שלשות פיתגורס ו־SAT solving

מוטי בן־ארי

המחלקה להוראת המדעים

מכון ויצמן למדע

http://www.weizmann.ac.il/sci-tea/benari/

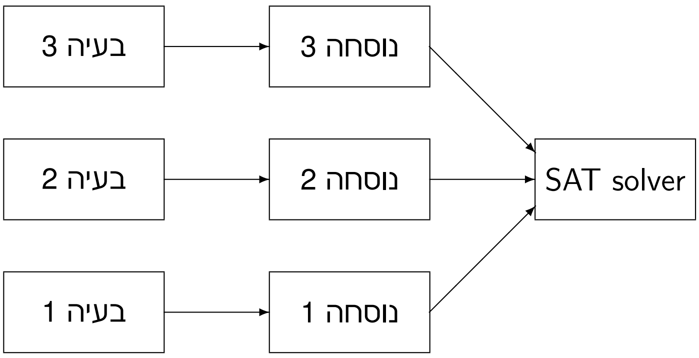
© 2018 by Moti Ben-Ari.

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit `http://creativecommons.org/licenses/by-sa/3.0/` or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

מסמך זה מניח היכרות עם תחשיב הפסוקים בלוגיקה ועם NP-completeness. סקירה של נושאים אלה נמצאת בנספחים.

1 מבוא

SAT solving היא שיטה לפתרון בעיות על ידי תרגומן לנוסחאות בתחשיב הפסוקים, ואז חיפוש הצבה שמספקת את הנוסחה. היתרון של SAT solving היא שניתן להשתמש בתוכנית יעילה אחת כדי לפתור בעיות רבות.



במסמך זה אביא סקירה של SAT solving ותיאור איך Heule ו־Kullman השתמשו בשיטה כדי לפתור בעיה במטימקטיה פתוחה.

Marijn J.H. Heule and Oliver Kullmann. The Science of Brute Force. *Communications of the ACM* 60(8), 2017, 70–79.

2 אלגוריתם DPLL עבור SAT

אין טעם להיכנס לדיכאון כאשר מגלים שבעיה היא NP-complete ולכן כמעט בטוח שאין לה אלגוריתם יעיל. המשמעות של "אלגוריתם יעיל" היא שהאלגוריתם יעיל על **כל** המקרים של הבעיה. בפועל אפשר להסתפק בדרישה פחות קשוחה: נהיה מרוצים אם יש לנו אלגוריתם ל־SAT שיעיל עבור **רוב** או **הרבה** נוסחאות ב־CNF.

בסעיף זה נציג את האלגוריתם DPLL שפתוח בשנות 1960-1962 על ידי Martin Davis, Hilary Putman, George Logemann, Donald Loveland. אלגוריתם DPLL הוא הבסיס של רוב ה־SAT solvers המודרניים. ניתן להוכיח שהאלגוריתם אינו יעיל כי יש משפחה של נוסחאות שהוא אינו מסוגל לפתור בזמן פולינומיאלי, אבל הניסיון מראה שהאלגוריתם יעיל מאוד בנוסחאות רבות.

האלגוריתם DPLL מורכב משני צעדים שמבצעים שוב ושוב בלולאה או ברקורסיה.

- **החלטה** בחר אטום שטרם קיבל הצבה והצב בו T או F.
- **הפצת יחידות** (**Unit propagation**) פשט את הנוסחה באמצעות **יחידות**: פסקאות של ליטרל אחד בלבד.

אם כתוצאה מביצוע אחד מהצעדים האלה מתקבלת סתירה, הפסקה שגרמה לסתירה נקראת **פסקת התנגשות** (**conflict clause**). אם נוצר פסקת התנגשות, חזור אחורה בחישוב ונסה החלטה אחרת: הצבת F במקום T עבור אטום זה או הצבה לאטום אחר. אם כתוצאה מביצוע הצעדים הנוסחה מקבלת ערך אמת T מצאנו הצבה שמספקת את הנוסחה.[1] אם בדקנו את כל ההצבות האפשריות ולא מצאנו הצבה המספקת את הנוסחה, הנוסחה אינה ספיקה.

נפעיל את האלגוריתם DPLL על הנוסחה A:

$$A = (\neg\, p \vee q \vee r) \;\wedge\; (\neg\, q \vee r) \;\wedge\; (\neg\, p \vee \neg\, r) \wedge (p \vee q)$$

חשוב להבין שהספיקות של נוסחה לא מושפע ממחיקה של פסקה הכוללת ליטרל שערכו T, וגם לא מושפעת ממחיקה מפסקה של ליטרל שערכו F.

- נחליט להציב T עבור p. ניתן למחוק את הפסקה הרביעית $p \vee q$, וגם את הליטרל $\neg\, p$ מהפסקה הראשונה ומהפסקה השלישית. התוצאה היא:

$$A' = (q \vee r) \;\wedge\; (\neg\, q \vee r) \;\wedge\; (\neg\, r)\,.$$

- נבצע הפצת יחידות. הנוסחה A' יכול לקבל ערך אמת T רק אם הפסקה הרביעית $\neg\, r$ (שהיא פסקה יחידה) תקבל ערך אמת T, וזה קורה רק אם מציבים את הערך F באטום r. כעת ניתן למחוק את הליטרל r משתי הפסקאות הראשונות של A'. התוצאה היא:

$$A'' = q \wedge \neg\, q\,.$$

- כל החלטה, להציב T או F ב־q, גורמת לערך האמת של A'' להיות F. מכאן, שההצבות $\{p = T, q = F, r = F\}$ ו־$\{p = T, q = T, r = F\}$ אינן יכולה לספק את A.
- הצבה $\{p = T, q = F, r = F\}$ גורמת לפסקה $\neg\, p \vee q \vee r$ להיות פסקת התנגשות.

3 אלגוריתם CDCL עבור SAT

הבעיה עם DPLL היא שהוא מנסה את כל ההצבות האפשרויות ללא אבחנה. עבור הנוסחה A, ראינו מטבלת האמת שהצבה ספיקה מתקבלת רק אחרי שבודקים את כל ההצבות האחרות.

כאשר הפעלנו את DPLL על A, ברגע שהצבנו $\{p = T, r = F\}$, אין כל חשיבות לערך שנציב ב־q. ב־1996, João P. Marques Silva ו־Karem A. Sakallah הראו שכאשר פסקת התנגשות נמצאת, ניתן למצוא את הסיבה להתנגשות ולבטא את הסיבה בפסקה חדשה. אפשר להוסיף את **הפסקה הנלמדת** לנוסחה המקורית ב־CNF בלי לשנות אם הנוסחה היא ספיקה או לא. זה מאפשר לאלגוריתם ב־DPLL לבדוק פחות הצבות כי תמצא פסקת התנגשות חדשה עם הצבה

[1] ההצבה יכולה להיות חלקית, אבל ערך האמת של הנוסחה יהיה T עובר כל הרחבה של ההצבה להצבה מליאה.

חלקית קטנה. השיטה נקראת **למידת פסקאות מונחת התנגשות** והרבה SAT solvers מודרניים משתמשים בה. ראו:

J. P. Marques-Silva, I. Lynce, S. Malik. *Conflict-Driven Clause Learning SAT Solvers*, 131–153, in A. Biere, M. Heule, H. Van Maaren, T. Walsh (eds.), *Handbook of Satisfiability*, IOS Press, 2009.

LearnSAT היא SAT solver שפיתחתי שמאפשרת למשתמש לעקוב אחרי הפעולות של התכנית. התכנה כוללת מדריך ל־SAT solving ודומאות רבות שניתנות לפתרון בעזרת LearnSAT.

M. Ben-Ari. (2018). LearnSAT: A SAT Solver for Education. *Journal of Open Source Software*, 3(24), 639, https://doi.org/10.21105/joss.00639

4 שלשות שור

סעיף זה מדגים פתרון של בעיה מתמטית על ידי SAT solver.

> **שלשת שור (Schur triple)** נתונה חלוקה **כלשהי** של המספרים הטבעיים $S = \{1,\ldots,n\}$ לשתי תת־קבוצות זרות S_1, S_2, האם קיימים שלושה מספרים $a,b,c \in S_i$, כך ש:
>
> $$a = b + c\,,$$
>
> עבור לפחות אחת מ־S_1, S_2?

דוגמה

עבור $n = 8$ והחלוקה:

$$S_1 = \{1,2,3,4\}, S_2 = \{5,6,7,8\}\,,$$

קיימת שלשת שור $\{3,2,1\}$. עבור החלוקה

$$S_1' = \{1,2,4,8\}, S_2' = \{3,5,6,7\}\,,$$

אין שלשת שור.

עבור $n = 8$, קיימות חלוקות המכילות שלשת שור וחלוקות אחרות שלא מכילות שלשת שור.

מה עם $n = 9$?

משפט

עבור **כל** חלוקה של $S = \{1,\ldots,9\}$ לשתי תת־קבוצות זרות, קיימת שלשת שור בתת־קבוצה אחת לפחות.

הוכחה

פשוט מאוד, בדוק כל אחת מ־$2^9 = 512$ החלוקות.

ברור שזו משימה מייגעת ביותר. ננסה למצוא דרך קלה יותר.

הוכחה

כדי להוכיח את המשפט ננסה להוכיח את שלילתו ולמצוא סתירה, כלומר, ננסה למצוא חלוקה של $\{1,\ldots,9\}$ **שאינה** מכילה שלשת שור.

תחילה נבדוק אם 1 ו־3 יכולים להיות באותה תת־קבוצה, נניח, S_1. אם כן, 2 לא יכול להיות ב־S_1 כי $1+2=3$, ולכן הוא חייב להיות ב־S_2. באופן דומה, $1+3=4$ חייב להיות ב־S_2. אם נמשיך, נגלה ש־9 חייב להיות בשתי תת־הקבוצות!

S_1	S_2
$1,3$	
$1,3$	$2,4$
$1,3,6$	$2,4,7,9$
$1,3,6,9*$	$2,4,7,9*$

מכאן, שכל חלוקה המשימה את 1 ו־3 באותה תת־קבוצה לא יכולה להכיל שלשת שור.

ננסה עכשיו למצוא חלוקה עם שלשת שור כאשר 1 ו־3 נמצאים בתת־קבוצות שונות. אי אפשר להסיק משהו כאשר יש מספר אצד בכל תת־קבוצה, אז נקח החלטה נוספת: 5 נמצא ב־S_2. 2 ו־8 חייבים להיות ב־S_1, כי $2+3=5$ ו־$3+5=8$. נמשיך ונגלה שוב ש־9 חייב להיות בשתי התת־קבוצות.

S_1	S_2
1	$3,5$
$1,2,8$	$3,5$
$1,2,8$	$3,5,9$

אם נשים את 4 ב־S_1, $2+4=6$ חייב להיות ב־S_2, ומכאן $3+6=9$ חייב להיות ב־S_1, שוב סתירה.

$1,2,4,8$	$3,5,6,9$
$1,2,4,8,9*$	$3,5,6,7,9*$

מאידך, אם נשים את 4 ב־S_2, $5+4=9$ חייב להיות ב־S_1, סתירה.

$1,2,8$	$3,4,5,9$
$1,2,8,9*$	$3,4,5,6,7,9*$

לבסוף, ננסה לשים את 5 ב־S_1. שוב, זה מוביל לסתירה.

S_1	S_2
$1,5$	3
$1,5$	$3,4,6$
$1,2,5,7,9$	$3,4,6$
$1,2,5,7,9*$	$3,4,6,9*$

הראנו שאין חלוקה של $\{1,\ldots,9\}$ לשתי קבוצות זרות, כך קיימת שלשה a,b,c, כך ש־$a=b+c$, וכל המספרים בשלשה לא נמצאים באותה תת־קבוצה S_1 או S_2. ללא השלילה הכפולה: לכל חלוקה של $\{1,\ldots,9\}$ לשתי קבוצות זרות, לפחות תת־קבוצה אחת תכיל a,b,c ו־$a=b+c$.

דוגמה $S_1=\{1,2,5,7\},\ S_2=\{3,4,6,8,9\}$ ו־$3+6=9$ ב־S_2.

5 הוכחת תכונות של שלשות שור באמצעות SAT solving

נראה איך SAT solver יכול למצוא חלוקה של $\{1,\ldots,8\}$ לשתי תת־קבוצות כך שאין שלשת שור באף אחת מהן. אחר כך נוכיח שאין חלוקה דומה עבור $\{1,\ldots,9\}$. נציג את המעקבים מ־LearnSAT.

קידוד הבעיה של שלשות שור

קיים אטום עבור כל מספר בסדרה. עבור $n=8$ האטומים הם:

```
x1,x2,x3,x4,x5,x6,x7,x8.
```

אם נציב T באטום המספר נמצא בתת־קבוצה הראשונה, ואם נציב F באטום, המספר נמצא בתת־קבוצה השניה.

להלן רשימה של שלשות שור האפשריות:

$$3=1+2,\ 4=3+1,\ \ldots,\ 7=3+4,\ 8=3+5\,.$$

עבור כל שלשה אפשרית יהיו שתי פסקאות:

```
[x1,x2,x3], [~x1,~x2,~x3],
[x1,x3,x4], [~x1,~x3,~x4],
[x1,x4,x5], [~x1,~x4,~x5],
[x1,x5,x6], [~x1,~x5,~x6],
[x1,x6,x7], [~x1,~x6,~x7],
[x1,x7,x8], [~x1,~x7,~x8],
[x2,x3,x5], [~x2,~x3,~x5],
[x2,x4,x6], [~x2,~x4,~x6],
[x2,x5,x7], [~x2,~x5,~x7],
[x2,x6,x8], [~x2,~x6,~x8],
[x3,x4,x7], [~x3,~x4,~x7],
[x3,x5,x8], [~x3,~x5,~x8]
```

למשל, עבור $7=3+4$, הפסקה `[x3,x4,x7]` דורשת שבלפחות אחד האטומים יוצב T, והפסקה `[~x3,~x4,~x7]` דורשת שבלפחות אחד האטומים יוצב F. כל הצבה שמספקת את שתי הפסקאות מייצגת חלוקה כך ש־$3,4,7$ **אינם** באותה תת־קבוצה, ולכן הם **לא** מהווים שלשת שור.

אם אין הצבה המספקת את הפסקאות, אין חלוקה שלא מכילה שלשת שור. ללא השלילה הכפולה: כל חלוקה מכילה שלשת שור.

מציאת חלוקה עבור $n=8$

הנה החישוב של ה־SAT solver:

```
LearnSAT v1.4.4.  Copyright 2012-13 by Moti Ben-Ari.  GNU GPL.
Decision assignment:  x1=0
Decision assignment:  x2=0
Propagate unit:  x3 (x3=1) derived from:  1.  [x1,x2,x3]
Decision assignment:  x4=0
Propagate unit:  x5 (x5=1) derived from:  5.  [x1,x4,x5]
Propagate unit:  x6 (x6=1) derived from:  15.  [x2,x4,x6]
Propagate unit:  ~x8 (x8=0) derived from:  24.  [~x3,~x5,~x8]
Propagate unit:  x7 (x7=1) derived from:  11.  [x1,x7,x8]
Satisfying assignments:
[x1=0,x2=0,x3=1,x4=0,x5=1,x6=1,x7=1,x8=0]
```

נחוץ רק שלוש החלטות: לשים את האטומים x1, x2, x4 באותה בתת־קבוצה הראשונה על ידי הצבה של F. אז, הפצת יחידות מוצא הצבה מספקת במהירות המתאימה ל־$S_1 = \{1, 2, 4, 8\}, S_2 = \{3, 5, 6, 7\}$. קל לבדוק שאין שלשת שור באף אחת משתי תת־הקבוצות.

הוכחה שאין חלוקה עבור $n = 9$

יש להוסיף לקידוד זוגות של פסקאות עבור השלשות הנוספות:

$$9 = 1 + 8,\ 9 = 2 + 7,\ 9 = 6 + 3,\ 9 = 5 + 4\,.$$

הנה החישוב של ה־SAT solver:

```
LearnSAT v1.4.4.  Copyright 2012-13 by Moti Ben-Ari.  GNU GPL.
Decision assignment:  x1=0,
Decision assignment:  x2=0
Decision assignment:  x4=0
Conflict clause:  30.  [~x3,~x6,~x9]
Decision assignment:  x4=1
Conflict clause:  28.  [~x3,~x5,~x8]
Decision assignment:  x2=1,
Decision assignment:  x3=0
Conflict clause:  20.  [~x2,~x5,~x7]
Decision assignment:  x3=1
Conflict clause:  18.  [~x2,~x4,~x6]
Decision assignment:  x1=1,
Decision assignment:  x2=0
Decision assignment:  x3=0
Conflict clause:  17.  [x2,x4,x6]
Decision assignment:  x3=1
Conflict clause:  19.  [x2,x5,x7]
Decision assignment:  x2=1,
```

```
Decision assignment:  x4=0
Conflict clause:  27.  [x3,x5,x8]
Decision assignment:  x4=1
Conflict clause:  29.  [x3,x6,x9]
Unsatisfiable
```

תחילה לקוחים שלוש החלטות כדי לשים את האטומים x1, x2, x4 באותה תת־קבוצה. לאחר הפצת יחידות, ערך האמת של [~x3,~x6,~x9] הוא F, ולכן הפסקת היא פסגת התנגשות, כלומר, אין הצבה מספקת המכיל הצבה חלקית זו. כדי לחסוך במקום, הפצת היחידות לא מוצגת, אבל ניתן לראות אותה בהרצת LearnSAT.

למידת פסקאות מונחת התנגשות (CDCL)

האלגוריתם ל־CDCL די מורכב, לכן רק אדגים אותו על הנוסחה להלן:

```
[x1, x31, ~x2], [x1, ~x3], [x2, x3, x4],
[~x4, ~x5], [x21, ~x4, ~x6], [x5, x6]
```

לאחר שלושת ההחלטות שלהן, הפצת יחידות גורמת להופעת פסקת התנגשות:

```
Decision assignment:  x21=0
Decision assignment:  x31=0
Decision assignment:  x1=0
...
Conflict clause:  [x5,x6]
```

CDCL מובילה ללמידה של פסקה חדשה שמתווספת לקבוצה המקורית של פסקאות:

```
Learned clause:  [x21,~x4]
```

לאחר שלוש החלטות נוספות נמצא הצבה מספקת:

```
Satisfying assignments:
[x21=0,x31=0,x1=1,x2=0,x3=1,x4=0,x5=0,x6=1]
```

הפעלת האלגוריתם עם CDCL מחייבת רק שש החלטות לעומת תשע החלטות עם DPLL בלבד. הפסקה שנלמדה מאפשרת הצבה מיידית של F ל־x4 כי x21 כבר קיבל הצבה של F.

```
Propagate unit:  ~x4 (x4=0) derived from:  [x21,~x4]
```

6 שלשות פיתגרוס

שלשות פיתגורס דומות לשלשות שור רק שהיחס בין המספרים הוא לפי משפט פיתגורס במקום חיבור. בנוסף ההבעיה מבקשת חלוקה של כל המספרים הטבעיים ולא רק תת־סדרה סופית.

> **שלשת פיתגורס** נתונה חלוקה **כלשהי** של המספרים הטבעיים N לשתי תת־קבוצות זרות N_1, N_2, האם קיים $a, b, c \in N_i$, כך ש:
>
> $$a^2 = b^2 + c^2\,,$$
>
> עבור לפחות אחת מ־N_1, N_2?

דוגמה נחלק את המספרים הטבעיים למספרים זוגיים ואי־זוגיים:

$$N_1 = 1, 3, 5, 7, \ldots$$
$$N_2 = 2, 4, 6, 8, \ldots$$

אין שלשות פיתגורס ב־N_1 אבל $6, 8, 10 \in N_2$ ו־$10^2 = 8^2 + 6^2$.

משפט

עבור **כל** חלוקה של N לשתי תת־קבוצות זרות, לפחות תת־קבוצה אחת מכילה שלשת פיתגורס.

קיים מספר אינסופי של חלוקות של המספר האינסופי של המספרים טבעיים, ולכן נראה שאין סיכוי להוכיח את המשפט באמצעות ייצוג סופי בנוסחה. אבל, מספיק למצוא $n \in N$ **כלשהי** כך שעבור **כל** חלוקה של $\{1, \ldots, n\}$ לשתי תת־קבוצות זרות, לפחות אחת מכילה שלשות פיתגורס. הסיבה היא שכל חלוקה של הספרים טבעיים חייב לחלק את המספרים $\{1, \ldots, n\}$ לתת־קבוצות. אם כל חלוקה של $\{1, \ldots, n\}$ מכילה שלשת פיתגורס, גם החלוקה לתת־קבוצות אינסופיים מכילה שלשת פיתגורס.

למשל, נניח שנתונה לנו חלוקה של N ל־N_1, N_2, ונניח שהוכחנו שכל חלוקה של $\{1, \ldots, 20\}$ מכילה שלשת פיתגורס (לא נכון, אבל לצורך הדוגמה נניח שזה נכון). לכן, קיים שלשת פיתגורס עבור החלוקה של $\{1, \ldots, 20\}$ שמתקבלת מהחלוקה של כל המספרים ל־ N_1, N_2 . שלשה זו היא גם שלשת פיתגורס של N_1, N_2:

$$N_1 = \{1, 3, 5, 7, 12, 15, 16, 20\} \cup \{\text{numbers in } N_1 > 20\}$$
$$N_2 = \{2, 4, 6, 8, 9, 10, 11, 13, 17, 18, 19\} \cup \{\text{numbers in } N_2 > 20\}$$

עבור כל n, ניתן לקודד את הקיום של שלשת פיתגורס בדיוק כמו שעשינו עבור שלשות שור:

```
[x6,x8,x10], [~x6,~x8,~x10]
```

אם באמת קיימת חלוקה ללא שלשת פיתגורס באף אחת מהתת־קבוצות, נמצא אותה בקלות כפי מצאנו חלוקה ללא שלשת שור עבור $n = 8$.

משפט

לכל $n \leq 7824$, קיימת חלוקה של $\{1, \ldots, n\}$ לשתי תת־קבוצות זרות, כך **שאין** שלשת פיתגורס באף אחת מהקבוצות.

Heule ו־Kullman הוכיחו משפט זה באמצעות SAT solver שהתבצע במשך דקה בלבד על המחשב.

אחר כך הם הוכיחו:

משפט

לכל חלוקה של $\{1, \ldots, 7825\}$ לשתי תת־קבוצות זרות, קיימת שלשת פיתגורס בתת־קבוצה אחת לפחות.

משפט זה קשה הרבה יותר להוכיח מהמשפט הקודם, כי קיימות 2^{7825} חלוקות שיש לבדוק כדי לוודא שיש שלשה באחת מהתת־קבוצות.

כדי לקבל תחושה על המספרים שיש לעבוד איתם, נזכור שהראנו שאין חלוקה של $n = 9$ ללא שלשת שור, כי $2, 7$ חייבים להיות בתת־קבוצה אחת, בזמן ש־$3, 6$ חייבים להיות בתת־קבוצה השניה. מכאן, אנו מקבלים סתירה כי $9 = 2 + 7 = 3 + 6$.

Heule ו־Kullman מצאו ש־$5180, 5865$ חייבים להיות בתת־קבוצה אחת, ו־$625, 7800$ חייבים להיות בתת־קבוצה השניה. מכאן, אנו מקבלים סתירה כי:

$$\begin{aligned} 5180^2 + 5865^2 &= 7825^2 \\ 625^2 + 7800^2 &= 7825^2 \,, \end{aligned}$$

אני בטוח שאתם זוכרים משוואות אלה מבי"ס תיכון!

יצירת כל חלוקה אפשרית ובדיקתה יקח בערך 10^{600} שנים. ההערכה היא שגיל היקום הוא 10^{10} שנים בלבד, כך שאין להעלות על הדעת פתרון באמצעות חישוב ישיר.

Heule ו־Kullman השתמשו ב־SAT solver מתקדם, והצליחו להוכיח את המשפט ב־$35,000$ שעות זמן מחשב "בלבד". החישוב התבצע על מחשב עם 800 ליבות שעבדו במקביל וארך יומיים בלבד.

7 האם אפשר לסמוך על הוכחה שהתקבלה ממחשב?

ברוב תכניות המחשב יהיו באגים. אם כן, איך אפשר לסמוך על הוכחה מתמטית שנוצרה על ידי מחשב? התלבטות זו עלתה לראשונה עם ההוכחה ב־1976 של הבעיה לצבוע מפה עם ארבעה צבעים. ההוכחה הראתה שאם המשפט נכון אז יש קבוצה של 1936 מפות שיש להן תכונה מסויימת. בדיקה זו התבצעה בהצלחה באמצעות מחשב.

ההוכחה של המשפט על שלשות פיתגורס השתמשה בשיטה חדשה כדי שנהיה בטוח יותר שההוכחה נכונה. ה־SAT solver כתב מעקב של נוסחאות, כך שאם יש לנוסחאות תכונה מסויימת, ההוכחה נכונה. תכונה זו נבדקה על ידי תכנית יחסית פשוטה וקצרה, ותכנית זו הוכחה בשיטות מתמטיות. בכל זאת, מדובר במאמץ ניכר כי נדרשו $200,000$ גיגבייט כדי לשמור את המעקב.

8 שיעורי בית

הנה הרחבה של בעיית שלשות פיתגורס:

> נתונה חלוקה **כלשהי** של המספרים הטבעיים N **לשלוש** תת־קבוצות זרות N_1, N_2, N_3, האם קיים $a, b, c \in N_i$, כך ש:
>
> $$a^2 = b^2 + c^2 \,,$$
>
> בלפחות אחת מ־N_1, N_2, N_3?

נקבל פתרון עם קיים n כך שלכל חלוקה של $\{1, \ldots, n\}$ לשלוש תת־קבוצות זרות, קיימת שלשת פיתגורס בלפחות תת־קבוצה אחת.

ייתכן שהתשובה לא תימצא לעולם, כי n, **אם הוא קיים**, גדול מ־10^7.

נספחים

א' Satisfiability בתחשיב הפסוקים

נוסחאות בתחשיב הפסוקים מורכבות מ־**טענות אטומיות** או **אטומים**, ופעולות. נשתמש בשלוש פעולות: פעולה על נוסחה אחת $\neg$ (שלילה), ושתי פעולות על שתי נוסחאות $\wedge$ (וגם) $\vee$ (או). **ליטרל** (literal) הוא אטום או שלילה של אטום.

נשתמש בנוסחאות מהצורה conjunctive normal form (CNF) המורכבות מפסקאות המחוברות ב־"וגם", כאשר כל פסקה מורכבת המחוברים ב־"או". הנה נוסחאות נכונה תחבירית ב־CNF:

$$A = (\neg p \vee q \vee r) \;\wedge\; (\neg q \vee r) \;\wedge\; (\neg r)\,.$$

בגלל שמיקמן של הפעולות בנוסחה ב־CNF ידועה, לעתים נשמתמש בסימון של קבוצות:

$$A = \{\{\neg p, q, r\},\; \{\neg q, r\},\; \{\neg r\}\}\,.$$

הסמנטיקה של נוסחה בתחשיב הפסוקים מתקבלת על ידי הצבה של ערכי האמת $\{T, F\}$ לאטומים, ואז חישוב ערך האמת של הנוסחה. אם קיימת הצבה כך שערך האמת של הנוסחה הוא T, הנוסחה היא **ספיקה** (satisfiable).

ערך האמת של נוסחה מחושב מערכי האמת של האטומים וזה לפי ההגדרות שלהלן:

p	q	$\neg p$	$p \vee q$	$p \wedge q$
T	T	F	T	T
T	F	F	T	F
F	T	T	T	F
F	F	T	F	F

דרך אחת להחליט אם נוסחה היא ספיקה היא לבנות **טבלת אמת**, שיש בה שורה לכל הצבה אפשרית לאטומים, ובטור האחרון ערך האמת של הנוסחה עבור אותה הצבה. הנה טבלת האמת עבור הנוסחה A:

p	q	r	A
T	T	T	F
T	T	F	F
T	F	T	F
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	T

ההצבה בשורה האחרונה גורמת ל־A לקבל את ערך האמת T, ולכן A היא ספיקה.

בעיית SAT היא למצוא אלגוריתם הקולט נוסחה בצורת CNF ומחזירה הצבה עבורה הנוסחה ספיקה, או להודיע שאין הצבה מתאימה והנוסחה אינה ספיקה. תכנית המיישמת את האלגוריתם נקראת SAT solver.

בניית טבלאות אמת היא אלגוריתם ל־SAT כך שקיים לפחות אלגוריתם אחד לבעיה, אבל טבלאות אמת הן מאוד לא יעילות כי יש 2^n שורות, כאשר n הוא מספר האטומים בנוסחה.

פרטים נוספים על תחשיב הפסוקים ועל SAT solving ניתן למצוא בפרקים $2, 4, 6$ של הספר:

M. Ben-Ari. *Mathematical Logic for Computer Science (Third Edition)*, Springer, 2012.

ב' SAT היא NP-complete

בעיה Q היא NP-complete אם:

- ניתן לבדוק בזמן פולינומיאלי אם פתרון מוצע ל־Q נכון.
- ניתן לתרגם את כל הבעיות במשפחה לבעיות מסוג Q, כך שאם יש ל־Q אלגוריתם פלינומיאלי, אזי לכל הבעיות במשפחה יש אלגוריתם פולינומיאלי.

נתונה הצבה עבור נוסחה A בצורה CNF, קל לבדוק עם ערך האמת של A הוא T, ולכן בעיית SAT מקיימת את התנאי הראשון. בעיית SAT היא NP-complete כי היא מקיימת את התנאי השני, כפי שהוכח על ידי Stephen Cook (1971), Leonid Levin (1973).

התנאי הראשון שקול לטענה שבעיה במשפחת NP-complete ניתנת לפתרון בזמן פולינומיאלי על ידי אלגוריתם *non-deterministic*. השאלה אם בעיות NP-complete ניתנות לפתרון בזמן פולינומיאלי על ידי אלגוריתם deterministic ידועה בשם $\mathcal{P} = \mathcal{NP}$?.

אם קיים אלגוריתם יעיל (המתבצע בזמן פולינומיאלי) בחישוב deterministic לבעיה במשפחת NP-complete, אז קיים אלגוריתם יעיל לכל הבעיות במשפחה. נכון להיום, לא ידוע אם קיים אלגוריתם יעיל לאף אחת מבעיות.

אם ניתן להוכיח **שאין** אלגוריתם יעיל לאחת מהבעיות במשפחת NP-complete, אז אין אלגוריתם יעיל לכל הבעיות במשפחה. נכון להיום, אין הוכחה שאין אלגוריתם יעיל לאחת מהבעיות במשפחה.

ה־Clay Mathematics Institute מציע פרס של מיליון דולר למי שיצליח למצוא תשובה ל־$\mathcal{P} = \mathcal{NP}$?

`http://claymath.org/millennium-problems/p-vs-np-problem`.

NP-completeness מוצגת בספרי לימוד בתיאוריה של מדעי המחשב כגון:

Hopcroft, J.E, Motwani, R., Ullman, J.D. *Introduction to Automata Theory, Languages, and Computation*, Third edition, Addison-Wesley, 2006.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. *Introduction to Algorithms*, Second edition, MIT Press, 2001.

Sipser, M. *Introduction to the Theory of Computation.* PWS Publishing, 1997.