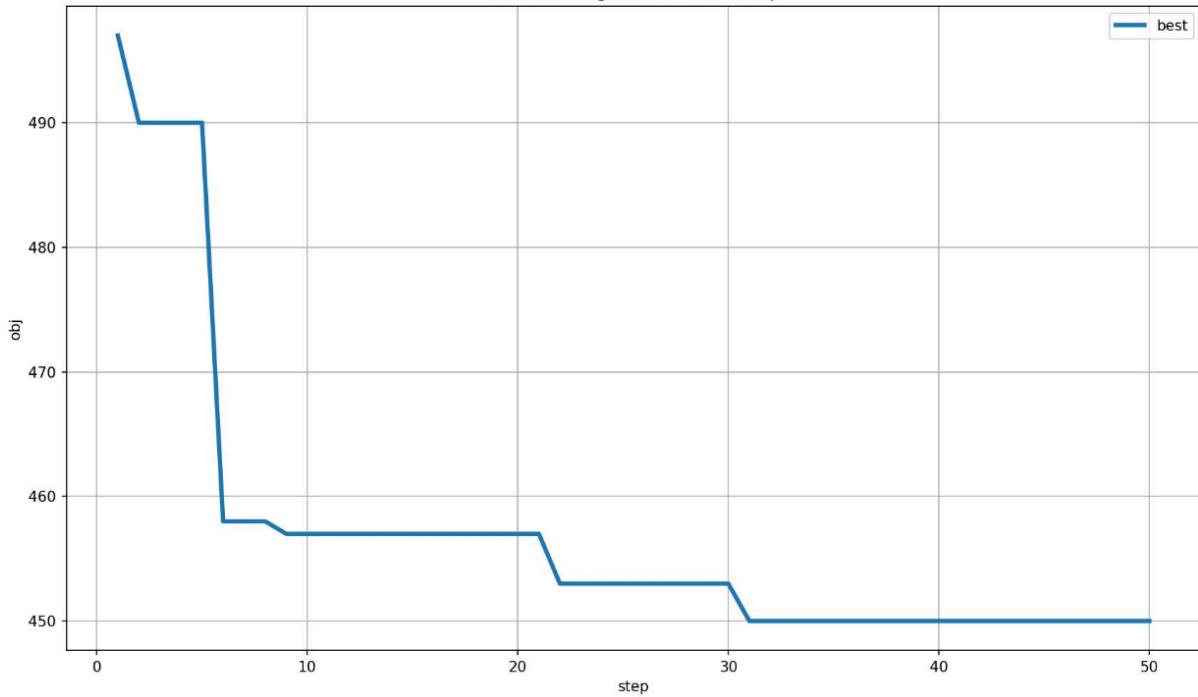
[ALNS] iteration = 1/50 | best = 497.00 | accepted = True | time = 0.00s
[ALNS] iteration = 2/50 | best = 490.00 | accepted = True | time = 0.00s
[ALNS] iteration = 3/50 | best = 490.00 | accepted = True | time = 0.00s
[ALNS] iteration = 4/50 | best = 490.00 | accepted = True | time = 0.00s
[ALNS] iteration = 5/50 | best = 490.00 | accepted = True | time = 0.00s
[ALNS] iteration = 6/50 | best = 458.00 | accepted = True | time = 0.01s

[ALNS] iteration = 48/50 | best = 450.00 | accepted = True | time = 0.04s
[ALNS] iteration = 49/50 | best = 450.00 | accepted = True | time = 0.05s
[ALNS] iteration = 50/50 | best = 450.00 | accepted = False | time = 0.05s
[INFO] ALNS runtime: 72.23s

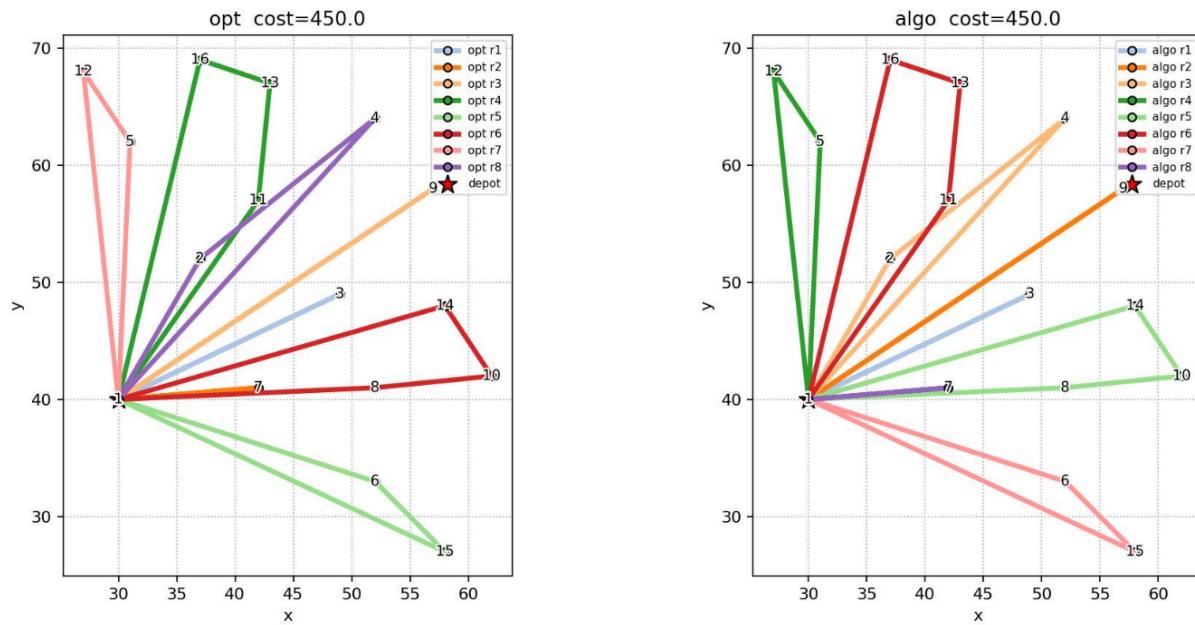
[RESULT] solution:
route 1: 1 → 3 → 1
route 2: 1 → 9 → 1
route 3: 1 → 2 → 4 → 1
route 4: 1 → 5 → 12 → 1
route 5: 1 → 8 → 10 → 14 → 1
route 6: 1 → 11 → 13 → 16 → 1
route 7: 1 → 6 → 15 → 1
route 8: 1 → 7 → 1
total cost: 450.00
```



ALNS convergence - P-n16-k8.vrp



ALNS - P-n16-k8.vrp



• עבור הביעות בקטגורית Intermediate

- נתנו לכל בעיה לרווח זמן באורך ביןוני, לכל היותר 2000 איטרציות (לכל בעיה)
- הפרמטרים עבור ריצה זו:

- Number of iterations = 2000
- Fraction removal bounds $\in [0.1, 0.3]$
- Weight updates interval = 100
- K value for KNN = 10
- Regret K = 2
- Temperature change (epsilon) = 10^{-6}
- Penalty for exceeding vehicle capacity = 10^6
- Stagnation threshold = 50

להלן התוצאות:

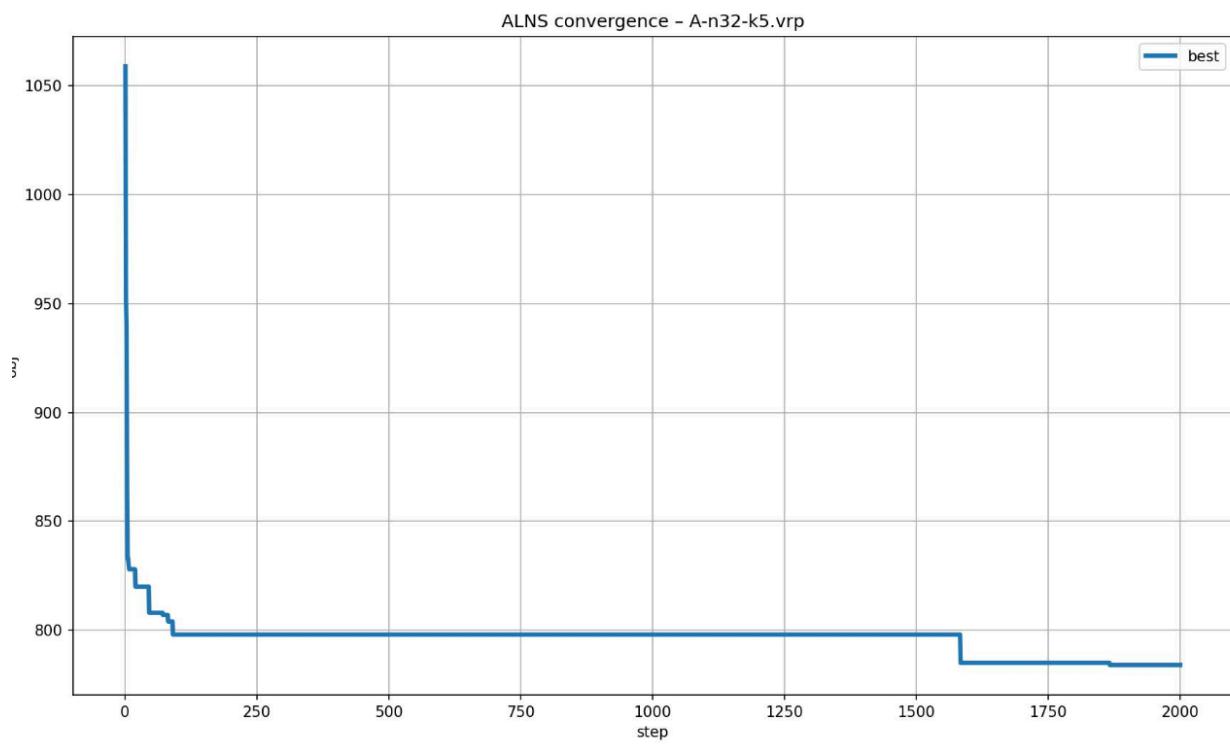
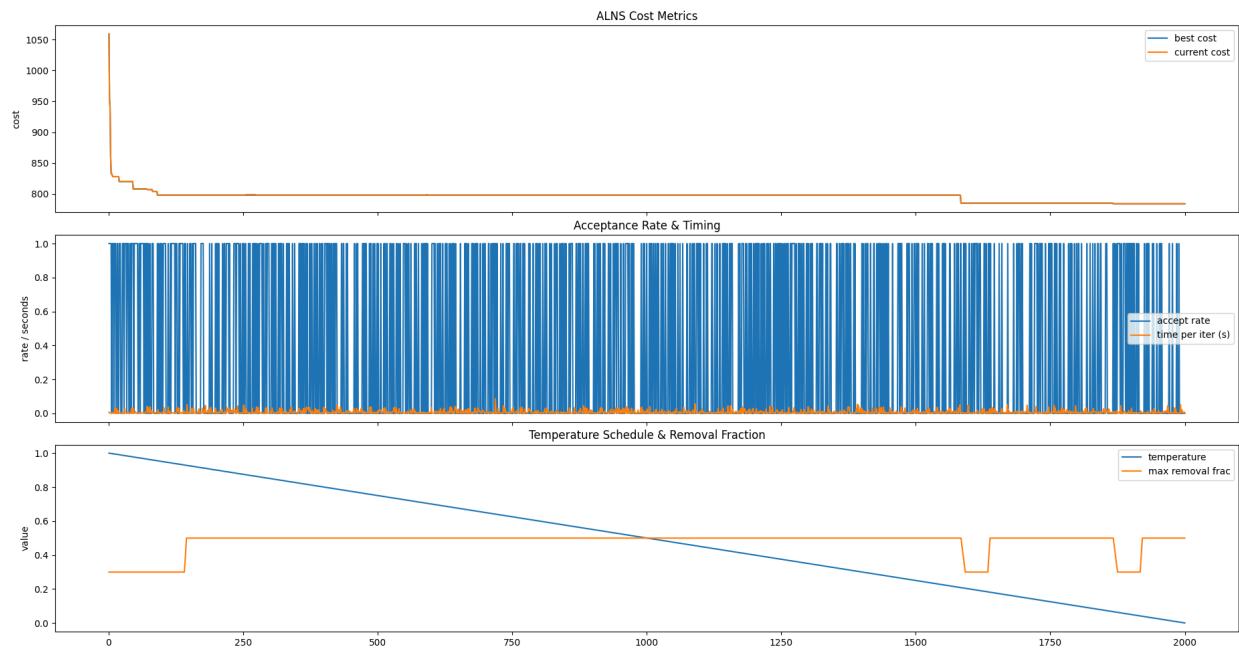
A-n32-k5

```
[ALNS] iteration = 1998/2000 | best = 784.00 | accepted = False | time = 15.90s
[ALNS] iteration = 1999/2000 | best = 784.00 | accepted = False | time = 15.91s
[ALNS] iteration = 2000/2000 | best = 784.00 | accepted = False | time = 15.91s
    [INFO] ALNS runtime: 63.51s
[RESULT] solution:
route 1: 1 → 31 → 17 → 2 → 13 → 1
route 2: 1 → 25 → 28 → 1
route 3: 1 → 22 → 32 → 20 → 18 → 14 → 8 → 27 → 1
route 4: 1 → 15 → 29 → 12 → 5 → 24 → 3 → 4 → 7 → 1
route 5: 1 → 21 → 6 → 26 → 11 → 16 → 23 → 10 → 9 → 19 → 30 → 1
total cost: 784.00
```

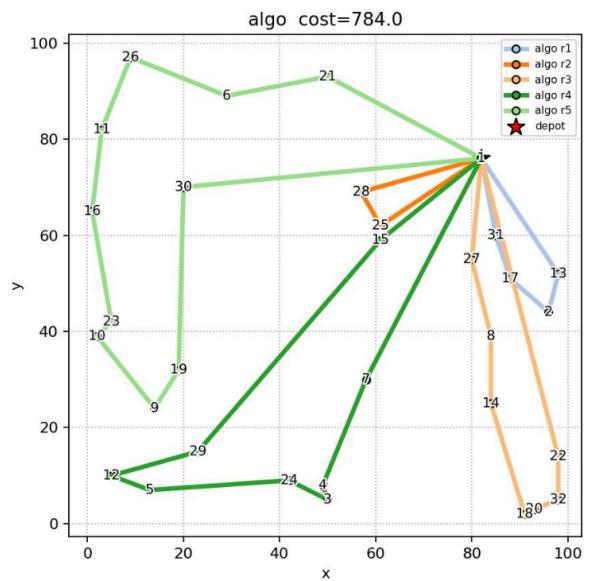
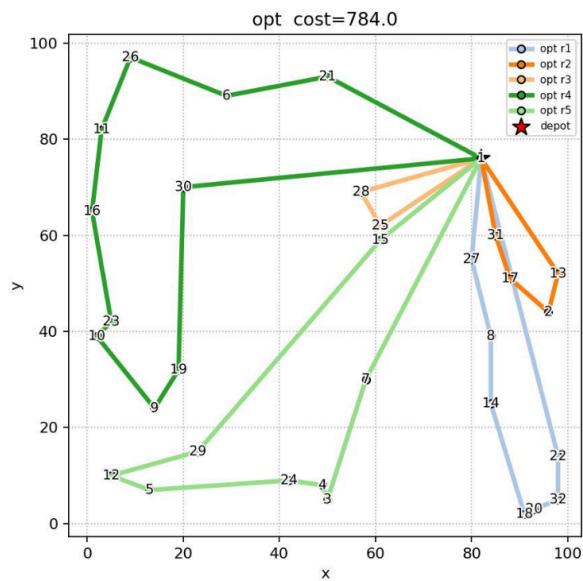
```
[INFO] ===== problem A-n32-k5.vrp =====
[INFO] parsed 'A-n32-k5.vrp' → cap=100, nodes=32, depot=1, minVeh=None
[INFO] reference .sol found - 5 routes, cost=784.0
[INFO] --- ALNS ---
```

Starting ALNS with 2000 iterations, with exploration of 10 neighbors per iteration.

```
[ALNS] iteration = 1/2000 | best = 1059.00 | accepted = True | time = 0.01s
[ALNS] iteration = 2/2000 | best = 952.00 | accepted = True | time = 0.01s
[ALNS] iteration = 3/2000 | best = 940.00 | accepted = True | time = 0.01s
[ALNS] iteration = 4/2000 | best = 862.00 | accepted = True | time = 0.02s
[ALNS] iteration = 5/2000 | best = 833.00 | accepted = True | time = 0.02s
[ALNS] iteration = 6/2000 | best = 833.00 | accepted = False | time = 0.03s
```



ALNS - A-n32-k5.vrp

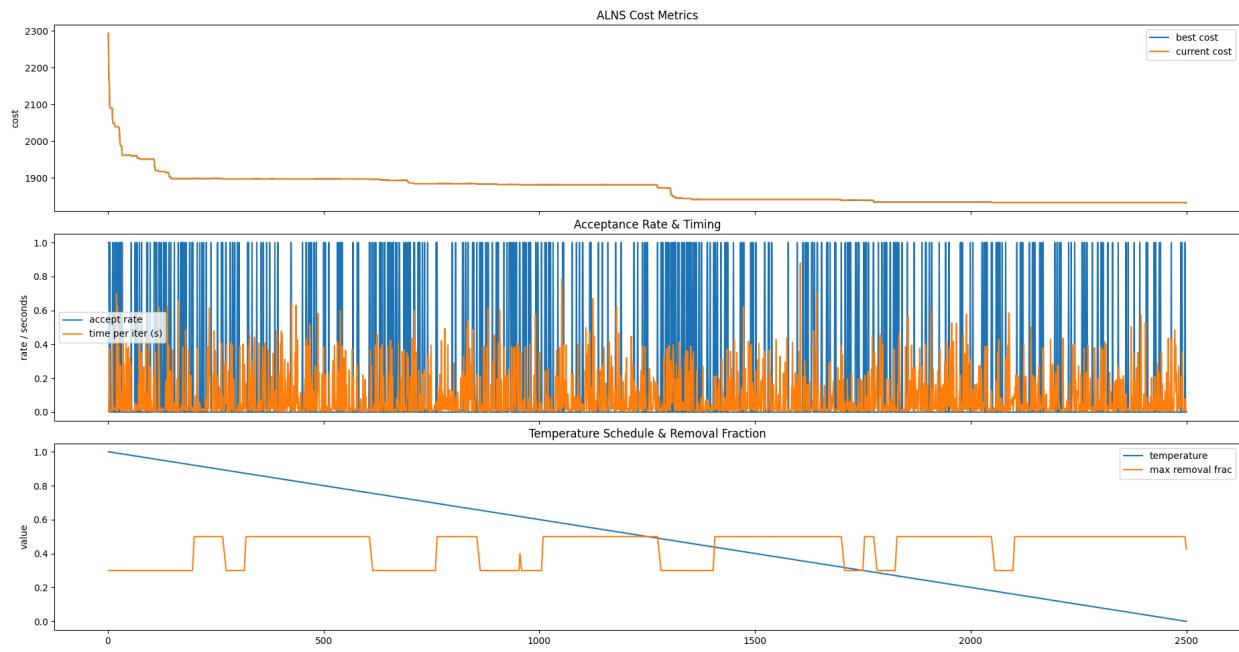


A-n80-k10

הערה:

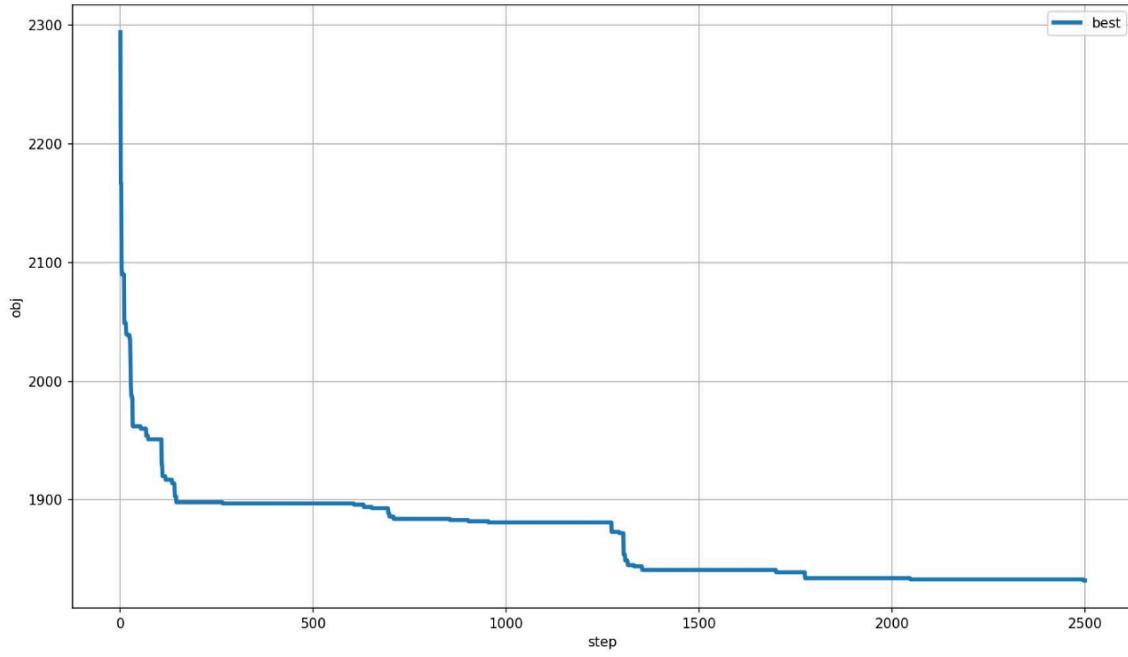
- מכיוון שבבניה זו ישנו הרבה פתרונות גדל משמעותי משמעותית, כדי להיחשף ליותר אפשרויות, בחרנו להגדיל עבורה בעיה זו את ערך ה-KNN Neighbors ל-50, כך שכל פתרון שנתקן ייחשף להרבה יותר אפשרויות מאשר בהרצה הראשונית, ובכך עשוי למצוא תיקון שמשפר משמעותית את ערך ה-cost.
- כדי לתת לאלגוריתם יותר זמן להגעה לתוצאות טובות, הגדלנו את מספר האיטרציות ל-2500.

```
[INFO] ===== problem A-n80-k10.vrp =====  
[INFO] parsed 'A-n80-k10.vrp' → cap=100, nodes=80, depot=1, minVeh=None  
[INFO] reference .sol found - 10 routes, cost=1763.0  
[INFO] --- ALNS ---  
  
Starting ALNS with 2500 iterations, with exploration of 50 neighbors per iteration.  
  
[ALNS] iteration = 1/2500 | best = 2294.00 | accepted = True | time = 0.00s  
[ALNS] iteration = 2/2500 | best = 2167.00 | accepted = True | time = 0.02s  
[ALNS] iteration = 3/2500 | best = 2167.00 | accepted = False | time = 0.39s  
[ALNS] iteration = 4/2500 | best = 2095.00 | accepted = True | time = 0.42s  
[ALNS] iteration = 5/2500 | best = 2090.00 | accepted = True | time = 0.43s  
  
[ALNS] iteration = 2498/2500 | best = 1832.00 | accepted = False | time = 222.54s  
[ALNS] iteration = 2499/2500 | best = 1832.00 | accepted = False | time = 222.55s  
[ALNS] iteration = 2500/2500 | best = 1832.00 | accepted = False | time = 222.56s  
[INFO] ALNS runtime: 240.91s  
[RESULT] solution:  
route 1: 1 → 68 → 71 → 73 → 55 → 10 → 56 → 57 → 70 → 66 → 36 → 27 → 76 → 21 → 28 → 1  
route 2: 1 → 2 → 8 → 22 → 41 → 1  
route 3: 1 → 77 → 51 → 46 → 23 → 5 → 33 → 59 → 1  
route 4: 1 → 14 → 4 → 75 → 30 → 45 → 63 → 1  
route 5: 1 → 52 → 78 → 40 → 61 → 47 → 42 → 26 → 32 → 18 → 1  
route 6: 1 → 64 → 12 → 25 → 7 → 24 → 13 → 1  
route 7: 1 → 43 → 37 → 67 → 39 → 74 → 50 → 1  
route 8: 1 → 72 → 53 → 29 → 80 → 19 → 49 → 15 → 1  
route 9: 1 → 31 → 79 → 62 → 69 → 9 → 38 → 3 → 35 → 11 → 1  
route 10: 1 → 6 → 60 → 17 → 44 → 58 → 20 → 48 → 16 → 34 → 65 → 54 → 1  
total cost: 1832.00
```

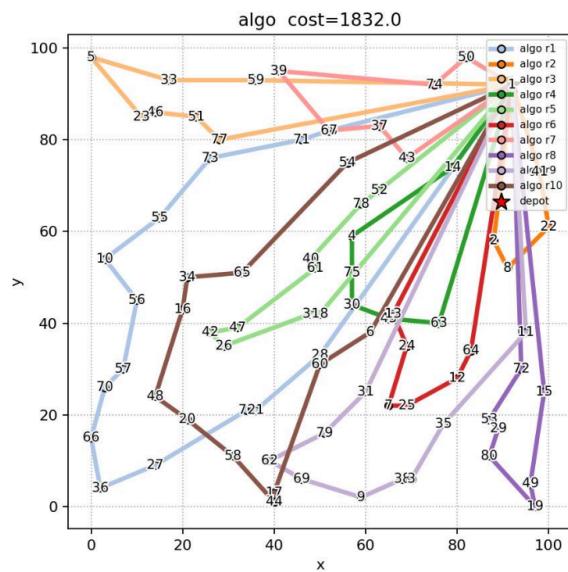
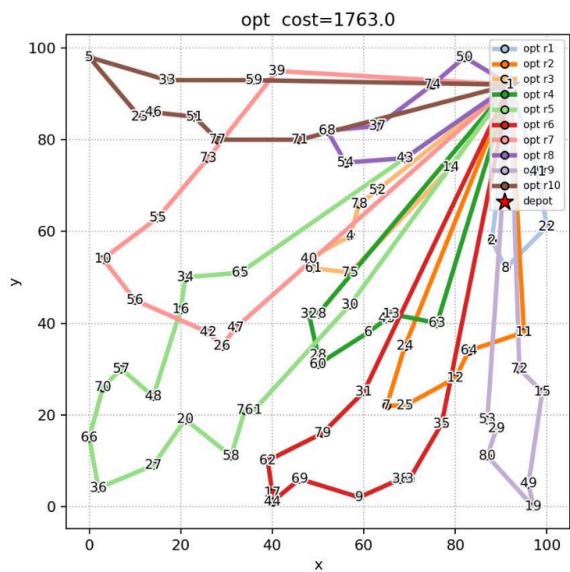


time: 240.91s

ALNS convergence - A-n80-k10.vrp



ALNS - A-n80-k10.vrp



• עבור הביעות בקטגוריות Advanced

- נתנו לכל בעיה לרווח זמן יחסית ארוך, כ-3000 איטרציות (לכל בעיה)
- הפרמטרים עבור ריצה זו:

- Number of iterations = 3000
- Fraction removal bounds $\in [0.1, 0.3]$
- Weight updates interval = 100
- K value for KNN = 50
- Regret K = 2
- Temperature change (epsilon) = 10^{-6}
- Penalty for exceeding vehicle capacity = 10^6
- Stagnation threshold = 50

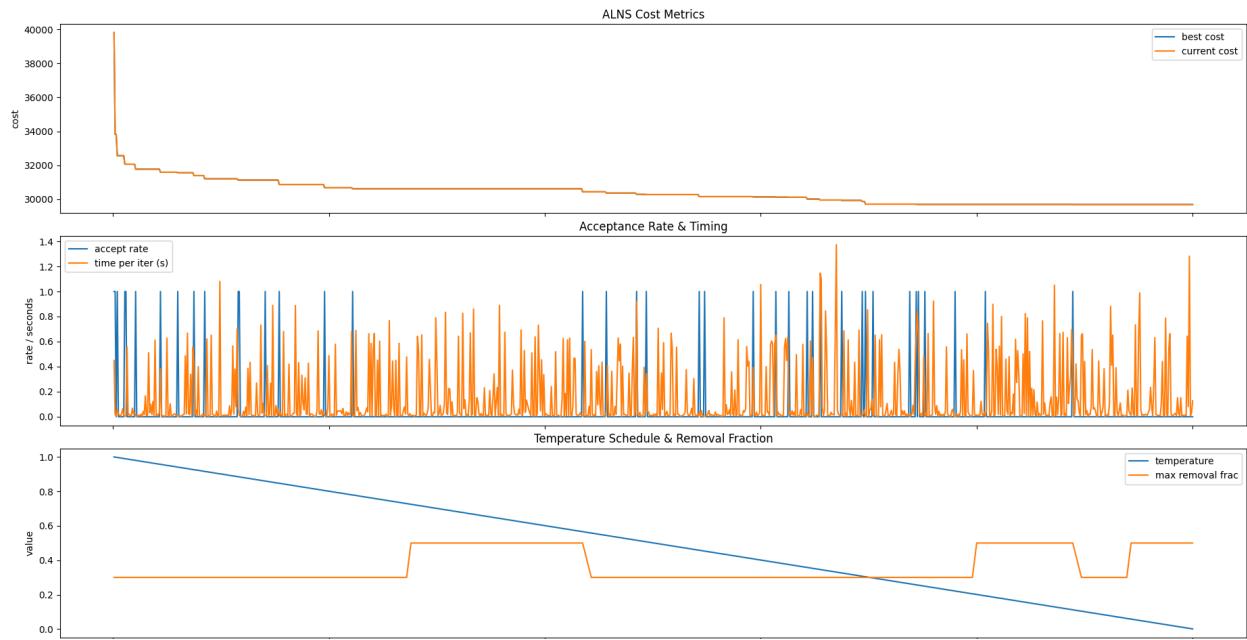
M-n200-k17

```
[INFO] ===== problem M-n200-k17.vrp =====
[INFO] parsed 'M-n200-k17.vrp' → cap=200, nodes=200, depot=1, minVeh=None
[INFO] reference .sol found - 17 routes, cost=1275.0
[INFO] --- ALNS ---

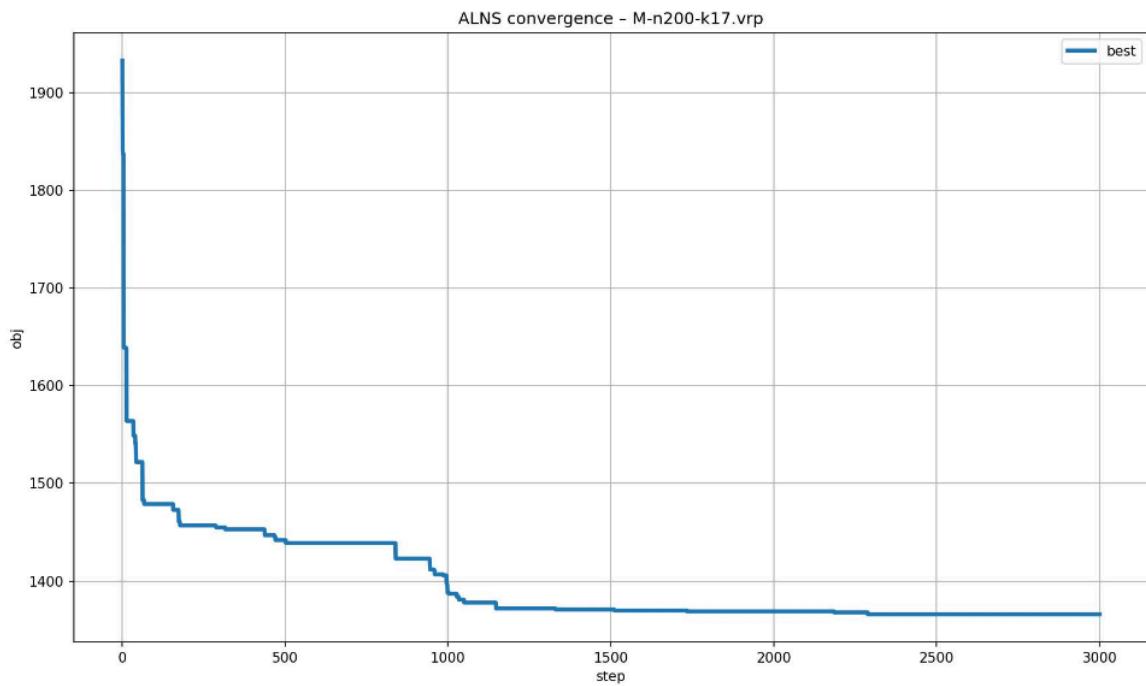
Starting ALNS with 3000 iterations, with exploration of 50 neighbors per iteration.

[ALNS] iteration = 1/3000 | best = 1989.00 | accepted = False | time = 0.77s
[ALNS] iteration = 2/3000 | best = 1924.00 | accepted = True | time = 7.98s
[ALNS] iteration = 3/3000 | best = 1924.00 | accepted = False | time = 8.63s
[ALNS] iteration = 4/3000 | best = 1857.00 | accepted = True | time = 14.10s
[ALNS] iteration = 5/3000 | best = 1857.00 | accepted = False | time = 14.42s

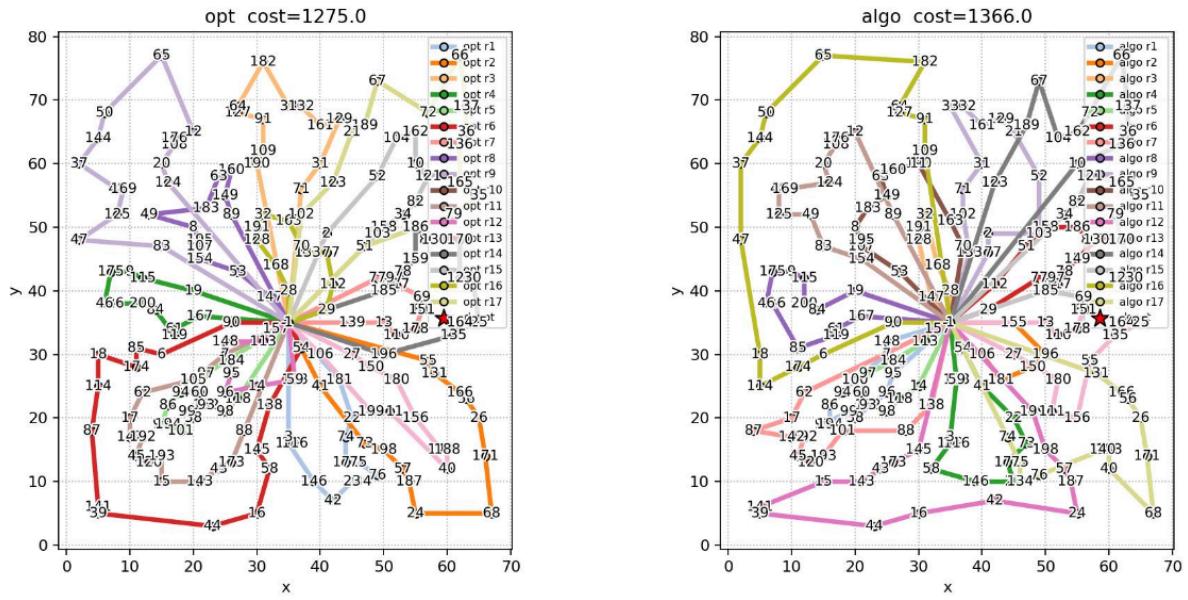
[ALNS] iteration = 2997/3000 | best = 1366.00 | accepted = False | time = 2342.58s
[ALNS] iteration = 2998/3000 | best = 1366.00 | accepted = True | time = 2342.81s
[ALNS] iteration = 2999/3000 | best = 1366.00 | accepted = False | time = 2342.83s
[ALNS] iteration = 3000/3000 | best = 1366.00 | accepted = False | time = 2342.88s
[INFO] ALNS runtime: 2342.93s
[RESULT] solution:
route 1: 1 → 95 → 96 → 38 → 99 → 92 → 194 → 86 → 94 → 105 → 100 → 7 → 148 → 1
route 2: 1 → 54 → 106 → 181 → 150 → 196 → 139 → 155 → 1
route 3: 1 → 28 → 168 → 32 → 191 → 128 → 147 → 1
route 4: 1 → 41 → 22 → 73 → 172 → 23 → 146 → 58 → 179 → 116 → 3 → 153 → 59 → 1
route 5: 1 → 157 → 113 → 184 → 97 → 60 → 152 → 93 → 98 → 118 → 14 → 1
route 6: 1 → 112 → 51 → 158 → 186 → 80 → 130 → 159 → 4 → 78 → 117 → 197 → 77 → 29 → 1
route 7: 1 → 138 → 88 → 101 → 193 → 120 → 45 → 192 → 142 → 87 → 17 → 62 → 1
route 8: 1 → 167 → 61 → 119 → 85 → 126 → 46 → 175 → 9 → 115 → 200 → 84 → 19 → 1
route 9: 1 → 177 → 2 → 103 → 52 → 21 → 129 → 161 → 132 → 33 → 31 → 71 → 102 → 133 → 1
route 10: 1 → 53 → 107 → 195 → 8 → 183 → 149 → 63 → 160 → 190 → 11 → 163 → 70 → 1
route 11: 1 → 154 → 83 → 49 → 125 → 48 → 169 → 124 → 20 → 108 → 176 → 12 → 89 → 1
route 12: 1 → 145 → 173 → 43 → 143 → 15 → 141 → 39 → 44 → 16 → 42 → 24 → 187 → 57 → 198 → 199 → 1
route 13: 1 → 13 → 110 → 178 → 81 → 151 → 164 → 25 → 135 → 55 → 156 → 5 → 111 → 180 → 27 → 1
route 14: 1 → 123 → 189 → 67 → 104 → 162 → 72 → 66 → 137 → 36 → 136 → 10 → 1
route 15: 1 → 185 → 69 → 122 → 30 → 170 → 79 → 35 → 165 → 121 → 82 → 34 → 1
route 16: 1 → 90 → 6 → 174 → 114 → 18 → 47 → 37 → 144 → 50 → 65 → 182 → 64 → 127 → 91 → 109 → 1
route 17: 1 → 131 → 166 → 56 → 26 → 171 → 68 → 40 → 188 → 140 → 76 → 134 → 75 → 74 → 1
total cost: 1366.00
```



time: 2342.93s



ALNS - M-n200-k17.vrp



X-n101-k26

```
[INFO] ===== problem X-n101-k26.vrp =====  
[INFO] parsed 'X-n101-k26.vrp' → cap=206, nodes=101, depot=1, minVeh=None  
[INFO] reference .sol found - 26 routes, cost=27591.0  
[INFO] --- ALNS ---
```

Starting ALNS with 3000 iterations, with exploration of 50 neighbors per iteration.

```
[ALNS] iteration = 1/3000 | best = 39782.00 | accepted = True | time = 0.02s  
[ALNS] iteration = 2/3000 | best = 39782.00 | accepted = False | time = 0.60s  
[ALNS] iteration = 3/3000 | best = 39768.00 | accepted = True | time = 0.61s  
[ALNS] iteration = 4/3000 | best = 34161.00 | accepted = True | time = 0.64s  
[ALNS] iteration = 5/3000 | best = 33991.00 | accepted = True | time = 0.75s  
[ALNS] iteration = 6/3000 | best = 33991.00 | accepted = False | time = 0.77s  
[ALNS] iteration = 7/3000 | best = 33991.00 | accepted = False | time = 0.78s  
[ALNS] iteration = 8/3000 | best = 33174.00 | accepted = True | time = 0.79s
```

```
[ALNS] iteration = 2996/3000 | best = 28429.00 | accepted = False | time = 307.44s  
[ALNS] iteration = 2997/3000 | best = 28429.00 | accepted = False | time = 307.47s  
[ALNS] iteration = 2998/3000 | best = 28429.00 | accepted = False | time = 307.48s  
[ALNS] iteration = 2999/3000 | best = 28429.00 | accepted = False | time = 307.92s  
[ALNS] iteration = 3000/3000 | best = 28429.00 | accepted = False | time = 307.95s
```

[INFO] ALNS runtime: 307.95s

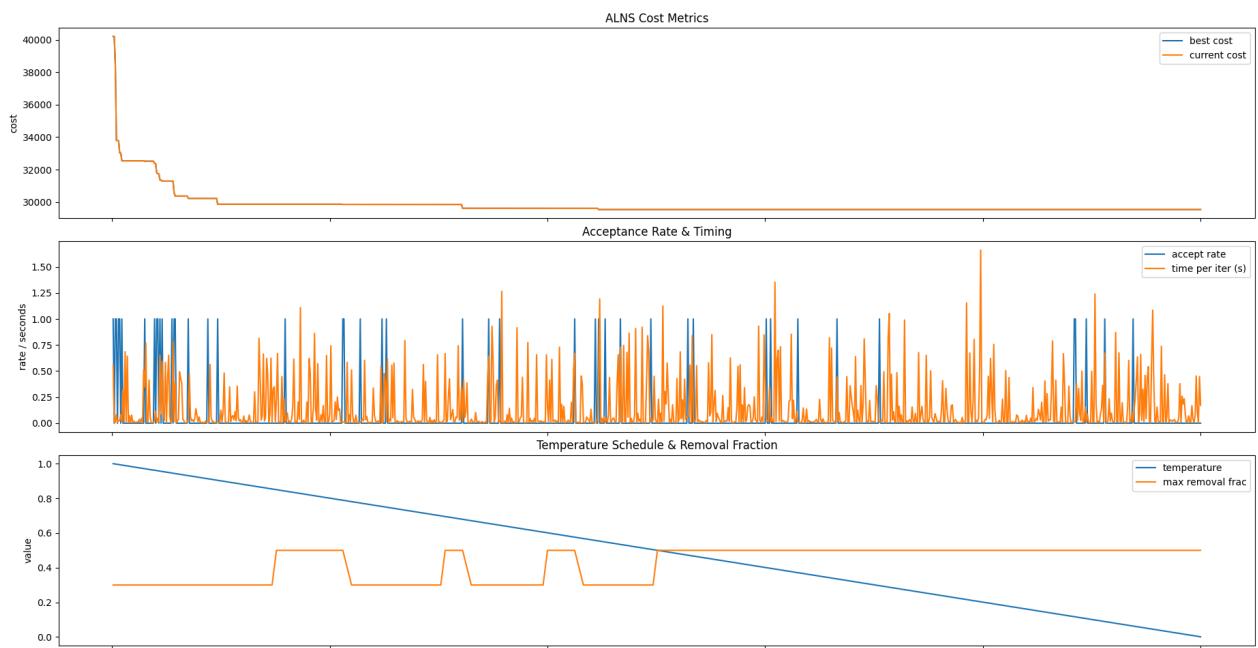
[RESULT] solution:

```
route 1: 1 → 47 → 32 → 25 → 1  
route 2: 1 → 31 → 86 → 12 → 20 → 51 → 1  
route 3: 1 → 6 → 13 → 59 → 1  
route 4: 1 → 24 → 62 → 22 → 1  
route 5: 1 → 80 → 1  
route 6: 1 → 81 → 35 → 65 → 4 → 41 → 1  
route 7: 1 → 9 → 18 → 1  
route 8: 1 → 23 → 87 → 55 → 1  
route 9: 1 → 94 → 76 → 1  
route 10: 1 → 78 → 83 → 29 → 15 → 1
```

```

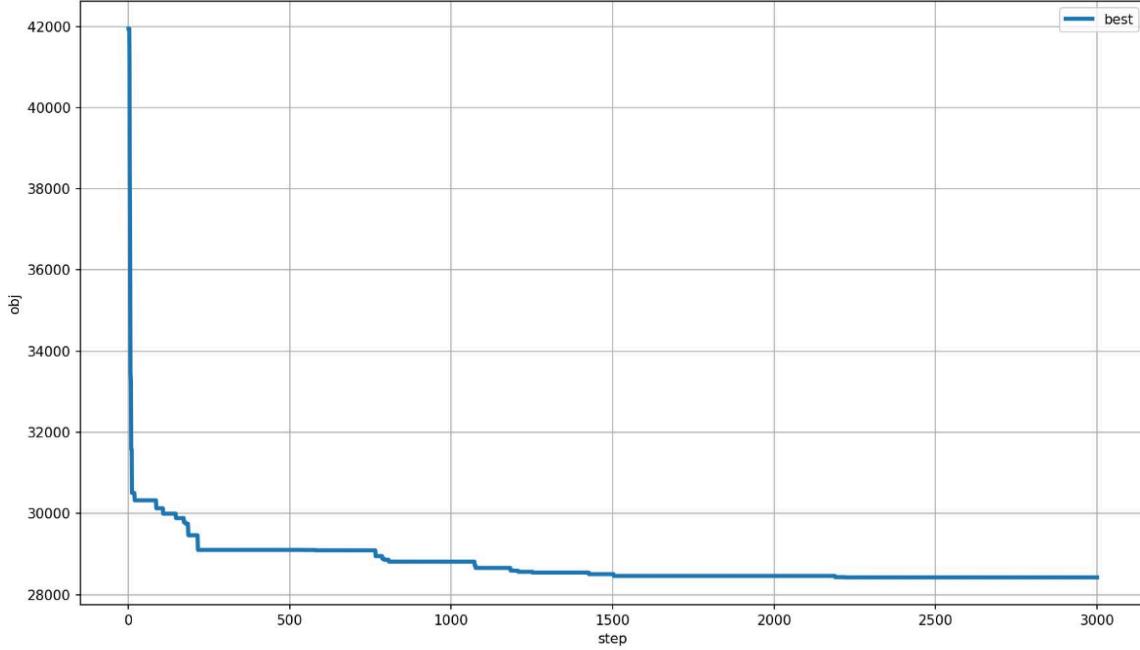
route 10: 1 → 78 → 83 → 29 → 15 → 1
route 11: 1 → 36 → 21 → 42 → 16 → 1
route 12: 1 → 39 → 49 → 88 → 61 → 1
route 13: 1 → 33 → 96 → 74 → 54 → 34 → 1
route 14: 1 → 101 → 28 → 92 → 1
route 15: 1 → 45 → 60 → 97 → 95 → 1
route 16: 1 → 40 → 11 → 43 → 1
route 17: 1 → 89 → 68 → 73 → 64 → 1
route 18: 1 → 53 → 84 → 82 → 1
route 19: 1 → 67 → 85 → 91 → 69 → 1
route 20: 1 → 93 → 71 → 2 → 1
route 21: 1 → 100 → 52 → 98 → 1
route 22: 1 → 14 → 75 → 56 → 77 → 1
route 23: 1 → 66 → 79 → 50 → 37 → 30 → 58 → 1
route 24: 1 → 72 → 63 → 99 → 90 → 48 → 27 → 57 → 1
route 25: 1 → 10 → 17 → 70 → 1
route 26: 1 → 8 → 3 → 46 → 44 → 38 → 7 → 1
route 27: 1 → 19 → 5 → 26 → 1
total cost: 28429.00

```

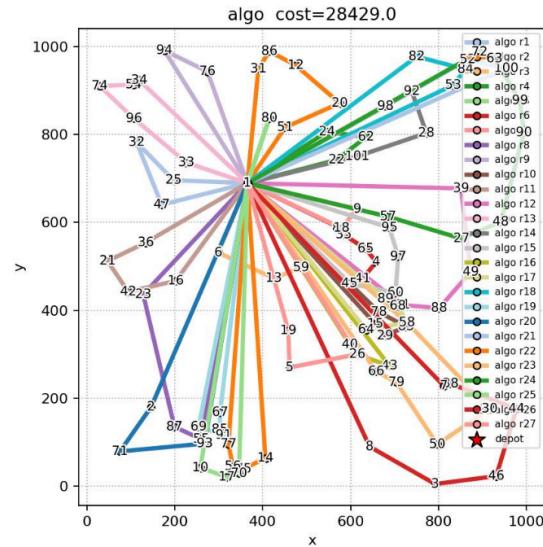
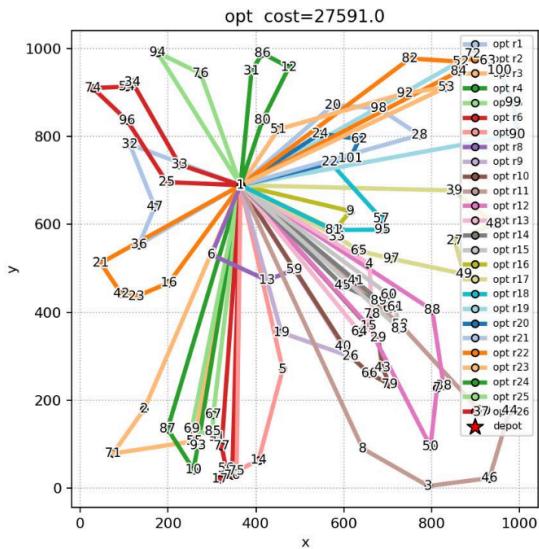


time: 307.95s

ALNS convergence - X-n101-k26.vrp



ALNS - X-n101-k26.vrp



מקנות ודיון על התוצאות:

ה-ALNS שביצעו מבוסס על ארבעה רכיבים עיקריים:

1. **פתרון התחלתי Nearest-Neighbor** – בונה מסלולים תקפים במהירות אך במחיר ראשוני גבוה.
2. **מחולל שכנות Destroy / Repair** – שלוש טכניקות הרס (Random, Worst-Distance, Shaw) ושלוש טכניקות בנייה (Greedy, Regret-2, Greedy-Penalty). משקל כל טכניקה מתעדכן אחת ל-100 איטרציות על-פי יחס ניקוד-שימוש.
3. **קריטריון קבלה בסגנון Simulated Annealing** – טמפרטורה יורדת לנינאיות מערך התחלתי 1.0 אל אפס כמעט מוחלט לאורך הריצה.
4. **חיפוש מקומי** – 2-Opt בעור מסלול, Relocate ו-Swap בהסתברויות נמוכות.

הלוגים וגרפי ההתקנסות מגלים כיצד ארבעת הרכיבים הללו " משחקים " זה עם זה לאורך הריצה.

ירידת עלות חדה בתחילת הריצה

בכל האינסנסים נרשמת צניחה ניכרת בעלות במהלך העשרים-שלושים האיטרציות הראשונות:

- בקובץ k26-k101-X העלות יורדת מכ-42 אלף לכ-30 אלף בחמשים איטרציות.
- בקובץ k17-k200-n200-M העלות יורדת מכ-1 900 לכ-1 460 בכ-150 איטרציות.

שלב זה מתרחש כשהטמפרטורה גבוהה וה-*removal fraction* max עדין מותר הרס ממשועוטי (סבב שלושים אחוז), כך שהקריטריון הסטטיסטי מאשר קפיצות גדולות.

געילה מוקדמת של קריטריון הקבלה

לאחר כ-25 אחוז מן הריצה הטמפרטורה יורדת לרמה שבה כמעט שום פתרון חדש אינו מתתקבל. גם כשה-*max removal fraction* מטפס בהדרגה עד 45 אחוז, מרבית ההצעות נדוחות. لكن עקומת ההתקנסות כמעט נבלמת סבב עלות 1 350 בקובץ k17-k200-n200-M וסבב 28 400 בקובץ k26-k101-X.

התפתחות משקל האופרטורים

עדכון המשקלים התקופתי מעניק עדיפות הולכת וגוברת ל-*Greedy Insertion* – הוא כמעט תמיד מחזיר פתרון תקין וכן צובר ניקוד. אופרטורים הרסניים יותר מאבדים משקל לאחר שהשיפור הראשוני נעלם, ונבחרים לעיתים רוחקות בהמשך. דפוא זה מסביר את הירידה בגין האפקטיבי החל מאמצע הריצה.

כמה איטרציות באמת תורמות לשיפור?

E-n22-k4: מtower 50 איטרציות רק שש הניבו עלות טובה יותר – כ-12 אחוז. כבר לאחר עשר איטרציות האלגוריתם הגיע לאופטימום 375.

P-h16-k8: באינסטנס קטן זה נרשמו שניים-עשר שיפורים מtower חמישים איטרציות – חמישה-עشر אחוז – מיד אחר-כך התוצאות מוחלטת על העלות 450.

A-h80-k10: כ-1500 איטרציות (tower 2500) שיפורו את העלות, בעיקר עד סביב איטרציה 2000 לאחר מכן לא התקבלו פתרונות זולים יותר.

M-h200-k17: רק כ-900 איטרציות מtower 3000 (כ-30 אחוז) הביאו לשיפור, ורובן התרחשו בשליש הראשון של הריצה.

X-h101-k26: כ-100 איטרציות מtower 3000 תרמו לשיפור, אך כמעט כל התרחשו במאתיים האיטרציות הראשונות; לאחר איטרציה 300 הקרייטריון התרמי מנע כמעט כל קבלה נוספת.

כל שמספר הלקחות גדול, שיעור האיטרציות שאין תורמות עולה – לא משומש שהשכונות דומות, אלא מפני שהטמפרטורה הנוכחית חוסמת את רוב הקפיצות.

מבנה הפתרונות הסופיים

- ב uninstantiations הקטנים (k8, P-h16-k4, E-n22-k4) המסלולים הסופיים כמעט חופפים לאופטימום.
- ב-k10-h80-A הפתרון מסתאים בכמה "מסלול לוין" קצרים סיבוב הדיפו ובמסלולים ארוכים בשוליהם – תוצאה של Greedy Insertion ששמער מבנה מוקדם.
- ב-k17-h200-M וב-k101-h26-X מתקנים מסלולים רדיליים עם חיתוך מגזרי לא מאוזן; לאחר שהטמפרטורה נוכחית, לקוחות כמעט אינטגרליים בין מסלולים ו-Opt-2 בטור מסלול אינו מתקן את חלוקת המגזרים.

השפעת max removal fraction בפועל

ה-*max removal fraction* עולה אט-אט כל עוד לא חל שיפור במשר חמישים איטרציות, אך בפועל מספר הלקחות שימושיים נשאר ברוב הזמן סיבוב עשרה-חמשה-עشر אחוז. משמע – האלגוריתם " רשאי" להרווים כמעט מן הפתרון, אך המשקלים שהצטברו דוחפים לבחירה באופרטורים זחים יותר.

סיכום

ה-ALNS מפיק את מרבית השיפור בשלבים הראשוניים, כאשר הטמפרטורה גבוהה והרס-בנייה אגרסיבי. עם ירידת הטמפרטורה, החיפוש נעצר כמעט לחדוטין על ערך עלות-שפלה מקומי וממשיך בעיקר בחיפוש מקומי עדין. משקל האופרטורים מתכנס במהירות לטכניות זהיות, מה שמקטין את הגיוון ואת יכולת פרוץ לעלות טובה יותר בשלבים מאוחרים. لكن מתקנים פתרונות אופטימליים בעיות קטנות ופער מותן (שלושה עד שבעה אחוזים) בעיותBINNING וגדלות. בחירה בפתרון התחלתי פשוט, דעיכה לニアרית של הטמפרטורה, ודינמייקה קשייה של משקל האופרטורים – כל אלה קובעים בפועל את רף הביצועים שלו מגיע האלגוריתם בכל אינסטנס.

Branch and Bound with Limited Discrepancy Search heuristic

מעבר הביעות בקטגורית **Beginner** •

○ הפרמטרים מעבור ריצה זו:

- Time Limitation = 10
- Max_discrepancies = 10
- Knn_k_value = 10
- Initial relax factor = 1.15, allows broader search
- Final_relax_factor = 1.05 (tight near timeout)
- Num_restarts_allowed = 3
- Kmeans_iterations = 8
- distance_weight = 10.0
- Balance_weight = 0.05
- capacity_weight = 5.1

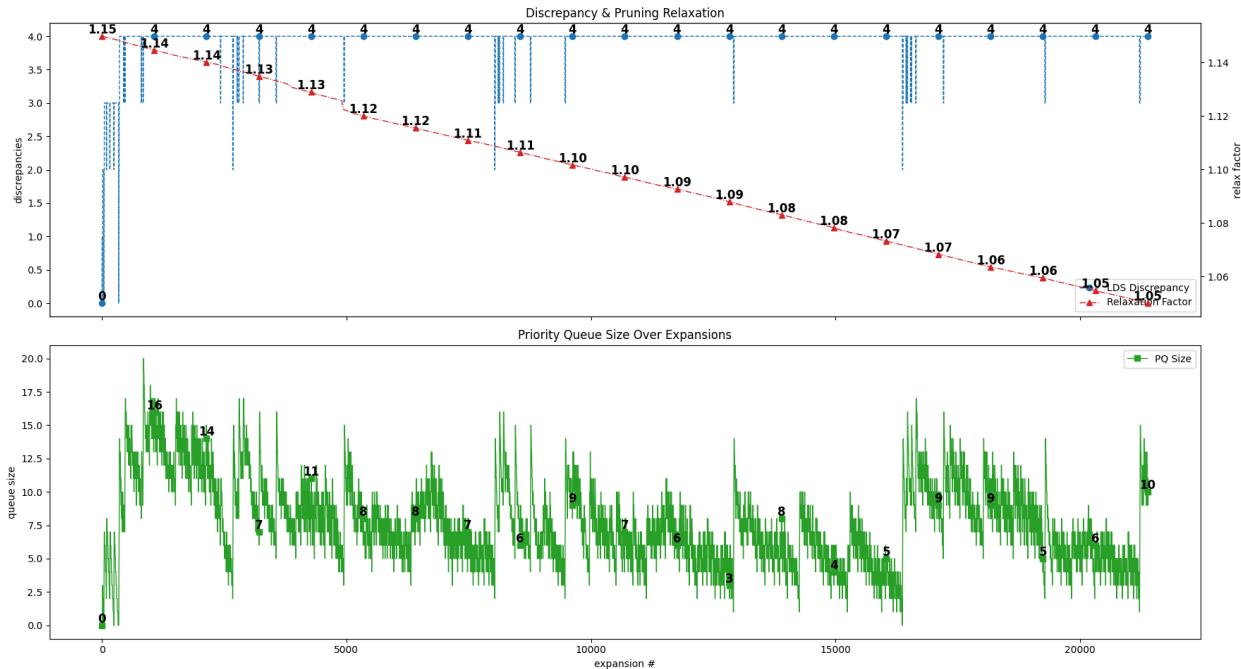
E-n22-k4

```
[INFO] ===== problem E-n22-k4.vrp =====
[INFO] parsed 'E-n22-k4.vrp' → cap=6000, nodes=22, depot=1, minVeh=4
[INFO] reference .sol found - 4 routes, cost=375.0
[INFO] --- BB ---

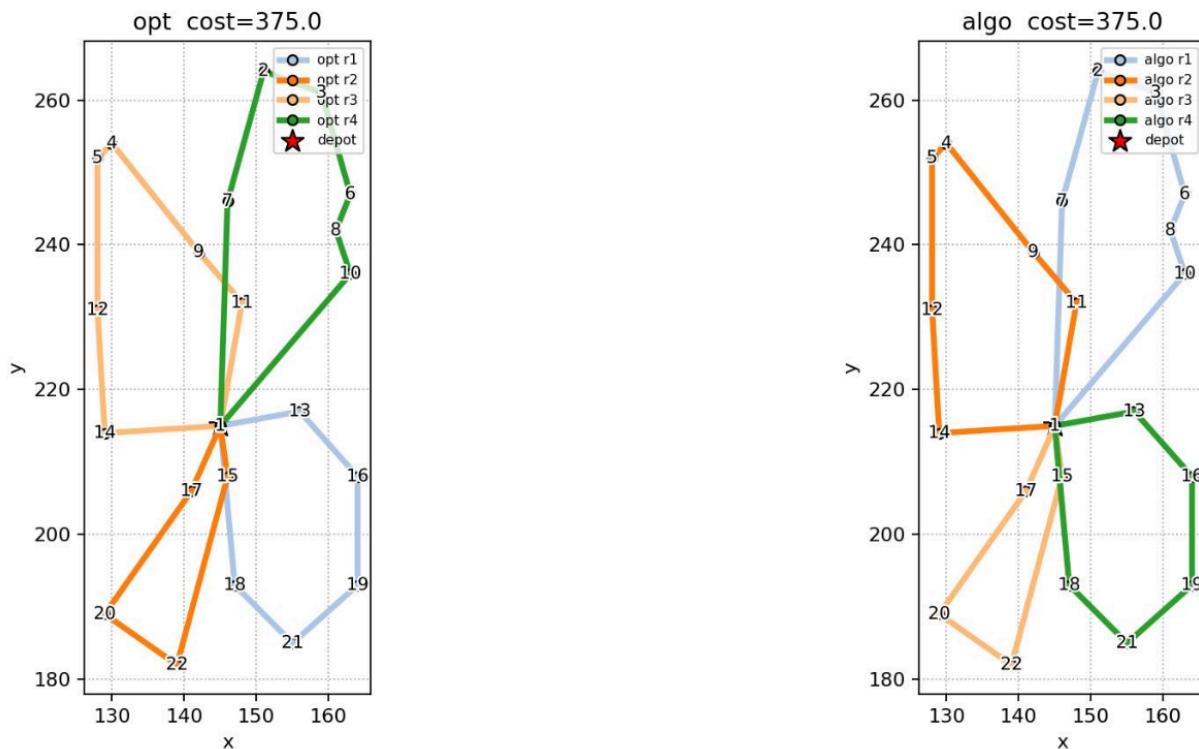
Starting B&B + LDS with 22 nodes and a time limit of 10 seconds

[Branch and Bound with LDS] | nodes expanded: 100 | best fit: 4569.93 | time elapsed: 0.07s
[Branch and Bound with LDS] | nodes expanded: 200 | best fit: 4569.93 | time elapsed: 0.11s
[Branch and Bound with LDS] | nodes expanded: 300 | best fit: 4569.93 | time elapsed: 0.16s
[Branch and Bound with LDS] | nodes expanded: 400 | best fit: 4569.93 | time elapsed: 0.22s
[Branch and Bound with LDS] | nodes expanded: 500 | best fit: 4569.93 | time elapsed: 0.27s
[Branch and Bound with LDS] | nodes expanded: 600 | best fit: 4569.93 | time elapsed: 0.33s
[Branch and Bound with LDS] | nodes expanded: 700 | best fit: 4569.93 | time elapsed: 0.38s
[Branch and Bound with LDS] | nodes expanded: 800 | best fit: 4569.93 | time elapsed: 0.43s
[Branch and Bound with LDS] | nodes expanded: 900 | best fit: 4125.36 | time elapsed: 0.48s
[Branch and Bound with LDS] | nodes expanded: 1000 | best fit: 3845.81 | time elapsed: 0.52s

[Branch and Bound with LDS] | nodes expanded: 23000 | best fit: 3760.43 | time elapsed: 9.93s
[Branch and Bound with LDS] | nodes expanded: 23100 | best fit: 3760.43 | time elapsed: 9.97s
[INFO] BB runtime: 12.80s
[RESULT] solution:
route 1: 1 → 10 → 8 → 6 → 3 → 2 → 7 → 1
route 2: 1 → 14 → 12 → 5 → 4 → 9 → 11 → 1
route 3: 1 → 17 → 20 → 22 → 15 → 1
route 4: 1 → 13 → 16 → 19 → 21 → 18 → 1
total cost: 375.00
```



BB - E-n22-k4.vrp



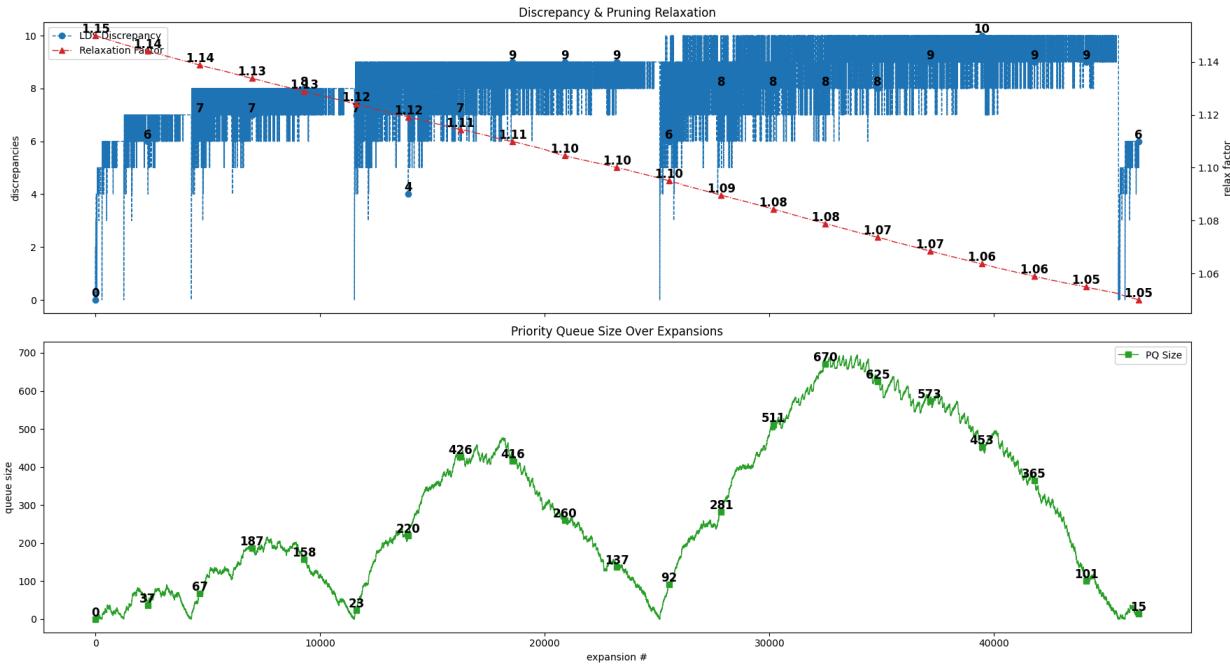
P-n16-k8

```
[INFO] ===== problem P-n16-k8.vrp =====  
[INFO] parsed 'P-n16-k8.vrp' → cap=35, nodes=16, depot=1, minVeh=None  
[INFO] reference .sol found - 8 routes, cost=450.0  
[INFO] --- BB ---
```

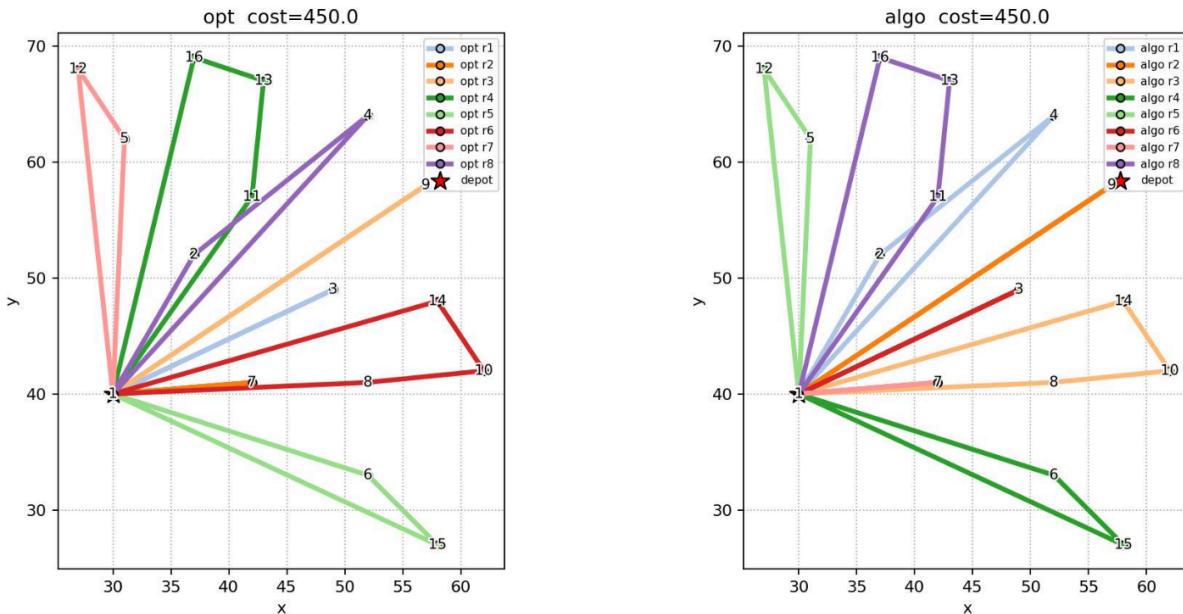
Starting B&B + LDS with 16 nodes and a time limit of 10 seconds

```
[Branch and Bound with LDS] | nodes expanded: 100 | best fit: 4783.16 | time elapsed: 0.03s  
[Branch and Bound with LDS] | nodes expanded: 200 | best fit: 4783.16 | time elapsed: 0.06s  
[Branch and Bound with LDS] | nodes expanded: 300 | best fit: 4783.16 | time elapsed: 0.09s  
[Branch and Bound with LDS] | nodes expanded: 400 | best fit: 4783.16 | time elapsed: 0.12s  
[Branch and Bound with LDS] | nodes expanded: 500 | best fit: 4783.16 | time elapsed: 0.14s  
[Branch and Bound with LDS] | nodes expanded: 600 | best fit: 4783.16 | time elapsed: 0.17s
```

```
[Branch and Bound with LDS] | nodes expanded: 46100 | best fit: 4510.53 | time elapsed: 9.92s  
[Branch and Bound with LDS] | nodes expanded: 46200 | best fit: 4510.53 | time elapsed: 9.94s  
[Branch and Bound with LDS] | nodes expanded: 46300 | best fit: 4510.53 | time elapsed: 9.96s  
[Branch and Bound with LDS] | nodes expanded: 46400 | best fit: 4510.53 | time elapsed: 9.99s  
[INFO] BB runtime: 46.88s  
[RESULT] solution:  
route 1: 1 → 2 → 4 → 1  
route 2: 1 → 9 → 1  
route 3: 1 → 14 → 10 → 8 → 1  
route 4: 1 → 6 → 15 → 1  
route 5: 1 → 5 → 12 → 1  
route 6: 1 → 3 → 1  
route 7: 1 → 7 → 1  
route 8: 1 → 16 → 13 → 11 → 1  
total cost: 450.00
```



BB - P-n16-k8.vrp



• עבור הביעות בקטגורית Intermediate

- הפרמטרים עבור ריצה זו:

- Time Limitation = 90
- Max_discrepancies = 10
- Knn_k_value = 10
- Initial relax factor = 1.55, allows broader search
- Final_relax_factor = 1.15 (tight near timeout)
- Num_restarts_allowed = 3
- Kmeans_iterations = 8
- distance_weight = 10.0
- Balance_weight = 0.05
- capacity_weight = 5.1

A-n32-k5

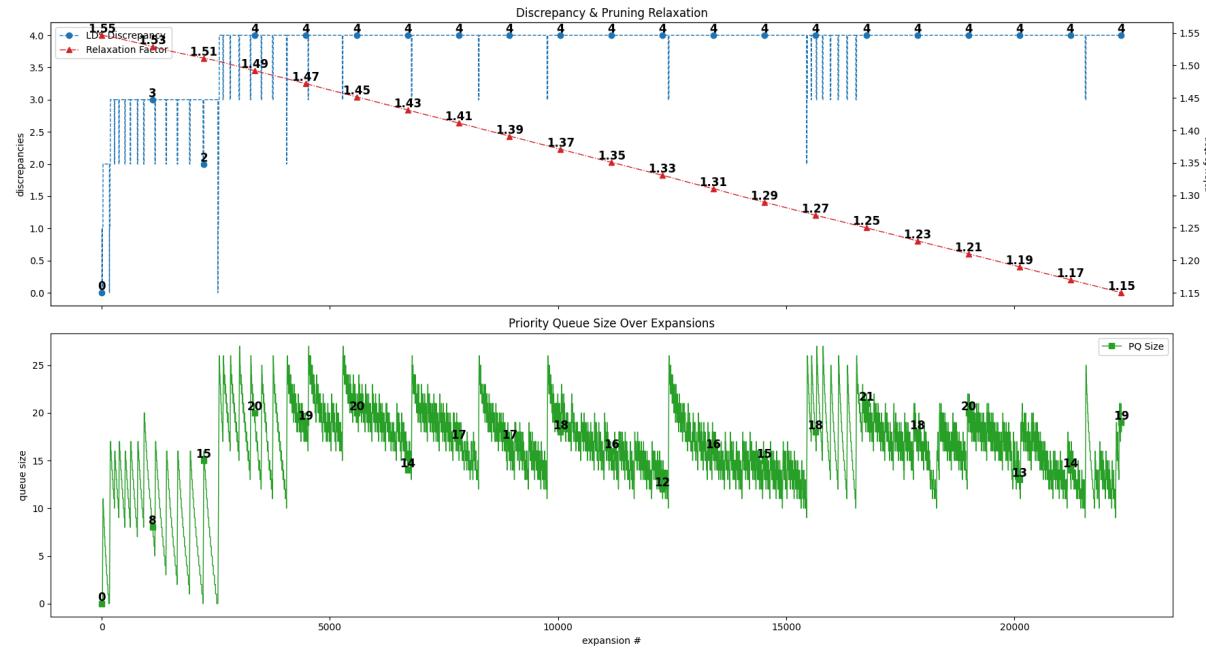
```
[INFO] ===== problem A-n32-k5.vrp =====
[INFO] parsed 'A-n32-k5.vrp' → cap=100, nodes=32, depot=1, minVeh=None
[INFO] reference .sol found - 5 routes, cost=784.0
[INFO] --- BB ---

Starting B&B + LDS with 32 nodes and a time limit of 90 seconds

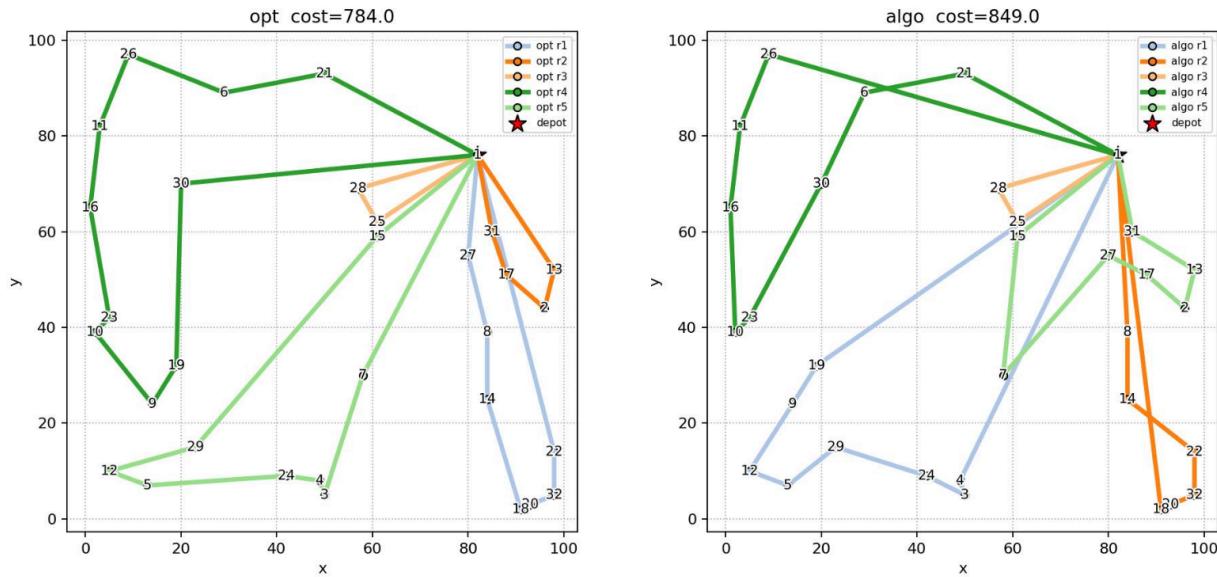
[Branch and Bound with LDS] | nodes expanded: 2000 | best fit: 9064.28 | time elapsed: 1.57s
[Branch and Bound with LDS] | nodes expanded: 4000 | best fit: 9064.28 | time elapsed: 2.78s
[Branch and Bound with LDS] | nodes expanded: 6000 | best fit: 9064.28 | time elapsed: 3.93s
[Branch and Bound with LDS] | nodes expanded: 8000 | best fit: 9064.28 | time elapsed: 5.14s

[Branch and Bound with LDS] | nodes expanded: 150000 | best fit: 8721.15 | time elapsed: 87.14s
[Branch and Bound with LDS] | nodes expanded: 152000 | best fit: 8721.15 | time elapsed: 88.37s
[Branch and Bound with LDS] | nodes expanded: 154000 | best fit: 8721.15 | time elapsed: 89.61s
[INFO] BB runtime: 110.38s

[RESULT] solution:
route 1: 1 → 4 → 3 → 24 → 29 → 5 → 12 → 9 → 19 → 1
route 2: 1 → 8 → 14 → 22 → 32 → 20 → 18 → 1
route 3: 1 → 25 → 28 → 1
route 4: 1 → 21 → 6 → 30 → 23 → 10 → 16 → 11 → 26 → 1
route 5: 1 → 31 → 13 → 2 → 17 → 27 → 7 → 15 → 1
total cost: 849.00
```



BB - A-n32-k5.vrp



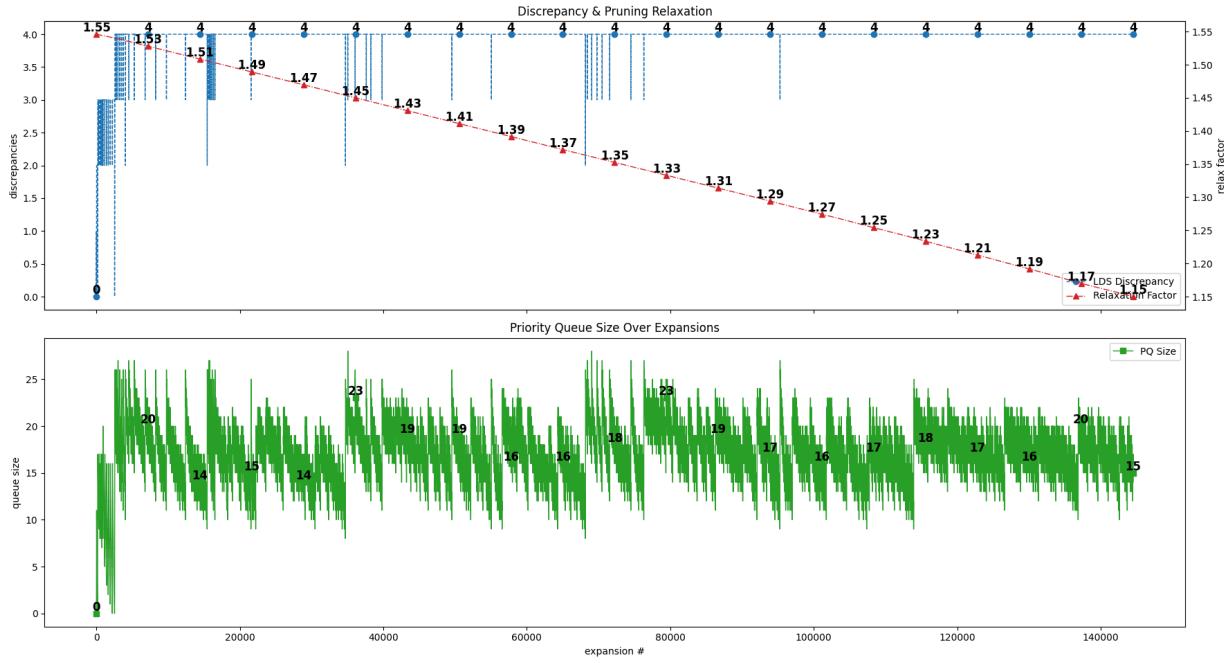
A-n80-k10

```
[INFO] ===== problem A-n80-k10.vrp =====
[INFO] parsed 'A-n80-k10.vrp' → cap=100, nodes=80, depot=1, minVeh=None
[INFO] reference .sol found - 10 routes, cost=1763.0
[INFO] --- BB ---

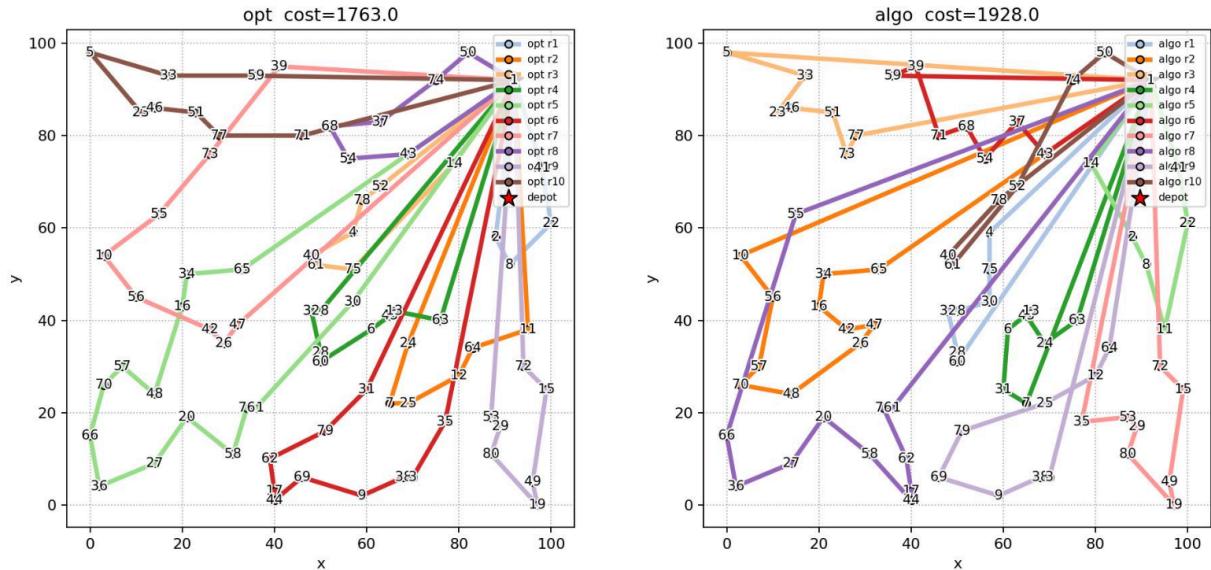
Starting B&B + LDS with 80 nodes and a time limit of 90 seconds

[Branch and Bound with LDS] | nodes expanded: 2000 | best fit: 20343.34 | time elapsed: 7.94s
[Branch and Bound with LDS] | nodes expanded: 4000 | best fit: 20343.34 | time elapsed: 15.61s
[Branch and Bound with LDS] | nodes expanded: 6000 | best fit: 20343.34 | time elapsed: 23.84s
[Branch and Bound with LDS] | nodes expanded: 8000 | best fit: 20343.34 | time elapsed: 31.92s
[Branch and Bound with LDS] | nodes expanded: 10000 | best fit: 20343.34 | time elapsed: 40.07s
[Branch and Bound with LDS] | nodes expanded: 12000 | best fit: 20343.34 | time elapsed: 48.15s

[Branch and Bound with LDS] | nodes expanded: 136000 | best fit: 19783.52 | time elapsed: 564.32s
[Branch and Bound with LDS] | nodes expanded: 138000 | best fit: 19783.52 | time elapsed: 572.90s
[Branch and Bound with LDS] | nodes expanded: 140000 | best fit: 19783.52 | time elapsed: 581.19s
[Branch and Bound with LDS] | nodes expanded: 142000 | best fit: 19783.52 | time elapsed: 589.28s
[Branch and Bound with LDS] | nodes expanded: 144000 | best fit: 19783.52 | time elapsed: 597.81s
[INFO] BB runtime: 2353.25s
[RESULT] solution:
route 1: 1 → 4 → 75 → 30 → 18 → 32 → 28 → 60 → 1
route 2: 1 → 65 → 34 → 16 → 42 → 47 → 26 → 48 → 70 → 57 → 56 → 10 → 1
route 3: 1 → 77 → 73 → 51 → 46 → 23 → 33 → 5 → 1
route 4: 1 → 63 → 24 → 45 → 13 → 6 → 31 → 7 → 1
route 5: 1 → 41 → 22 → 11 → 8 → 2 → 14 → 1
route 6: 1 → 43 → 37 → 54 → 67 → 68 → 71 → 39 → 59 → 1
route 7: 1 → 72 → 15 → 49 → 19 → 80 → 29 → 53 → 35 → 1
route 8: 1 → 21 → 76 → 62 → 17 → 44 → 58 → 20 → 27 → 36 → 66 → 55 → 1
route 9: 1 → 64 → 12 → 25 → 79 → 69 → 9 → 38 → 3 → 1
route 10: 1 → 50 → 74 → 52 → 61 → 40 → 78 → 1
total cost: 1928.00
```



BB - A-n80-k10.vrp



• עבור הביעות בקטגוריה Advanced

- הפרמטרים עבור ריצה זו:

- Time Limitation = 1800 (30 minutes)
- Max_discrepancies = 50
- Knn_k_value = 20
- Initial relax factor = 1.55, allows broader search
- Final_relax_factor = 1.15 (tight near timeout)
- Num_restarts_allowed = 10
- Kmeans_iterations = 10
- distance_weight = 10.0
- Balance_weight = 0.05
- capacity_weight = 5.1

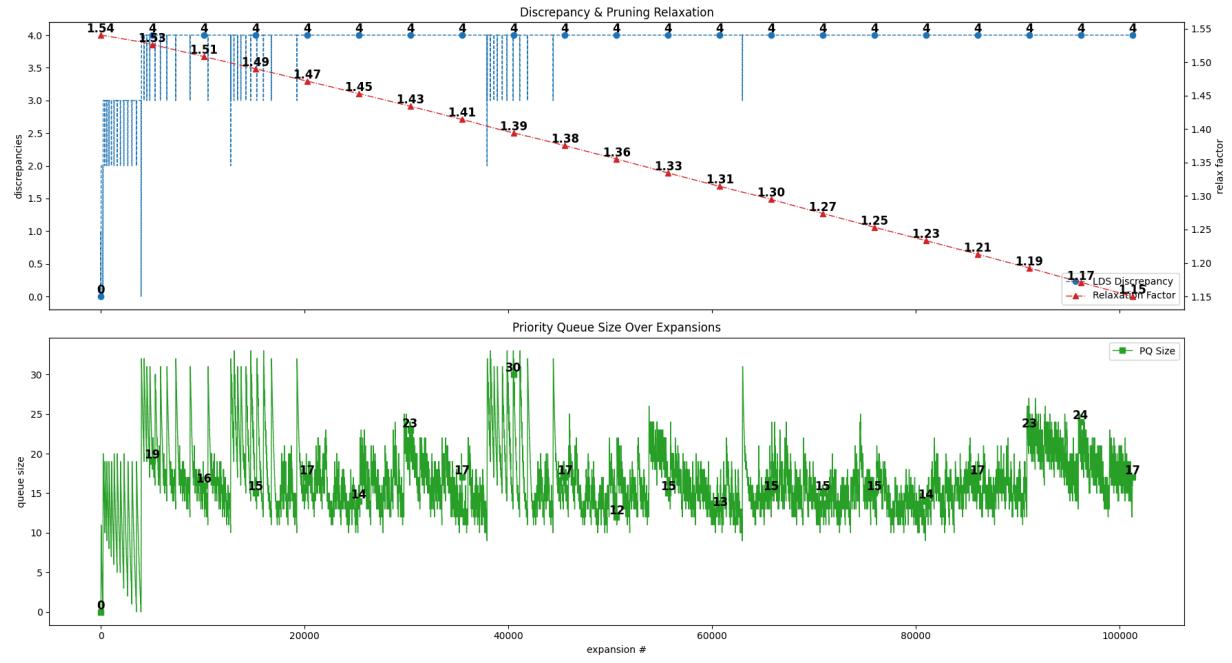
M-n200-k17

```
[INFO] ===== problem M-n200-k17.vrp =====
[INFO] parsed 'M-n200-k17.vrp' → cap=200, nodes=200, depot=1, minVeh=None
[INFO] reference .sol found - 17 routes, cost=1275.0
[INFO] --- BB ---

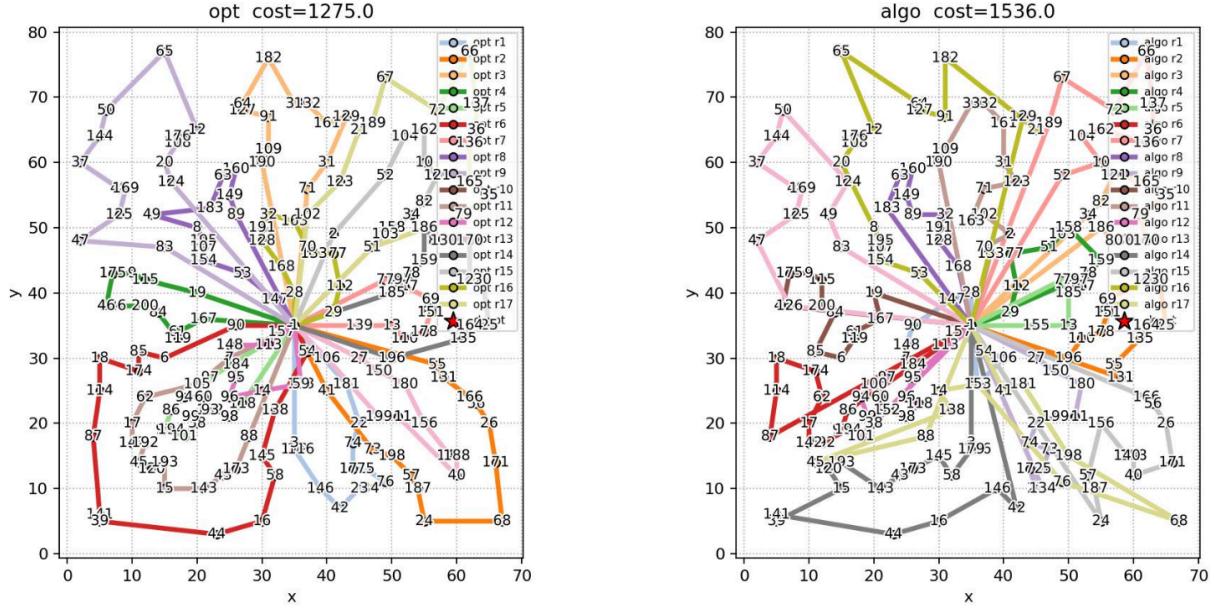
Starting B&B + LDS with 200 nodes and a time limit of 1800 seconds

[Branch and Bound with LDS] | nodes expanded: 2000 | best fit: 15630.51 | time elapsed: 65.73s
[Branch and Bound with LDS] | nodes expanded: 4000 | best fit: 15630.51 | time elapsed: 88.72s
[Branch and Bound with LDS] | nodes expanded: 6000 | best fit: 15630.51 | time elapsed: 121.31s
[Branch and Bound with LDS] | nodes expanded: 8000 | best fit: 15630.51 | time elapsed: 153.26s

[Branch and Bound with LDS] | nodes expanded: 96000 | best fit: 15630.51 | time elapsed: 1701.73s
[Branch and Bound with LDS] | nodes expanded: 98000 | best fit: 15630.51 | time elapsed: 1739.40s
[Branch and Bound with LDS] | nodes expanded: 100000 | best fit: 15630.51 | time elapsed: 1776.58s
[INFO] BB runtime: 2742.10s
[RESULT] solution:
route 1: 1 → 28 → 147 → 90 → 148 → 7 → 184 → 113 → 157 → 54 → 106 → 59 → 1
route 2: 1 → 196 → 110 → 178 → 81 → 151 → 69 → 122 → 30 → 25 → 164 → 135 → 55 → 131 → 1
route 3: 1 → 34 → 82 → 121 → 165 → 35 → 79 → 170 → 130 → 80 → 186 → 1
route 4: 1 → 29 → 112 → 177 → 51 → 103 → 158 → 4 → 159 → 78 → 117 → 197 → 1
route 5: 1 → 139 → 155 → 13 → 185 → 77 → 1
route 6: 1 → 86 → 92 → 194 → 192 → 142 → 17 → 62 → 174 → 18 → 114 → 87 → 1
route 7: 1 → 52 → 10 → 104 → 162 → 136 → 36 → 137 → 66 → 72 → 67 → 189 → 1
route 8: 1 → 168 → 128 → 191 → 32 → 89 → 149 → 160 → 63 → 183 → 1
route 9: 1 → 27 → 150 → 180 → 111 → 199 → 22 → 74 → 73 → 75 → 172 → 23 → 134 → 1
route 10: 1 → 19 → 167 → 61 → 119 → 6 → 85 → 84 → 200 → 115 → 9 → 175 → 46 → 1
route 11: 1 → 133 → 70 → 2 → 102 → 163 → 71 → 123 → 31 → 161 → 132 → 33 → 109 → 11 → 190 → 1
route 12: 1 → 95 → 96 → 118 → 98 → 93 → 152 → 60 → 97 → 105 → 100 → 94 → 99 → 38 → 101 → 1
route 13: 1 → 83 → 49 → 124 → 50 → 144 → 37 → 48 → 169 → 125 → 47 → 126 → 1
route 14: 1 → 3 → 116 → 179 → 58 → 145 → 173 → 43 → 143 → 193 → 120 → 15 → 39 → 141 → 44 → 16 → 146 → 42 → 1
route 15: 1 → 166 → 56 → 26 → 171 → 40 → 188 → 140 → 156 → 5 → 57 → 187 → 24 → 1
route 16: 1 → 53 → 154 → 107 → 195 → 8 → 20 → 108 → 176 → 12 → 65 → 64 → 127 → 91 → 182 → 21 → 129 → 1
route 17: 1 → 41 → 181 → 198 → 68 → 76 → 153 → 14 → 138 → 45 → 88 → 1
total cost: 1536.00
```



BB - M-n200-k17.vrp



X-n101-k26

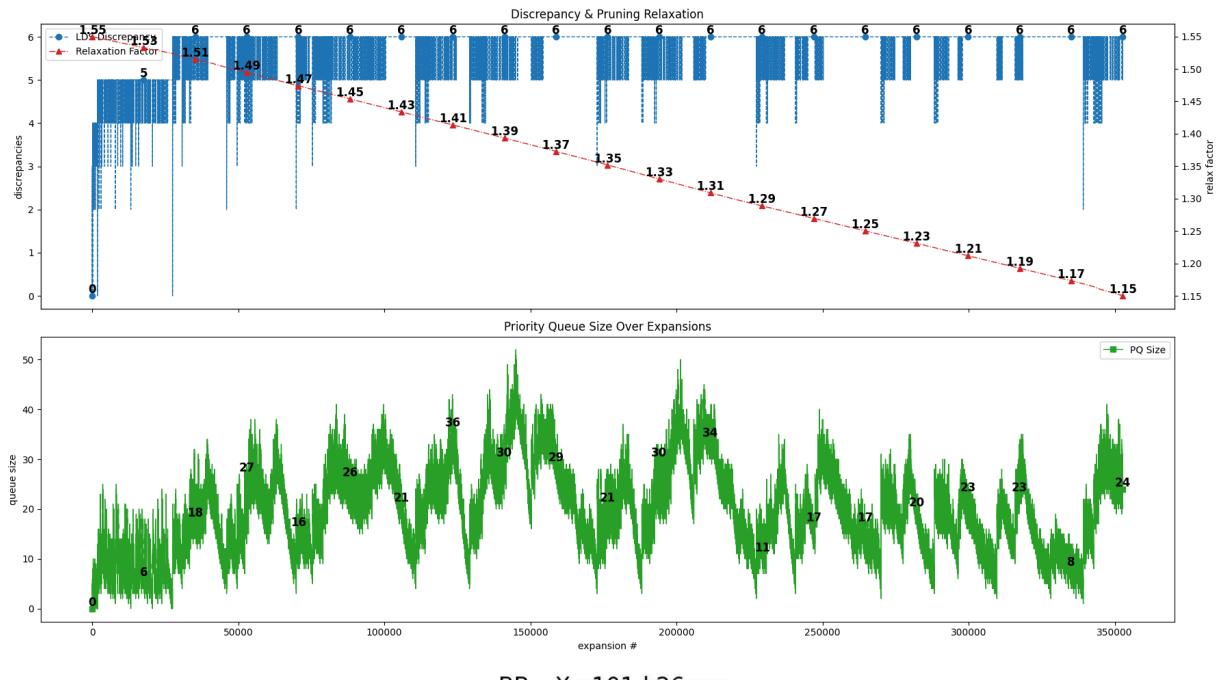
```
[INFO] ===== problem X-n101-k26.vrp =====
[INFO] parsed 'X-n101-k26.vrp' → cap=206, nodes=101, depot=1, minVeh=None
[INFO] reference .sol found - 26 routes, cost=27591.0
[INFO] --- BB ---

Starting B&B + LDS with 101 nodes and a time limit of 1800 seconds

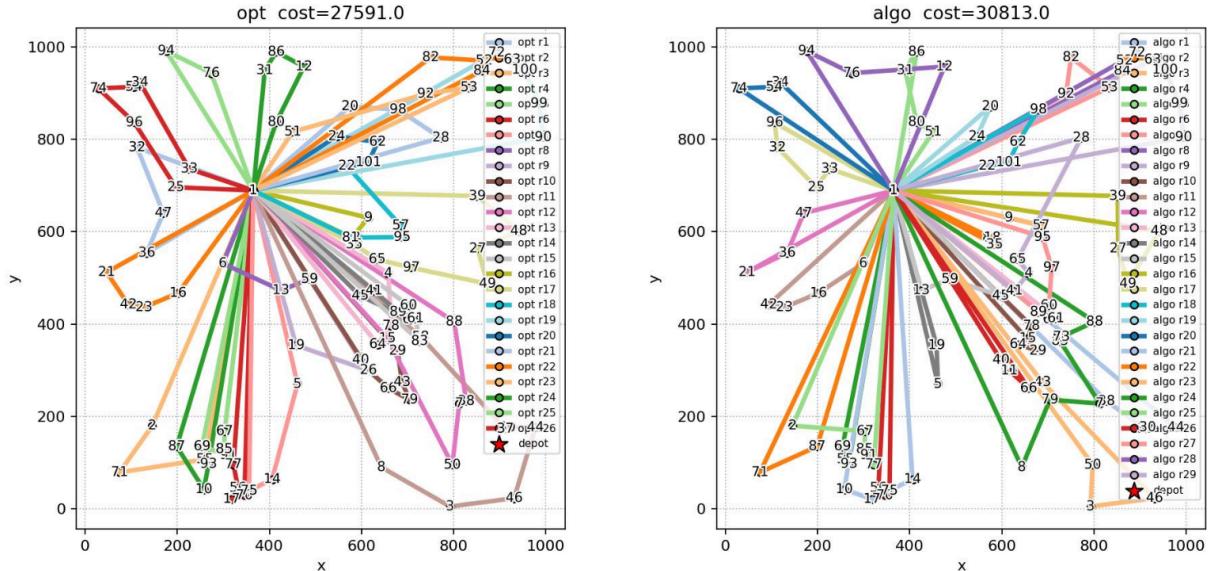
[Branch and Bound with LDS] | nodes expanded: 2000 | best fit: 319136.66 | time elapsed: 9.63s
[Branch and Bound with LDS] | nodes expanded: 4000 | best fit: 319136.66 | time elapsed: 18.53s
[Branch and Bound with LDS] | nodes expanded: 6000 | best fit: 319136.66 | time elapsed: 27.41s
[Branch and Bound with LDS] | nodes expanded: 8000 | best fit: 319136.66 | time elapsed: 36.07s
[Branch and Bound with LDS] | nodes expanded: 10000 | best fit: 319136.66 | time elapsed: 44.70s

[Branch and Bound with LDS] | nodes expanded: 346000 | best fit: 319136.66 | time elapsed: 1759.99s
[Branch and Bound with LDS] | nodes expanded: 348000 | best fit: 319136.66 | time elapsed: 1771.42s
[Branch and Bound with LDS] | nodes expanded: 350000 | best fit: 319136.66 | time elapsed: 1782.91s
[Branch and Bound with LDS] | nodes expanded: 352000 | best fit: 319136.66 | time elapsed: 1795.01s

[INFO] BB runtime: 1881.13s
[RESULT] solution:
route 1: 1 → 38 → 37 → 30 → 44 → 1
route 2: 1 → 81 → 18 → 35 → 1
route 3: 1 → 9 → 57 → 1
route 4: 1 → 69 → 55 → 93 → 1
route 5: 1 → 51 → 80 → 31 → 86 → 1
route 6: 1 → 56 → 70 → 75 → 1
route 7: 1 → 92 → 82 → 53 → 1
route 8: 1 → 84 → 52 → 72 → 1
route 9: 1 → 63 → 100 → 99 → 90 → 1
route 10: 1 → 78 → 15 → 64 → 29 → 1
route 11: 1 → 6 → 16 → 23 → 42 → 1
route 12: 1 → 47 → 36 → 21 → 1
route 13: 1 → 68 → 61 → 1
route 14: 1 → 19 → 5 → 1
route 15: 1 → 13 → 59 → 45 → 1
route 16: 1 → 39 → 27 → 49 → 48 → 1
route 17: 1 → 33 → 25 → 32 → 96 → 1
route 18: 1 → 22 → 101 → 62 → 98 → 1
route 19: 1 → 24 → 20 → 1
route 20: 1 → 34 → 54 → 74 → 1
route 21: 1 → 14 → 17 → 10 → 1
route 22: 1 → 87 → 71 → 1
route 23: 1 → 50 → 3 → 46 → 43 → 1
route 24: 1 → 8 → 79 → 7 → 83 → 58 → 73 → 88 → 1
route 25: 1 → 2 → 67 → 85 → 91 → 77 → 1
route 26: 1 → 40 → 26 → 11 → 66 → 1
route 27: 1 → 95 → 97 → 60 → 89 → 1
route 28: 1 → 12 → 76 → 94 → 1
route 29: 1 → 41 → 4 → 65 → 28 → 1
total cost: 30813.00
```



BB - X-n101-k26.vrp



מקנות ודיון על הפתרון:

האלגוריתם שלנו משלב Branch-and-Bound עם Limited-Discrepancy Search ו-“relax factor” שמצטמצם לאורך הריצה, וריסטארטים שמצונים מ-*k*-means-Clarke-Wright-i-factor. בדקנו שיש בעיות CVRP בגודלים שונים. להלן הממצאים העיקריים.

התוצאות מול העלות האופטימלית

- **(22) 22-k4-E ל��ות** – העלות ירדה ל-375 תוך פחות מעשר שניות וכ-25 אלף התרחבות. כאן השילוב B&B + LDS כיסה כמעט את כל מרחב הפתרונות במהירות.
- **(32) 32-k5-A ל��ות** – בשתי הבדיקות הראשונות מצאנו 849 (פער של כ-8% מהעלות, 784), ואחר-כך לא היה שיפור אף שהעץ התרחב ל-154 אלף קשרים. מיקדם ההקללה ירד מ-1.55 אל 1.17, ומרגע שעבר את 1.30 לא נספו פתרונות זולים יותר.
- **90 A-n80-k10-A-n101-k26-X** – ירידת חדה בתחלת הריצה, אחר-כך קיפאון ארוך. אחרי חלון זמן של 90 שניות (A-n80-k10) או חצי שעה (A-n101-k26-X) נשאר פער של כ-10-12% מהאופטימום.
- **1 (200) 200-k17-M-n200-k200 M ל��ות** – בחלון של 45 דקוט העץ גדל ל-100 אלף קשרים; העלות נעצרה ב-1.30. רוב הירידה התרחשה בשליש הראשון של הזמן.

מיקדם ההקללה ומספר הדיסקרפנציות

מיקדם ההקללה מתחילה ערך גבוה (1.55 באינסטנסים המתקדים, 1.15 בקטנים) ויורד ליניארית. ברגע שהוא יורד מתחת ל-1.30 כמעט שלא מתבלות יותר התרחבות משפרות. מספר הדיסקרפנציות המרבי (למשל 4 או 6) מנוצל רק בהתחלה; בהמשך רוב הקשרים שבתו עומדים כבר במגבלת הדיסקרפנציות, מה שמצוין את החיפוש בפועל הרבה לפני תום הזמן.

דפוסי תור-העדיפות

הגף הירוק (גודל PQ) מראה תנודות מחזוריות: התור תופח ל-30-50 קשרים, אחר-כך נחתר בסדרת פעולות גיזום, ושוב צומח. מחזור זהה מתרחש כל פעם שה-*relax factor* מתרחק מעט ומאפשר ענפים חדשים, אז מגננו הגיזום סוגר אותם ברגע שה-*ea* שלהם כבר לא עומד בסוף החדש.

השפעת הזרים והריסטארטים

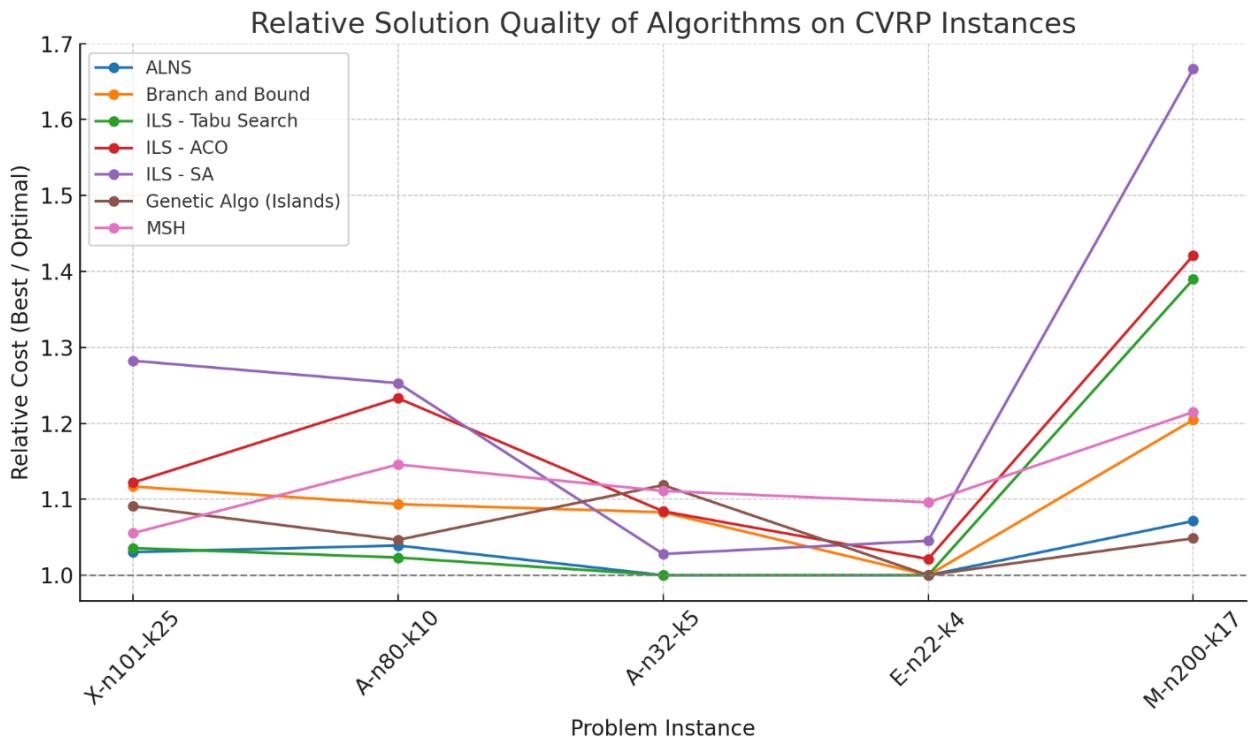
באינסטנסים הגדלים, ה-*k*-means-Clarke-Wright-i-factor מחדו מהר את העלות הראשונית. מרגע שנקבע *best_fit*, אף אחד מהריסטארטים המאוחרים לא הצליח לשפר אותו – מה שניכר בהיסטוריה bound-improvement-snouteraה-*ea*.

תמונה הביצועים הכללת

- בעיות קטנות (עד 22 לקוחות) האלגוריתם מגיע לאופטימום כמעט מיד.
- בעיות בינוניות וגדלות רוב השיפור קורה מוקדם מאוד, ואז מתחילה קיפאון: העץ ממשיר לגדייל, אבל המוקדם והדיסקרפנציות מעבירים את החיפוש למצב "זהיר" שלא נותן לשפר את הבעיות.
- ככל שמספר הלקוחות עולה, הפער מן האופטימום גדול בצורה עקבית – מאפס אחז ב-4-k22-n-E ועד כ-20% ב-200-k17-n-M.

התוצאות האלו מעידות שהשילוב LDS + Branch-and-Bound מסוגל למצוא פתרון חזק ולשפר פתרונות במעט, אך נטה להתקבע מוקדם כאשר הבעיה גדולה וה-relax factor מצטמצם.

סיכום התוצאות



עקבות כללית

- ALNS** נע בין 1.00 ל-1.07 לאורך כל האינסנסים.
- ה-GA במודל האיים** נע בין 1.00 ל-1.09.
- התרכחות של שני אלגוריתמים אלו נשמרת בתחום צר (פחות מ-9% חירגה), ללא תלות במספר הלקוחות.

בעיות קטנות (≤ 32 , E-n22-k4, A-n32-k5, A-n80-k10, X-n101-k25, 80-101-L קומות)

כל האלגוריתמים חוץ מ-ILS-ACO ו-MSH מגיעים לייחס ≥ 1.12 . Branch-and-Bound, ALNS, GA ו-ILS-Tabu מתייצבים ישירות על האופטימום (1.00), מה שמצויב על מרחב חיפוש קטן שבו מגבלות הזמן כמעט אין מוגבלות.

בעיות בינוניות (80-101, X-n101-k25, 80-101-A, A-n80-k10, E-n22-k4, M-n200-k17, L קומות)

- ALNS** ו-**GA ILS-Tabu** שומרים יחס 1.02-1.05.

עליה ל-1.09–1.12 – התרכחות העז איטית יותר מהגבלת הזמן.

- **ILS-ACO, ILS-SA** ו**MSH** עוברים את 1.15 ומודגימים פער גדול ככל שהבעיה נעשית פחות קומפקטיבית.

בעיה גדולה (200-k17, 200-k200-M לקוחות)

- **(1.05) (GA) (ALNS) (1.07)** נותרים מתחת ל-10% חריגה.
- **Branch-and-Bound** עבר ל-1.20.
- **ILS-SA-ILS-Tabu, ILS-ACO** קופצים ל-1.39, 1.42 ו-1.67 בהתאם – פער של עד 67% מהאופטימום.
- **MSH** שומר על תחום ביןים (1.21).

תבנית סטיה עם גודל הבעיה

- עד 80 לקוחות כמעט כל השיטות מציגות יחס ≥ 1.15 .
- מעבר ל-100 לקוחות, רק **ALNS** וה-**GA** שומרים על קצב עלות ליניארי מול האופטימום; השאר מציגים צמיחה חרדה.
- **Branch-and-Bound** מדוק בבעיות קטנות אך מראה עלייה אחידה ביחס לכל שטף הלקוחות גדול.

מסקנות יישורות מן הנתונים

1. **ALNS** וה-**GA** מפגינים את השילוב הטוב ביותר ביוטר של דיקוי ויציבות, עם חריגה מרבית של 9% ומעלה, עקב בכולם האינסטנסים.
2. **Tabu-ILS** יעיל עד גודל בינוני, אך מעל 100 לקוחות היחס עולה ב-35-40%.
3. **ILS-SA** הוא הרגיש ביותר לסקיל, עם עלייה רציפה עד 67% ב-200 לקוחות.
4. **Branch-and-Bound** עם **LDS** נשען על זמן ריצה: ברגע שתקורת הזמן חוסמת את התרחבות העץ, היחס גדל ל-1.12–1.20.
5. **MSH** שומר על פער בגין קבוע, ללא שיין ביצוע אך גם ללא קrise חרדה.

התפלגות היחסים ממחישה כיצד האלגוריתמים מותמודדים עם גידול אקספוננציאלי במרחב הפתרונות: שיטות אדפטיביות (ALNS, GA-Islands) משמרות אינטראקצייתם עם העלייה בממד הבעיה, בעוד שיטות החיפוש המדרגות או הגיזום הקשיח מאבדות דיק ניכר ככל שהאינסטנס גדול.