בס"ד

סיכום מאמר: Hybrid recursive number partitioning (Korf, 2011)

מגיש: כפיר גולדפרב

אבסטרקציה:

הבעיה: בהינתן n מספרים טבעיים (אי-שליליים) ומספר k, נדרש לחלק את המספרים ל-k קבוצות שונות כך שסכום הקבוצות כמה שיותר קרוב (סכומים קרובים בין כל הקבוצות).

קיימים מספר אלגוריתמים ידועים אשר יודעים לחלק בצורה טובה עבור חלוקה דו-כיוונית (חלוקה עבור 2 קבוצות), אבל חלוקה שהיא מולטי-כיוונית (חלוקה עבור מספר (k) קבוצות) היא יותר מאתגרת ומסובכת, במאמר זה ננסה לשלב מספר אלגוריתמים הפותרים בעיה זו בצורה מולטי-כיוונית.

כותבי המאמר עשו ניסויים על האלגוריתמים המוצגים במאמר על מספרים הבנויים מ-31 ביטים.

הקדמה:

דוגמה לחלוקה בין 2 קבוצות, נניח שכקלט קיבלנו את הקבוצה: $\{8,7,6,5,4\}$ אם נחלק אותם לשני קבוצות: $\{8,7\}$ ו- $\{6,5,4\}$ הסכום של כל קבוצה הוא 15, במצב כזה ניתן לומר שהתוצאה היא אופטימלית (כי שווה לחלוטין) או במילים אחרות: <u>חלוקה מושלמת</u>, במידה והסכום של המספרים לא ניתן לחלק במספר התתי-קבוצות – אז סכומי התתי-קבוצות בחלוקה מושלמת יהיה בהפרש של 1.

חלוקת מספרים היא כנראה הבעית ב-NP-complete הכי פשוטה לתיאור,

יישום אחד הוא ה-multi-processor scheduling (מתזמן של נספר מעבדים/ליבות) מאת Garey and Johnson 1979, בהינתן קבוצה של משימות (לכל משימה מקושר גם זמן השלמה – הזמן שלוקח למשימה להשלים), וגם 2 או יותר מעבדים/ליבות, נקצה כל משימה למעבד (שיבצע אותה) כך שכל המשימות יבוצעו כמה שיותר מהר.

יישום נוסף של חלוקת מספרים הוא voting-manipulation (מניפולציה של הצבעה) מאת Walsh 2009, קיימים 3 פונקציות טבעיות עבור חלוקת מספרים:

1. למזער את הקבוצה עם הסכום הגדול ביותר.
2. למקסם את הקבוצה עם הסכום הכי קטן.
3. למזער את ההפרש בין הקבוצה עם הסכום הכי קטן לקבוצה עם הסכום הכי גדול.

עבור חלוקת מספרים של שני קבוצות, שלושת השיטות הנ"ל זהות מבחינת יישום (שוות ערך), אבל עבור חלוקה של מספר קבוצות (יותר מ-2) אז שיטות אלה לא שוות ערך, בחרנו את המזעור של הקבוצה עם הסכום הכי גבוה אשר מתאים למזעור הזמן הכולל במתזמן.

שימוש בסט של אלגוריתמים יכול למקסם את הקבוצה עם הסכום המינימלי, שזו המטרה של היישום של המניפולציה של הצבעה, למזער את הקבוצה עם הסכום המקסימלי יכול לאפשר לאלגוריתם חלוקת המספרים שלנו להיות כמו אלגוריתם אריזה, באלגוריתם אריזה כל קבוצת מספרים מקבלת פח כאשר כל קבוצה אינה מתאימה לגודל הקבוצה שלה, המטרה היא להתאים כל קבוצה בפח שלה בצורה מקסימלית כך שנשתמש כמה שפחות פחים.

בפועל, קירובים היוריסטיים לאריזות פחים כמו הורדה בהתאמה הטובה ביותר (best-fit decreasing) משתמשים רק בכמה פחים יותר מאשר הגבול התחתון (lower bound) כמו הסכום של כל המספרים המחולקים על ידי קיבולת פח, זה אסטרטגיה יעילה עובר חלוקה לפחים, והרעיון הוא להמשיך את האלגוריתם הזה כדי למזער את מספר הפחים עד שפתרון זה כבר לא אפשרי, או שהגענו לחסם התחתון.

מבין אלגוריתמי החלוקה שלנו מסוג branch-and-bound, האלגוריתם שממזער את הקבוצה עם הסכום הכי גבוהה הוא נותן את התוצאה הכי טובה שראינו עד כה, כאשר עם אלגוריתם זה אנחנו יכולים לפתור את בעיית

החלוקה לאריזות פחים עם מספר מדויק של k פחים שונים, על ידי חלוקה של המספרים ל-k קבוצות שונות בהתאם, עם הקבוצה עם הסכום המקסימלי ששווה לתכולה של הפח.

עבודה קודמת:

ראשית נתאר ונסביר על אלגוריתמים עבור חלוקת מספרי.

אלגוריתם חמדן שלם (Complete Greedy Algorithm - *CGA*):

תהליך האלגוריתם החמדן (לא שלם):

1. ממיין את המספרים בסדר יורד.
2. לכל מספר מקצה קבוצה עם סכום הכי קטן עד כה.

תהליך האלגוריתם החמדן השלם:

1. ממיין את המספרים בסדר יורד.
2. ולאחר מכן מחפש בעץ כאשר כל רמה בעץ (level) מיועד למספר אחר, וכל זרוע מיועדת לקבוצה אחרת.
3. כדי להימנע מכפילות ניקח את הפרמוטציות של כל תת-קבוצה וניקח את ההפרש ביניהם.
4. נמשיך לעקוב אחר הקבוצה עם הסכום הכי גבוה עם הפתרון הכי טוב.
    a. אם סכום הקבוצות חרג או שווה לגבול שהצבנו לו, האלגוריתם נכשל.
    b. אם הגענו לחלוקה מושלמת או שהסכום של הקבוצה הכי גדולה שווה למספר הכי גדול, נחזיר את התוצאה כמו שהיא בלי חלוקה מושלמת.

סיבוכיות האלגוריתם היא קצת יותר טובה מ: $O(k^n)$.

אלגוריתם (Karmarkar-Karp Heuristic – *KK*):

אלגוריתם הפועל בזמן פולינומיאלי, יכול לעבוד עבור כל מספר של תת-קבוצות אבל עובד הכי טוב עבור שני קבוצות, תהליך האלגוריתם לשני קבוצות:

1. ממיין את המספרים בסדר יורד.
2. לאחר מכן מחליף בין שני המספרים הכי גדולים בקבוצה בסדר ממוין בין ההפרש שלהם.
3. לאחר מכן נפריד בניהם לשני קבוצות שונות (שני המספרים הכי גדולים).
4. נמשך את התהליך עד שנגיע למספר אחיד שהוא ההפרש הסופי בין שני סכומי התתי-קבוצות.
5. נוסיף את ההפרש הנותר לשני סכומי תתי-הקבוצות ואז נחלק ב-2.

אלגוריתם (*Complete* Karmarkar-Karp Heuristic – *CKK*):

כמו האלגוריתם הקודם רק עבור יותר משני תתי-קבוצות,

כאשר אלגוריתם *KK* מפריד בין שני המספרים הכי גדולים בקבוצה לשני תתי-קבוצות אז האפשרות היחידה האחרת היא להקצות אותן לאותה תת-קבוצה על ידי החלפתן בסכום שלהם, *CKK* מחפש בעץ בינארי שבו הענף השמאלי של הצומת מחליף את השניים המספרים הגדולים ביותר לפי ההפרש שלהם, והענף הימני ממקם אותם לפי הסכום שלהם. אם המספר הגדול ביותר שווה או עולה על סכום המספרים הנותרים, כולם ממוקמים בתת-קבוצה נפרדת. הפתרון הראשון שנמצא הוא פתרון *KK*, אבל *CKK* בסופו של דבר מוצא ומאמת פתרון אופטימלי.

בעיית סכום של תת-קבוצה:

הבעיה היא למצוא תת-קבוצה של קבוצה נתונה של מספרים שהסכום שלהם הוא הקרוב ביותר לערך יעד נתון. חלוקה דו-כיוונית שווה ערך לבעיית סכום משנה שבו ערך היעד הוא חצי מהסכום של כל המספרים.

נציג שני אלגוריתמים לבעיית סכום המשנה:

אלגוריתם (Horowitz and Sahni - *HS*):

1. ראשית מחלק את n המספרים לשתי חצאי קבוצות של גודל $\frac{n}{2}$.
2. מג'נרט את כל האפשרויות של תתי הקבוצות הללו (כולל הסט הריק).
3. ממיין אותן בסדר עולה לפי סכומיהם.
4. כל תת-קבוצה של המספרים המקוריים חייבת להיות מורכבת מתת-קבוצת מהמחצית הראשונה בתוספת קבוצת משנה מהמחצית השנייה.
5. כעת האלגוריתם מכיל שני מצביעים, אחד לכל רשימה ממוינת של סכומי תת-הקבוצות, המצביע הראשון מתחיל בתת-קבוצה הראשונה עם הסכום הקטן ביותר והמצביע השני מתחיל בתת-קבוצה השנייה עם הסכום הגדול ביותר.
   a. כעת נחבר בין שני הסכומים של שני המצביעים, אם הסכום שווה או גדול מהערך החיפוש מסתיים.
      i. אם הסכום הוא קטן מהערך, המצביע הראשון סכומו גדול מהערך.
      ii. אם הסכום גדול מהמטרה, המצביע השני מוקטן במיקום אחד.
   b. האלגוריתם מגדיל את המצביע הראשון ומקטין את השני עד שערך מטרה שלנו נמצא.

זמן הריצה של האלגוריתם הוא $O(n \cdot 2^{n/2})$.

גודל הזיכרון שנשמר עבור 2 הרשימות הוא $2^{\frac{n}{2}}$.

אלגוריתם (Schroeppel and Shamir - SS):

אלגוריתם זה משתמש בפחות זיכרון מאשר *HS*.

האלגוריתם הקודם (*HS*) ניגש לשני התתי-קבוצות (רשימות) בסדר ממוין, במקום למיין את הרשימות האלו מראש, *SS* קודם ממיין אותם ומסדר אותם לפי סדרי גודל, כדי לייצר את כל סכומי התת-קבוצות מהרשימה החצי הראשונה ב-*SS* מחלק את הרשימה לשתי רשימות *a* ו-*b*, כל אחת בגודל $\frac{n}{4}$, יוצר את כל קבוצות המשנה מכל רשימה, וממיין אותן בסדר עולה של הסכומים שלהן. לאחר מכן, נבנית ערימה מינימלית של זוגות של תת-קבוצות מ-*a* ו-*b*. בתחילה, הערימה כוללת את כל הזוגות של מצורה $(a1, bj)$, עם (*a1, b1*) למעלה. מכיוון שיש צורך בכל תת קבוצה מהרשימה המשולבת של *a* ו-*b*, היא מוסרת מראש הערימה. אלמנט זה (ai, bj) מוחלף ב הערימה עם הזוג $(ai + 1, bj)$. זה יוצר את כל הזוגות של קבוצות משנה מ-*a* ו-*b* בסדר עולה של הסכום הכולל שלהן.

באופן דומה, כל קבוצות המשנה מרשימת המחצית השנייה של המספרים נוצרים בסדר יורד של הסכום שלהם תוך שימוש בערימה מקסימלית (ההפך ממינימום). לאחר מכן אנו משתמשים בסכימת *HS* כדי ליצור את תת-הקבוצה שהסכום שלו הכי קרוב לערך היעד.

זמן הריצה של האלגוריתם הוא $O(n \cdot 2^{n/2})$.

והזיכרון הנדרש הוא $O(2n/4)$.

אלגוריתם (Recursive Number Partitioning - *RNP*):

זהו האלגוריתם הטוב ביותר הקודם לחלוקה רב-כיוונית (מספר קבוצות).

1. מפעיל את היוריסטיקה של *KK* כדי ליצור פתרון ראשוני.
   a. אם *k* הוא אי-זוגי, אז ניצור את כל תת-הקבוצות הראשונות שיכולות להיות חלק מ-*k* קבוצות, חלוקה זו טובה יותר מהטובה הנוכחית. עבור כל אחד, זה בצורה אופטימלית מחלק את המספרים הנותרים *k − 1* בדרכים.
   b. אם *k* זוגי, זה מייצר את כל המחיצות הדו-כיווניות האפשריות של המספרים, כך שחלוקה של כל תת-קבוצה ב-*k/2* דרכים עשויה להיגרם בחלוקת *k* טובה יותר מהטובה הנוכחית.
2. לאחר מכן הוא מחלק באופן אופטימלי את תת-הקבוצה הראשונה ב-$\frac{k}{2}$ דרכים, ואם התוצאה היא סכום מקסימלי של תת-קבוצה שקטן מזה של הטוב ביותר הנוכחי, כך שהוא מחלק בצורה אופטימלית את תת-הקבוצה השנייה *k/2* דרכים.
3. כדי ליצור את תת-קבוצות הראשונות עבור *k* אי זוגי, או לחלק את מספרים בשתי דרכים עבור *k* זוגי, *RNP* ממיין את המספרים בסדר יורד, ולאחר מכן מחפש עץ דו-נארי של הכללה-אי-הכללה. כל רמה בעץ מתאימה לרמה אחרת מספר, ובכל צומת סניף אחד כולל את המספר בקבוצת המשנה, והענף השני אינו כולל אותה. גיזום מתבצע כדי לחסל תת-קבוצות שהסכום שלהן ייפול מחוץ להן גבולות מוגדרים, שיידונו להלן. עבור חלוקה דו-כיוונית אופטימלית, *RNP* משתמש באלגוריתם *CKK*.

שיפור אלגוריתם קודם (Improved Recursive Number Partitioning):

כעת אנו מציגים את האלגוריתם החדש שלנו, שאנו קוראים לו חלוקת מספרים רקורסיבית משופרת (*IRNP*). אלגוריתם זה דורש רשימה של n מספרים שלמים, ומחזיר חלוקה אופטימלית ל-k תת-קבוצות. כמו *RNP* ו-*IRNP* הוא אלגוריתם הסתעפות וחיבור בכל עת תחילה מחשב את קירוב *KK*, ואז ממשיך למצוא פתרונות טובים יותר עד שימצא ויוודא פתרון אופטימלי.

אם הוא מוצא חלוקה מושלמת, או כזו שתת-הקבוצה המקסימלית שלה סכום שווה למספר הגדול ביותר, הוא מחזיר אותו מיד.

האלגוריתם החמדן הוא אופטימלי עבור $n \leq k + 2$:

קל להראות שאם $n \leq k + 2$ אז האלגוריתם החמדן מחזיר תוצאה אופטימלית, אבל עבור n כללי ייתן ולא יהיה אופטימלי באלגוריתם חמדן וגם באלגוריתם *KK*.

חלוקת מספרים של קבוצות קטנות:

בשביל לקבל חלוקה אופטימלית, כאשר $n \geq 5$ נשתמש באלגוריתם של חלוקה של שתי קבוצות (כמו שראינו לעיל), כאשר $n \leq 16$ אנחנו יכולים להשתמש ב-*CKK* וכאשר $n > 16$ נוכל להשתמש ב-*SS*, מכיוון ש-*CKK* רץ בסיבוכיות של $O(2^n)$, ו-*SS* רץ בסיבוכיות של $O(n \cdot 2^{\frac{n}{2}})$, אבל התקרה הקבוע של *SS* גדולה בהרבה משל *CKK*.

עבור חלוקה של יותר מ-2 קבוצות *CKK* פחות יעיל מאשר *CGA*, עבור $k > 2$ וערכים קטנים כאשר $n > k + 2$ נוכל להשתמש ב-*CGA* על מנת לחשב חלוקה אופטימלית, טבלה מספר 1 מראה את הערכים של n עבור *CKK* ($n = 2$) או *CGA* ($n > 2$) מהירים יותר מאשר *IRNP* (נקבע בניסוי).

טבילה מספר 1 (מראה עבור איזה n גדול *CKK* ו-*CGA* מהירים יותר):

k	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
n	*16*	*12*	*14*	*16*	*19*	*21*	*25*	*27*	*31*

עקרון רקורסיבי של אופטימליות:

עבור ערכים גדולים יותר של n, האלגוריתם שלנו מסתמך על עיקרון של אופטימליות, אותה אנו מגדירים ומוכיחים כאן.

ראשית, אנו מגדירים פירוק של חלוקה כחלוקה דו-כיוונית של תת-קבוצות של החלוקה (בצורה רקורסיבית).

<u>מקרה פרטי:</u>

לדוגמה, יש שלושה פירוקים שונים של חלוקה תלת-כיוונית, אחת עבור כל חלוקה לקבוצה אחת לעומת שתי הקבוצות האחרות. באופן דומה, יש שבעה פירוקים שונים של מחיצה ארבע כיוונית, ארבעה מהם מפרקים אותו לסט אחד לעומת שלושת האחרים, ושלושה מהם מפרקים אותו לשתי סטים לעומת שני סטים.

<u>מקרה כללי:</u>

בהינתן הגדרה זו, ופונקציה אובייקטיבית הממזערת את סכום תת-הקבוצה הגדול ביותר בחלוקה, אנו יכולים לקבוע ולהוכיח את המשפט הבא: בהינתן כל מחיצת עבור k אופטימלית, וכל פירוק של המחיצה הזו לקבוצות $k1$ ו-$k2$ קובע כך ש- $k1 + k2 = k$, ואז כל מחיצה אופטימלית של המספרים בקבוצות $k1$ לתוך קבוצות $k1$, בשילוב עם כל אחד חלוקה אופטימלית של המספרים בערכות $k2$ לקבוצות $k2$, מביא לחלוקה עבור k אופטימלית.

ההוכחה למשפט זה היא פשוטה מאוד: בהינתן אוסף של תת-קבוצות $k1$, חלוקה אופטימלית של המספרים באלו תת-ערכות $k1$ דרכים יכולות להשאיר רק את סכום המשנה המקסימלי שלהן זהה, או להפחית אותו, אבל זה לא יכול להגדיל אותו. לפיכך, בהינתן חלוקה עבור k אופטימלית, חלוקה מיטבית של המספרים בתתי קבוצות $k1$ ותת קבוצות $k2$ בדרכי $k1$ ובדרכים $k2$, בהתאמה, לא יכול להגדיל את סכום המשנה המקסימלי. לכן, גם חלוקה עובר k החדשה חייבת להיות אופטימלית.

ערכו של עיקרון זה, למשל, הוא שניתן לכל חלוקה דו-כיוונית של כל המספרים, נוכל לחשב את החלוקה הארבע-כיוונית הטובה ביותר שמכבדת את המחיצה הדו-כיוונית על ידי חלוקת משנה אופטימלית של כל אחת משתי תתי-הקבוצות בשתי דרכים.

יצירה (ג'ינרוט) של סכומי תתי הקבוצות בהינתן טווח:

עבור חלוקה תלת כיוונית של יותר משנים עשר מספרים, כמו *RNP, IRNP* נייצר את כל תתי-הקבוצות שיכולות להיות חלוקה האפשרית הטובה ביותר מהחלוקה הנוכחית, ולכל אחת מהן נחלק את המספרים הנותרים בשתי דרכים. זה דורש יצירה של כל תת-הקבוצות עם סכומים בטווח נתון, הגבולות מהם יתואר להלן.

בעוד *RNP* מייצר תת-קבוצות אלה על ידי חיפוש בעץ בינארי של הכללה-אי-הכללה, *IRNP* משתמש באלגוריתם חדש שהוא הרחבה של אלגוריתם *SS*. הרחבה זו נחוצה מכיוון ש-*SS* מחזיר רק תת-קבוצה בודדת שהסכום שלה הוא הקרוב ביותר ליעד שצוין (ערך). לשם הפשטות, אנו מתארים תחילה את ההרחבה שלנו ל-*HS*.

הרחבה עבור אלגוריתם (Horowitz and Sahni - *HS*):

באלגוריתם *HS* נשמרים שני מצביעים, מצביע ראשון לרשימה הראשונה של תת-הקבוצות ומצביע שני לרשימה השנייה. באלגוריתם ה-*HS* המורחב שלנו (*EHS (E = Extend)*), אנו מחליפים את המצביע השני עם שני מצביעים לרשימה השנייה, שאנו קוראים לה נמוך וגבוה. בכל זמן נתון, נמוך הוא המדד הגדול ביותר כזה סכום תת-הקבוצות שעליהן הצביעו הראשון והנמוך קטן מ- או שווה לגבול התחתון.

באופן דומה, גבוה הוא המדד הקטן ביותר, כך שסכום תת-הקבוצות הצביע עליו קודם וגבוה גדול מהגבול העליון או שווה לו. הראשון הוא גדל, ונמוך וגבוה מופחתים כדי לשמור על משתנה הזה, ובכך יוצר את כל תת-הקבוצות שהסכומים שלהן בין הגבולות התחתונים והעליון.

הרחבה עבור אלגוריתם (Schroeppel and Shamir - SS):

הרחבת אלגוריתם *SS* ליצירת כל תת-קבוצות שהסכומים שלהן נמצאים בין הטווח נתון הוא קצת יותר מורכב. החלק העליון של הערימה מתאימה לתת-קבוצת עליה מצביע המצביע הראשון ב-*HS*, והחלק העליון של הערימה המקסימלית מתאים לתת-הקבוצה שמצביע עליו המצביע השני. כדי להרחיב את זה לטווח של קבוצות משנה, אנו מאחסנים במפורש ברשימה נפרדת את תת-הקבוצות הללו שהסכומים שלהם הם בין שני המצביעים שהצביעו עליהם הנמוך לגבוה (המצביעים ב-*EHS*).

תת-הקבוצה הראשונה ברשימה זו מתאימה לתת-קבוצה שמצביע עליה המצביע הנמוך ב-*EHS*, ותת-הקבוצה האחרונה תואמת לתת-הקבוצה שעליה מצביע המצביע הגבוה ב-*EHS*. בכל זמן נתון, תת-הקבוצות שהסכומים שלהן נמצאים בין הגבולות הם קבוצות המשנה ברשימה זו, בשילוב עם תת-הקבוצה בראש הערימה המינימלית. כשאנחנו מקפיצים קבוצות משנה מהערימה מינימלית, המתאימה להגדלת המצביע הראשון פנימה *EHS*, אנו מוסיפים תת-קבוצות עם סכומים קטנים יותר לרשימה על ידי הוצאה שלהם מהערימה המקסימלית, בהתאמה להקטנת המצביע נמוך, והסרת תת-קבוצות מהרשימה שהסכומים שלהן עולים על הגבול העליון בשילוב עם זה שמעל הרשימה ערימה מינימלית, המקבילה להורדת המצביע הגבוה.

רשימה זו של תת-הקבוצות מיושמת כמאגר מעגלי עם מצביעים לראש ולזנב, בהתאמה לנמוכים ונקודות גבוהות של *EHS*. שינוי זה מגדיל את החלל נחוץ ל-*SS*, אבל בכל הניסויים שלנו, רשימה של מאה אלף תת-קבוצות היו די והותר.

גבולות הדוקים של סכומי תתי-הקבוצות:

ESS מייצר את כל תת-הקבוצות שהסכומים שלהן נמצאים במסגרת הגבולות הנתונות. לשם הפשטות אנו מתארים את הגבולות הללו עבור חלוקה תלת-כיוונית, אך הם מכלילים בצורה פשוטה לחלוקה עבור k קבוצות.

נגדיר את m להיות סכום המשנה המקסימלי בפתרון הטוב ביותר הנוכחי, ונגדיר את s להיות הסכום של כל המספרים. כל סכום משנה חייב להיות קטן מ-m, וכך $m-1$ הוא גבול עליון לכל תת-הקבוצות.

סכום תת-הקבוצה הראשון חייב להיות גם מספיק גדול כדי שהמספרים הנותרים יכולים להיות מחולק לשתי תת-קבוצות עם שני הסכומים הנמוכים מ-m.

לפיכך, $s-2(m-1)$ הוא גבול תחתון על תת-הקבוצה הראשונה, אנו יכולים להדק את הגבולות הללו על ידי ביטול חלקים כפולים שנבדלים רק על ידי תמורה של תת-הקבוצות. בכל תת-קבוצה תלת כיוונית, חייבת להיות לפחות תת קבוצה אחת שלה עם סכום קטן או שווה ל-$\left\lfloor\frac{n}{3}\right\rfloor$. לפיכך, *RNP* משתמש בגבול תחתון של $s-2(m-1)$, וגבול עליון של $\left\lfloor\frac{n}{3}\right\rfloor$ עבור תת-קבוצה ראשונה במחיצה תלת כיוונית.

IRNP מהדק את הגבולות הללו עוד יותר. בכל חלוקה של 3 תתי-קבוצה חייבת להיות סכום תת-קבוצה אחד לפחות גדול מ- או שווה ל-$\left\lceil\frac{n}{3}\right\rceil$, משתמש בגבול תחתון של $\left\lceil\frac{n}{3}\right\rceil$ והגבול העליון של $m-1$ בתת-הקבוצה הראשונה. עבור חלוקה תלת כיוונית, ההבדל בין הגבול התחתון והעליון של *IRNP* הוא חצי מזה של *RNP*, מה שמוביל ליצירה של בערך חצי ממספר תת-הקבוצות הראשונים מאשר *RNP*.