One Dollar Each Eliminates Envy

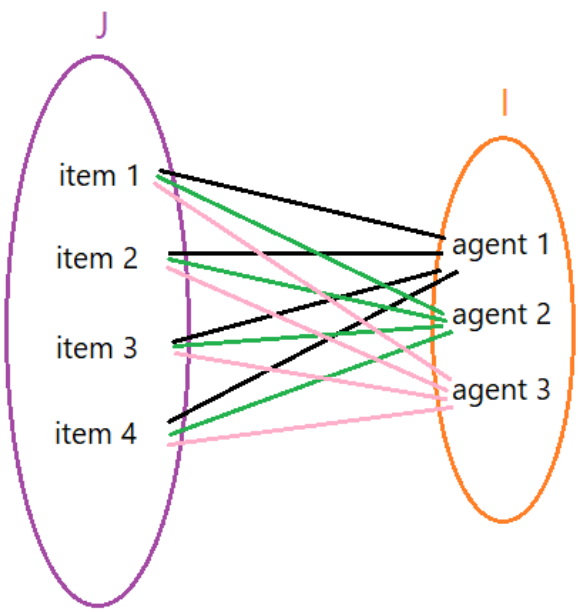
$Algorithm$ 1 in this paper construct an $EF1$ allocation with the following Setting:

- Agents have additive valuations
- The items are all goods
- The set of the agents is I s.t. $|I| = n$
- The set of the items is J s.t. $|J| = m$
- The $valuation\ graph\ H$ is the complete bipartite graph on vertex sets I and J, where edge (i, j) has weight $v_i(j)$
- $H[\hat{I}, \hat{J}]$ is the subgraph of H induced by $\hat{I} \subseteq I$ and $\hat{J} \subseteq J$

The Algorithm:

Algorithm 1: Iterated Matching Algorithm

$A_i \leftarrow \emptyset$ for all $i \in I$;
$t \leftarrow 1; J_1 \leftarrow J$;
while $J_t \neq \emptyset$ **do**
  Compute a *maximum-weight matching* $M^t = \{(i, \mu_i^t)\}_{i \in I}$ in $H[I, J_t]$;
  Set $A_i \leftarrow A_i \cup \{\mu_i^t\}$ for all $i \in I$;
  Set $J_{t+1} \leftarrow J_t \setminus \cup_{i \in I}\{\mu_i^t\}$;
  $t \leftarrow t + 1$;
end



When I tried to expand the algorithm to construct a $EF1$ allocation for chores, I encountered the following issues:

1. When agents has negative valuations on items, we can not assume that for round T, there are exactly n items remaining. The reason is that dummy items (items that each agent has a utility of 0 for them) cannot be added, because now 0 is the best utility an agent can have.
2. If we were assume 1 (added dummy items), the following property can not be assumed: $v_i(\mu_i^t) \geq v_i(j)$ for any item $j \in J_{t+1}$.

   Here is an example:
   - Assume that $n = 3$, $m = 4$, and agents as negative utilities on items ($\neq 0$).
   - Also assume that the maximum-weight matching in the first round ($J_1 = J$) is: $\{(1,1),\ (2,2),\ (3,3)\}$
   - Now we are left to assign only the fourth item, and 2 dummy items.
   - Suppose agent 1 has the maximum value of utility over this item, so he will get it.
   - But now $v_1(\mu_1^1) < v_1(j)$ when $\mu_1^1 = 1,\ j = \mu_2^2\ (dummy)$.

Observation 1:
if $m = kn,\ for\ any\ k \in N$, the algorithm is good also for chores, because the $v_i(\mu_i^t) \geq v_i(j)$ property exists.

Observation 2:
If $m < n$: it's good as well, because each agent will get 0 or 1 items, so the envy of one agent over another can be at most in one item.

So we'll assume that $m > n\ and\ m(mod\ n) \neq 0$

…maybe use the envy graph

אפשר לחשוב על דבר כזה:
אם נשאר פריט אחד שלא חולק, אז נקצה אותו לזה שאין לו צלעות יוצאות בגרף הקנאה.

אם לכולם יש צלעות יוצאות בגרף הקנאה, אז זה אומר בהכרח שיש מעגלים, אבל אי אפשר לפרק אותם כי יש מגבלות קיבולת !!!

האלגוריתם עובד גם **למטלות**, כך:
עם הוספת מטלות dummy אבל כשמוסיפים הכל בהתחלה, ואז התכונה שכתבתי ב-(2) נשמרת:

Chores Algorithm:

1. $A_i \leftarrow \Phi \;\; for\; all\; i \in I;$
2. Suppose $|J| = an - k \;\; for\; k \in \{0,\ldots, n-1\}$
3. $D \leftarrow k\; dummy\; chores\; (each\; agent\; has\; 0\; utility\; on\; them)$
4. $J \leftarrow J \cup D \;\; [now\; |J| = an]$
5. $t \leftarrow 1;\; J_1 \leftarrow J;$
6. Do the same “while” loop from the above algorithm.

נשים לב שכעת כל התכונות שמבטיחות את נכונות האלגוריתם נשמרות, וההוכחה זהה לזו שבמאמר (3.2).

מטלות עם קטגוריות ואילוצי קיבולת:

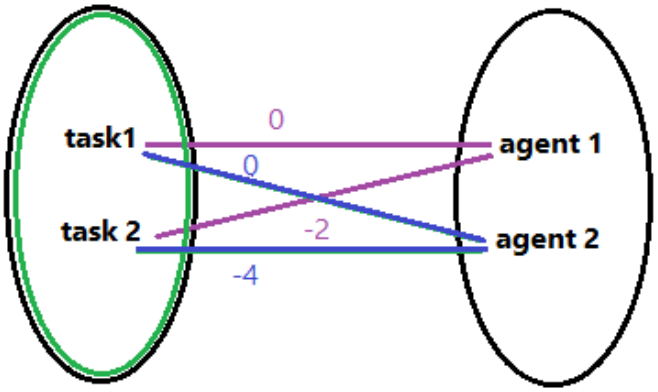
נניח את התכונה הזו:

$\left\lceil \frac{|C_h|}{n} \right\rceil \leq k_h$ (כאשר k_h היא הקיבולת עבור קטגוריה h ו-C_h הוא גודל קבוצת המטלות מקטגוריה h),
כלומר מספר המטלות שניתן לקבל מכל קטגוריה הוא לפחות מספר החפצים מהקטגוריה חלקי מספר הסוכנים (נשים לב שחייב את זה אם לא רוצים לאפשר זריקת חפצים).

רעיון 1:
לבצע את chores algorithm עם הוספת לולאה חיצונית שרצה על כל הקטגוריות (וכל פעם מוסיפה מטלות dummy עבור כל קטגוריה בנפרד).
כך אילוצי הקיבולת נשמרים (כי בכל פעם מקצים את המטלות **לכל** הסוכנים, ומניחים את התכונה שציינתי לעיל. דומה ל-round robin).
אבל הבעיה היא שלאו דווקא נשמור על הקנאה בלכל היותר חפץ אחד במעבר בין קטגוריות, למשל:

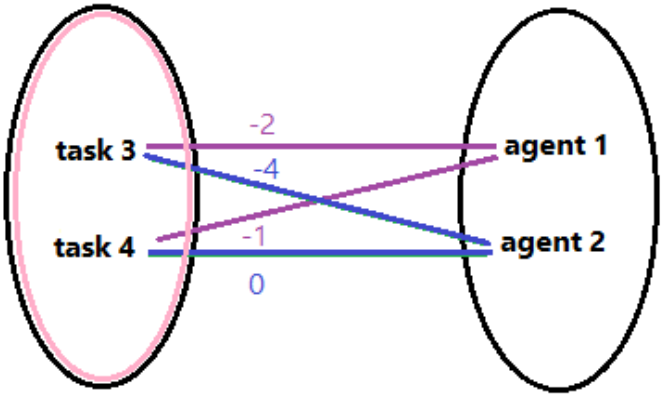
	task 1	task 2	task 3	task 4
agent 1	0	-2	-2	-1
agent 2	0	-4	-4	0

אזי לפי האלגוריתם:
- חלוקת הקטגוריה הירוקה:



השידוך המקסימלי הוא $A_1 = \{2\},\ A_2 = \{1\}$. כעת סוכן 1 חושב שיש לו תועלת של 2-, ושלסוכן 2 יש תועלת של 0, לכן הוא מקנא בו (כמובן בלכל היותר חפץ אחד), וסוכן 2 חושב שיש לו תועלת של 0' לכן הוא לא מקנא.

- חלוקת הקטגוריה הורודה:



השידוך המקסימלי הוא $B_1 = \{3\},\ B_2 = \{4\}$. יחד עם ההקצאות בקטגוריה הראשונה, סוכן 1 חושב שיש לו תועלת של 4-, ושלסוכן 2 יש תועלת של 1-, לכן הורדת מטלה אחת לא תספיק כדי שסוכן 1 לא יקנא!

<u>רעיון 2:</u>

להתחשב איכשהו במצב הנוכחי (ההקצאות מכל הקטגוריות שיצאו עד כה), כדי לא ליצור מצב שמי שכבר מקנא במישהו בחפץ אחד יקנא בעוד חפץ בקטגוריה החדשה.

אבל בשביל זה צריך לדאוג לכך שבטוח קיים מישהו שלא יקנא , כדי להקצות לו את המטלה החדשה - הבעיה הגדולה שלנו.

בנוסף, אם מקצים למישהו שלא מקנא יכולים ליצור מעגל קנאה, ולא בטוח שתמיד אפשר לפרק אותו, כי אנו יכולים לפגוע ב-EF1 (ואז בסבב הבא לא יהיה מישהו שלא יקנא בכלל ולא נדע למי להקצות).

<u>רעיון 3:</u>

התכונה הבעייתית שנכונה לגבי האלגוריתמים שכן עובדים (לגבי goods או chores ללא אילוצי קיבולת),

אבל לא נכונה עם קטגוריות היא: $v_i(\mu_i^t) \geq v_i(j)$ עבור כל פריט $j \in J_{t+1}$.

אם נשמור עליה כאן, האלגוריתם יעבוד.

- אם המטלות היו מחולקות בעצמן לקטגוריות "מטלות קשות", "מטלות בינוניות", "מטלות קלות", אז היה ניתן לחלק אותן מהקל לקשה והתכונה הייתה נשמרת.
- האם אפשר לעשות זאת בצורה מלאכותית תוך שמירה על מגבלות הקיבולת?