Approximation Schemes for Scheduling on Parallel Machines

Noga Alon * Yossi Azar * Gerhard J. Woeginger * Tal Yadid

מבוא-

הגדרת הבעיה- בהינתן m מכונות זהות $M_1, \dots, M_m$ וn מטלות בלתי תלויות עם זמני עיבוד $p_1, \dots, p_n$. נגדיר תזמון כך- חלוקת כל המטלות לm המכונות. עבור כל תיזמון נגדיר לכל מכונה M_i זמן סיום C_i. עבור פונקציה קבועה $f: R_0^+ \to R_0^+$ המאמר מתרכז בשתי בעיות:

1. $minimize \sum_{i\in[m]} f(C_i)$
2. $minimize \max_{i\in[m]} f(C_i)$

בצורה פורמלית ניתן לסמן את הבעיות כך:

1. $P|\cdot|\sum f(C_i)$
2. $P|\cdot|\max f(C_i)$

הוכח כבר במאמרים קודמים שהבעיות הנ"ל שייכות למחלקה $strong\ NP\ complete$ ולכן מטרת המאמר היא למצוא אלגוריתם קירוב פולינומי ולא אלגוריתם אופטימלי. (ע"פ הגדרה, לבעיות מסוג זה אין אלגוריתם אופטימלי בזמן פסודו פולינומי)

עבור $\epsilon > 0$ נגדיר אלגוריתם קירוב כך: $|OPT - OUTPUT| \le \epsilon \cdot OPT$

משפחת השפות שקיים להן אלגוריתם קירוב פולינומי עבור כל $\epsilon > 0$ נקראת *PTAS*.

לאורך בשנים נעשו המון ניסויים לבנות אלגוריתמים לבעיות המתוארות מעלה עבור פונקציות f שונות. הבעיה הקלאסית ביותר היא כאשר $f(x) = x$, עבור בעיה זו נבנו כמה אלגוריתמים חמדניים מסוגים שונים הנותנים תוחלות קירוב שונות. $Graham$ למשל בנה אלגוריתם עבור $\epsilon = 1$ וגם עבור $\epsilon = \frac{1}{3}$

נדגיש שהשוני המרכזי המובא במאמר זה הוא שכאן אנו בונים אלגוריתם עבור פונקציות f כלליות ולא עבור פונקציות ספציפיות כמו שחקרו כבר במאמרים רבים.

נשים לב שמאמר זה מתרכז אך ורק בפונקציות f קמורות המקיימות את ההגדרה:

$$f(x + \Delta) + f(y - \Delta) \le f(x) + f(y)\ holds\ for\ all\ 0 \le x \le y\ with\ 0 \le \Delta \le y - x$$

בנוסף נדרוש שהפונקציה תקיים את התכונה הבאה:

$$(F*): For\ all\ \varepsilon > 0\ there\ exists\ a\ \delta > 0\ whose\ value\ depends\ only\ on\ \varepsilon,$$
$$such\ that: x, y \ge 0, |y - x| \le \delta x \Rightarrow |f(y) - f(x)| \le \varepsilon f(x)$$

נגדיר $L = \frac{1}{m}\sum_{j\in[n]} p_j$ כלומר L הוא זמן הסיום הממוצע של m המכונות.

בתחילת המאמר מוכח שאם יש מטלה שמשקלה גדול מL קיים תזמון אופטימלי בו המכונה המעבדת את המטלה הזו, לא מעבדת אף מטלה אחרת בנוסף אליה. כלומר את כל המטלות "הכבדות" נוכל לחלק בהתחלה למכונות נפרדות- כל מכונה שמקבלת מטלה "כבדה" יכולה לצאת מהתזמון ונישאר להתמודד עם המטלות הקלות. לכן מכאן ועד סוף המאמר נתעסק רק בתזמון של מטלות שמשקלן קטן מL.

אבחנה נוספת שמוכיחים במאמר היא שאם כל המטלות הן במשקל קטן מL, קיים פתרון אופטימלי בו זמן העיבוד של כל מכונה הוא בין $2L$ ל$0.5L$

באופן כללי הרעיון אותו מביא המאמר הוא כזה- ע"מ לקבל אלגוריתם קירוב לבעיה המקורית נבנה דוגמת קלט "מקורבת" וקטנה יותר. ואותה נפתור באופן אופטימלי ויעיל. את התוצאה של הבעיה המקורבת נמיר חזרה לבעיה המקורית ונקבל אלגוריתם קירוב פולינומי לבעיה המקורית.
כלומר מהלך האלגוריתם הוא כזה:

1. בהינתן מספר שלם λ נחשב את $I^{\#}(\lambda)$ מתוך I
2. נפתור את $I^{\#}(\lambda)$ בצורה אופטימלית בעזרת *ILP* ונקבל תזמון $\Sigma^{\#}$
3. נתרגם את $\Sigma^{\#}$ לתזמון Σ על פני המטלות בI
4. נקבל תזמון עם תוחלת קירוב שתלוי בλ

פירוט חלק 1.

אלגוריתם המרה לבעיה ה*transformed*:

בהינתן רשימת מטלות I ומספר שלם λ נגדיר רשימת מטלות $I^{\#}(\lambda)$ כך:

- לכל מטלה ב*j* כך ש: $p_j > \frac{L}{\lambda}$ נוסיף ל$I^{\#}(\lambda)$ מטלה $p_j^{\#}$ שמשקלה המספר השלם הקרוב ביותר אליה מלמעלה שהינו כפולה של $\frac{L}{\lambda^2}$
- יהא *S* סכום קבוצת כל המטלות ב*j* שזמן העיבוד שלהן $\geq \frac{L}{\lambda}$ נגדיר $S^{\#}$ משקל *S* מעוגל לשלם הקרוב ביותר מלמעלה שהינו כפולה של $\frac{L}{\lambda}$ נוסיף ל$I^{\#}(\lambda)$ $\frac{S^{\#}\lambda}{L}$ מטלות חדשות במשקל $\frac{L}{\lambda}$ כל אחת.

פירוט חלק 2.

הפסקה הבאה מפרטת על איך נראה קלט ופלא לאלגוריתם תכנות לינארי בשלמים: לא תרגמתי את ההגדרות מאחר והתיאור פה מתומצת ממילא ולא קריטי לרעיון הכללי:

First, observe that all jobs $in\ I^{\#}(\lambda)$ have processing times of the form kL/λ^2 with $\lambda \leq k \leq \lambda^2$. We use the following notation to represent the input and the assignments of jobs to machines: The input jobs are represented as a vector $n = (n_\lambda, \ldots, n_{\lambda^2})$, where n_k denotes the number of jobs whose processing time equals $\frac{kL}{\lambda^2}$. An assignment to a machine is a vector $u = (u_\lambda, \ldots, u_{\lambda^2})$, where u_k is the number of jobs of length $\frac{kL}{\lambda^2}$ assigned to that machine. The length of assignment u, denoted $C(u)$, is $\sum_{\lambda \leq k \leq \lambda^2} u_k \cdot \frac{kL}{\lambda^2}$. Denote by U the set of all assignment vectors u with $\frac{1}{2} \cdot L^{\#} < C(u) < 2L^{\#}$. It is easy to see that each assignment $u \in U$ consists of at most 2λ jobs and that therefore $|U| \leq \lambda^{4\lambda}$ holds.

אחרי שקיבלתי דוגמת קלט מקורבת נפתור אותה בעזרת תיכנון לינארי בשלמים *ILP* מערכת הבעיה מוגדרת כך:

$$\min \sum_{\vec{u} \in U} x_{\vec{u}} \cdot f(C(\vec{u}))$$

subject to

$$\sum_{\vec{u} \in U} x_{\vec{u}} = m$$

$$\sum_{\vec{u} \in U} x_{\vec{u}} \cdot \vec{u} = \vec{n}$$

$$x_{\vec{u}} \geq 0 \qquad \forall\ \vec{u} \in U$$

פירוט חלק 3.

עקרון ההמרה חזרה מעט מסובך בשבילי ולכן אני מביאה אותו כלשונו מהמאמר.

Proof. In schedule $\Sigma^{\#}$, replace every big job $J_j^{\#}$ by its corresponding job J_j in I. This cannot increase the machine completion times; any machine completion time that is decreased is decreased by at most a factor of $\lambda/(\lambda+1)$. Next, let $s_i^{\#}$ denote the number of jobs of length L/λ that are processed on machine M_i in schedule $\Sigma^{\#}$; rearrange on every machine the jobs in such a way that these small jobs are processed in the end. Below, we will show how to partition the small jobs in instance I into m parts $S_1^*, \ldots, S_m^*$ such that the total size of the jobs in part S_i^* is between $(s_i^{\#}-2)L/\lambda$ and $(s_i^{\#}+1)L/\lambda$. Then replacing on every machine M_i the $s_i^{\#}$ jobs of length L/λ by the small jobs in S_i^* yields the desired schedule that fulfills inequality (6).

In order to construct the partition $S_1^*, \ldots, S_m^*$, one proceeds as follows. In the first partitioning step, greedily assign to every set S_i^* a number of small jobs from I until the total size of assigned jobs exceeds $(s_i^{\#}-2)L/\lambda$. Note that the total size of jobs assigned to S_i^* is bounded by $(s_i^{\#}-1)L/\lambda$. Since

$$\sum_{i=1}^{m}(s_i^{\#}-1)L/\lambda \;\le\; S^{\#} - mL/\lambda \;\le\; S, \tag{7}$$

one can complete the first partitioning step without running out of jobs. In the second partitioning step, greedily add the remaining small jobs to the sets S_i^*, but without letting the total size assigned to S_i^* exceed $(s_i^{\#}+1)L/\lambda$. It can be shown that indeed all small jobs are assigned in the second partitioning step. ■

פירוט חלק 4.

תת הפרק האחרון במאמר עוסק באיכות הקירוב ביחס לλ

Finally, we will put everything together and derive the desired PTAS.
Consider some fixed real number $\varepsilon, 0 < \varepsilon < 1$.
Let $\delta > 0$ be a real number that fulfills
$f(y) \le (1+\frac{\varepsilon}{3})f(x)$ for all $x, y \ge 0$ with $|x-y| \le \delta x$.
Note that by condition (F∗) such a δ always exists.
Define a positive integer $\lambda := \left\lceil \frac{5}{\delta} \right\rceil$

במאמר מוכח שאם נבחר λ בצורה הזו נקבל

$$Approx \le \left(1+\frac{\varepsilon}{3}\right)^2 \cdot Opt \le (1+\varepsilon)Opt.$$