A tensor optimization algorithm for computing Lagrangians of hypergraphs

Jingya Chang[1] · Bin Xiao[1] · Xin Zhang[2]

Abstract The Lagrangian of a hypergraph is a crucial tool for studying hypergraph extremal problems. Though Lagrangians of some special structure hypergraphs have closed-form solutions, it is a challenging problem to compute the Lagrangian of a general large scale hypergraph. In this paper, we exploit a fast computational scheme involving the adjacency tensor of a hypergraph. Furthermore, we propose to utilize the gradient projection method on a simplex from nonlinear optimization for solving the Lagrangian of a large scale hypergraph iteratively. Using the Łojasiewicz gradient inequality, we analyze the global and local convergence of the gradient projection method. Numerical experiments illustrate that the proposed numerical method could compute Lagrangians of large scale hypergraphs efficiently.

✉ Xin Zhang
zhangxin0619@126.com

Jingya Chang
jychang@gdut.edu.cn

Bin Xiao
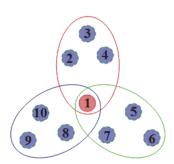2112114085@mail2.gdut.edu.cn

[1] School of Mathematics and Statistics, Guangdong University of Technology, Guangzhou 510006, People's Republic of China

[2] School of Arts and Science, Suqian University, Suqian 223800, People's Republic of China

1 Introduction

Hypergraphs have important applications in science and engineering, such as subspace clustering [7,11,31], hypergraph matching [18], image processing [9], and network analysis [4,25], due to its capability of modeling multiwise similarity. Hypergraph extremal problems, which maximize values of constrained multilinear functions, attract attentions of scholars from many fields of mathematics [15]. In this paper, we focus on one kind of hypergraph extremal problems: the Lagrangian of a hypergraph.

An r-uniform hypergraph (r-graph) $G = (V, E)$ consists of a vertex set $V := \{1, 2, \ldots, n\}$ and an edge set $E \subseteq V^{(r)}$, where $V^{(r)}$ denotes the collection of all subsets of V of size r, $m = |E|$ is the number of edges. For example the sunflower hypergraph in Figure 1 is a 4-uniform hypergraph with 10 vertices. The weight polynomial for G is defined as $w(G, \mathbf{x}) := \sum_{e \in E} \prod_{i \in e} x_i$, where legal weighting for G satisfies $x_i \geq 0$ for all $i \in V$ and $\sum_{i \in V} x_i = 1$. Let $\Delta := \{\mathbf{x} \in \Re_+^n : \mathbf{e}^T \mathbf{x} = 1\}$ be the legal weighting set that is a simplex. Here $\mathbf{e}$ is the all one vector. The Lagrangian of an r-graph G [26,29] is to maximize the weight polynomial $w(G, \mathbf{x})$ under the simplex constraint

$$\lambda(G) := \max \left\{ w(G, \mathbf{x}) = \sum_{e \in E} \prod_{i \in e} x_i \ : \ \mathbf{e}^T \mathbf{x} = 1, \ \mathbf{x} \in \Re_+^n \right\}. \tag{1}$$

Fig. 1 A 4-uniform hypergraph: sunflower

Motzkin and Straus [26] introduced the concept of Lagrangians for 2-graphs and built a bridge between Lagrangians and Turán's theorem. For $r = 2$, $\lambda(G)$ is achieved by equally distributing the weight over the vertices of the largest clique in G and setting $x_i = 0$ for all other vertices i. Moreover, Lagrangians of hypergraphs have some significant results. Let K_n^r denote the complete r-graph on $V := \{1, 2, \ldots, n\}$, where $E = V^{(r)}$. Then, it is pointed out in [20] that

$$\lambda(K_n^r) = \binom{n}{r} \frac{1}{n^r}$$

and the associated solution is $\mathbf{x}^* = (\frac{1}{n}, \ldots, \frac{1}{n})^T$. Let t be a positive integer and $\{f_1, f_2, \ldots, f_{t+2}\}$ be pairwise disjoint 3-subsets. Let $\mathcal{F}^*$ be the 3-graph with the vertex set $V = \bigcup_{1 \leq k \leq t+2} f_k$ and the edge set $\{f_1, f_2, \ldots, f_{t+2}\} \cup \{f \in V^3 : |f \cap f_k| \leq 1 \text{ for } k = 1, 2, \ldots, t+2\}$. Then,

$$\lambda(\mathcal{F}^*) = \left(27\binom{t+2}{3} + t + 2\right)\frac{1}{27(t+2)^3} < \lambda(K^3_{3t+4}).$$

Frankl and Füredi [14] conjectured in 1989 that an initial segment of the colexicographic order has the largest Lagrangian of any r-graph with size m. The Frankl–Füredi conjecture was studied by many authors [24,28]. Although the Frankl–Füredi conjecture is true in the case of $r = 3$ [29,16], Gruslys et al. [15] disproved it for $r \geq 4$.

While theoretical researches on Lagrangians of hypergraphs are rich, the values of Lagrangians of ordinary hypergraphs are still unclear with the aid of theoretical analysis. Therefore, we are going to design numerical method for computing the Lagrangian of general uniform hypergraphs.

Rich theoretical results imply that the Lagrangian of a hypergraph reveals special nature of the adjacency tensor of a hypergraph. However, it is a challenging problem to compute the Lagrangian of a large scale general hypergraph. In this paper, we customize the gradient projection algorithm from nonlinear optimization for computing the Lagrangian of a large scale general hypergraph. On one hand, from the viewpoint of spectral hypergraph theory, the Lagrangian of an r-graph G is closely related to the adjacency tensor $\mathcal{A}$ of G, which is an rth order structure tensor with valuable symmetry and sparsity. Fast computations of the structure tensors arising from a hypergraph could be employed for the Lagrangian problem. The computation cost for computing function values and gradients of the weight polynomial $w(G, \mathbf{x})$ is proportional to the size m of the r-graph and the square of r. On the other hand, the legal weighting set Δ is a simplex, which is a closed convex set. Since the projection onto the simplex is cheap, we design a gradient projection method for solving the Lagrangian of an r-graph G, where the initial step size at each iteration is Barzilai–Borwein step size.

Due to the semi-algebraic property of the Lagrangian of an r-graph G, the Łojasiewicz inequality holds. Using the Łojasiewicz inequality, we analyze the global convergence of the gradient projection algorithm, i.e., the sequence of iterates generated by the gradient projection algorithm converges to a critical point with linear or sublinear rate. Numerical experiments on small and large scale hypergraphs illustrate that the gradient projection algorithm is powerful and efficient. In particular, the gradient projection algorithm could compute Lagrangians of hypergraphs with thousands of edges.

The outline of this paper is drawn as follows. Section 2 presents the gradient projection algorithm and associated fast computations on hypergraphs. Global and local convergence is analyzed in Section 3. Numerical experiments on small and large scale hypergraphs are reported in Section 4. Finally, some concluding remarks are made in Section 5.

2 Gradient projection method

To handle large scale hypergraphs, there are roughly two kinds of strategies: (i) hardware acceleration develops and optimizes CPU/GPU kernels to process hypergraph algorithms and (ii) software acceleration uses tensors and tensor operators to represent hypergraph computing into a unique (compact) format that can be executed efficiently [22]. In this paper, we follow the software acceleration strategy to exploit tensor representations and fast computations for the purpose of the Lagrangian computing of hypergraphs. At the beginning, for an r-graph G, the weight polynomial $w(G, \mathbf{x})$ is determined by the adjacency tensor of G. First we introduce the definition of tensor and hypergraph related adjacency tensor.

Definition 2.1 (Symmetry tensor [27]) *A tensor*

$$\mathcal{T} = (t_{i_1 \ldots i_r}) \in \Re^{[r,n]}, \qquad for \ i_j = 1, \ldots, n, j = 1, \ldots, r$$

is an rth order n dimensional symmetric tensor if the value of $t_{i_1 \ldots i_k}$ is invariable under any permutation of its indices.

Definition 2.2 (Adjacency tensor [13]) *Let $G = (V, E)$ be an r-graph with n vertices. The adjacency tensor of G is an rth order n-dimensional symmetric tensor $\mathcal{A} = [a_{i_1 \ldots i_r}]$, of which elements are*

$$a_{i_1 \cdots i_r} = \begin{cases} \dfrac{1}{(r-1)!} & if \ \{i_1, \ldots, i_r\} \in E, \\ 0 & otherwise. \end{cases}$$

Utilizing the adjacency tensor $\mathcal{A}$ of the r-graph G, we have the following lemma.

Lemma 2.3 *Let G be an r-graph with n vertices. Its weight polynomial $w(G, \mathbf{x})$ could be rewritten as*

$$w(G, \mathbf{x}) = \frac{1}{r} \mathcal{A} \mathbf{x}^r,$$

where $\mathcal{A} \mathbf{x}^r := \sum_{i_1=1}^{n} \cdots \sum_{i_r=1}^{n} a_{i_1 \cdots i_r} x_{i_1} \cdots x_{i_r}$.

Proof By direct calculations, it yields that

$$\begin{aligned} \frac{1}{r} \mathcal{A} \mathbf{x}^r &= \frac{1}{r} \sum_{i_1=1}^{n} \cdots \sum_{i_r=1}^{n} a_{i_1 \cdots i_r} x_{i_1} \cdots x_{i_r} \\ &= \frac{1}{r} \sum_{\substack{\{i_1,\ldots,i_r\} \in E \\ i_1 < \cdots < i_r}} \frac{1}{(r-1)!} r! x_{i_1} \cdots x_{i_r} \\ &= \sum_{e \in E} \prod_{i \in e} x_i = w(G, \mathbf{x}). \end{aligned} \qquad (2)$$
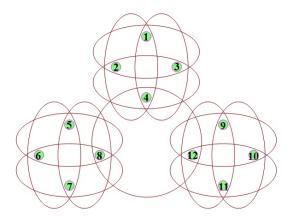
The proof is completed.

Fig. 2 A toy hypergraph.

Hence, to compute the Lagrangian of an r-graph G, we will solve the following optimization problem

$$\lambda(G) = \begin{cases} \max & f(\mathbf{x}) = \dfrac{1}{r}\mathcal{A}\mathbf{x}^r \\ \text{s.t.} & \mathbf{x} \in \Delta, \end{cases} \qquad (3)$$

which is equivalent to (1). Since the adjacency tensor $\mathcal{A}$ of G is symmetric, it holds [27,10] that $\nabla f(\mathbf{x}) = \mathcal{A}\mathbf{x}^{r-1}$, where elements of the vector $\mathcal{A}\mathbf{x}^{r-1}$ are

$$(\mathcal{A}\mathbf{x}^{r-1})_j = \sum_{i_2=1}^{n} \cdots \sum_{i_r=1}^{n} a_{ji_2\cdots i_r} x_{i_2} \cdots x_{i_r} \qquad \forall\, j = 1, \ldots, n.$$

Since the adjacency tensor $\mathcal{A}$ of an r-graph G is a structure tensor, the storage and computation involving $\mathcal{A}$ are economic [8]. For example, we consider the 3-graph illustrated in Figure 2, i.e., $r = 3$ here. The 3-graph has $n = 12$ vertices and $m = 13$ edges. The associated adjacency tensor is a 3rd order 12 dimensional symmetric tensor with $n^r = 1,728$ elements. Using the storage technique from [8], we save the incidence matrix of the 3-graph in a compact form

$$H = \begin{pmatrix} 1 & 1 & 1 & 2 & 5 & 5 & 5 & 6 & 9 & 9 & 9 & 10 & 4 \\ 2 & 2 & 3 & 3 & 6 & 6 & 7 & 7 & 10 & 10 & 11 & 11 & 8 \\ 3 & 4 & 4 & 4 & 7 & 8 & 8 & 8 & 11 & 12 & 12 & 12 & 12 \end{pmatrix}^T,$$

which only has $m \times r = 39$ positive integers.

To compute the scalar $\mathcal{A}\mathbf{x}^r$ for an input vector $\mathbf{x} = [x_j] \in \Re^n$, we introduce a matrix $Y = [y_{ik}] \in \Re^{m \times r}$ of which elements are

$$y_{ik} = x_{H_{ik}}, \qquad \forall\, i = 1, \ldots, m \text{ and } k = 1, \ldots, r.$$

Algorithm 1 Euclidean projection $\pi_\Delta(\mathbf{a})$.

1: Sort the input vector $\mathbf{a} \in \Re^n$ into an auxiliary vector $\mathbf{b}$ such that $b_1 \geq b_2 \geq \cdots \geq b_n$.
2: Find the largest index $\ell = \max\{j : jb_j + 1 - \sum_{i=1}^{j} b_i > 0, 1 \leq j \leq n\}$.
3: Set $\lambda = (1 - \sum_{i=1}^{\ell} b_i)/\ell$.
4: Output $\mathbf{x}$ with $x_i = \max\{a_i + \lambda, 0\}$.

Then, it can be deduced from (2) that

$$\mathcal{A}\mathbf{x}^r = r \sum_{i=1}^{m} \prod_{k=1}^{r} y_{ik}.$$

Similarly, to compute the vector $\mathcal{A}\mathbf{x}^{r-1}$, we define

$$M^{(k)} := [\delta(i, H_{jk})]_{ij} \quad \text{and} \quad \mathbf{y}^{(k)} := \left[\prod_{\substack{\ell=1 \\ \ell \neq k}}^{r} y_{j\ell} \right]_j,$$

for $k = 1, \ldots, r$, where $\mathbf{y}^{(k)}$ specifies values and $M^{(k)}$ indicates locations. Here, $\delta(\cdot, \cdot)$ stands for the Kronecker delta. Then, we have

$$\mathcal{A}\mathbf{x}^{r-1} = \sum_{k=1}^{r} M^{(k)}\mathbf{y}^{(k)}.$$

The computational cost for computing $\mathcal{A}\mathbf{x}^r$ and $\mathcal{A}\mathbf{x}^{r-1}$ is about $\mathcal{O}(mr^2)$, which is cheap due to the structure of an r-graph.

It is easy to see that the feasible region Δ is a closed convex set. The Euclidean projection

$$\pi_\Delta(\mathbf{a}) := \arg \min_{\mathbf{x} \in \Delta} \|\mathbf{x} - \mathbf{a}\|$$

exists and is unique for any vector $\mathbf{a} \in \Re^n$. Algorithm 1 provides an $\mathcal{O}(n \log n)$ approach for computing the Euclidean projection onto Δ [30,12].

Now, we give the gradient projection algorithm formally in Algorithm 2. The gradient projection algorithm is an iterative method from nonlinear optimization [5]. Starting from an initial iteration point $\mathbf{x}_0 \in \Delta$, we compute the gradient of the objective function and the associated gradient projection direction. Next, we select an initial step size by Barzilai–Borwein method [3, 21] and then perform backtracking line search along this gradient projection direction. At each iteration, only one projection is performed. If $\mathbf{g}_c$ in Step 3 of Algorithm 2 is sufficiently small or the total number of iterations is large enough, we terminate the algorithm.

Algorithm 2 A gradient projection algorithm (GPA).

1: Choose $0 < \underline{\alpha} \le \alpha_0 \le \overline{\alpha}$, $\beta \in (0,1)$, $\eta \in (0, 1/2]$, and $\mathbf{x}_0 \in \Delta$. Set $c \leftarrow 0$.
2: **while** the sequence of iterates does not converge **do**
3:     Set $\mathbf{g}_c = \pi_\Delta(\mathbf{x}_c + \alpha_c \nabla f(\mathbf{x}_c)) - \mathbf{x}_c$.
4:     Find the smallest nonnegative integer j such that

$$f(\mathbf{x}_c + \beta^j \mathbf{g}_c) - f(\mathbf{x}_c) \ge \eta \beta^j \mathbf{g}_c^T \nabla f(\mathbf{x}_c). \tag{4}$$

5:     Set $\rho_c = \beta^j$ and $\mathbf{x}_{c+1} = \mathbf{x}_c + \rho_c \mathbf{g}_c$.
6:     Choose $\alpha_{c+1} \in [\underline{\alpha}, \overline{\alpha}]$ and set $c \leftarrow c + 1$.
7: **end while**

---

3 Convergence analysis

Now, we analyze the convergence of the proposed GPA. Part of our analysis on basic theory is similar to [5,17] and we present here for the purpose of completion. First, we call $\mathbf{x}_* \in \Delta$ a critical point of (3) if

$$\pi_\Delta(\mathbf{x}_* + \nabla f(\mathbf{x}_*)) = \mathbf{x}_*.$$

Since the simplex Δ is a nonempty, closed, and convex set, we get the following lemma [17, Proposition 2.1].

Lemma 3.1 *Let $\mathbf{g}^\alpha(\mathbf{x}) = \pi_\Delta(\mathbf{x} + \alpha \nabla f(\mathbf{x})) - \mathbf{x}$. Then,*
1) $\|\mathbf{g}^\alpha(\mathbf{x})\|$ is nondecreasing in $\alpha > 0$ for all $\mathbf{x} \in \Delta$;
2) $\|\mathbf{g}^\alpha(\mathbf{x})\|/\alpha$ is nonincreasing in $\alpha > 0$ for all $\mathbf{x} \in \Delta$;
3) $\mathbf{g}^\alpha(\mathbf{x})^T \nabla f(\mathbf{x}) \ge \|\mathbf{g}^\alpha(\mathbf{x})\|^2/\alpha$ for all $\mathbf{x} \in \Delta$ and $\alpha > 0$;
4) for any $\mathbf{x} \in \Delta$ and $\alpha > 0$, $\mathbf{g}^\alpha(\mathbf{x}) = 0$ if and only if $\mathbf{x}$ is a stationary point for (3).

Because $f(\mathbf{x}) = \frac{1}{r}\mathcal{A}\mathbf{x}^r$ is a polynomial and the simplex Δ is compact, the Hessian $\nabla^2 f(\mathbf{x})$ is bounded, i.e., there exists a constant $M \ge 1$ such that

$$\|\nabla^2 f(\mathbf{x})\| \le M \qquad \forall \mathbf{x} \in \Delta.$$

Lemma 3.2 *There exists a constant $\kappa_1 > 0$ such that for any ρ_c generated by Algorithm 2*

$$\rho_c \ge \kappa_1.$$

Proof We will prove that the inequality (4) is valid when $\rho \in [0, \frac{2(1-\eta)}{\overline{\alpha}M}]$. By Lemma 3.1, it yields that $\|\mathbf{g}_c\|^2 \le \alpha_c \mathbf{g}_c^T \nabla f(\mathbf{x}_c)$. From Taylor's formula, we have

$$\begin{aligned}
f(\mathbf{x}_c + \rho \mathbf{g}_c) &\ge f(\mathbf{x}_c) + \rho \mathbf{g}_c^T \nabla f(\mathbf{x}_c) - \frac{M}{2}\rho^2\|\mathbf{g}_c\|^2 \\
&\ge f(\mathbf{x}_c) + \rho \mathbf{g}_c^T \nabla f(\mathbf{x}_c) - \frac{M}{2}\rho^2 \alpha_c \mathbf{g}_c^T \nabla f(\mathbf{x}_c) \\
&\ge f(\mathbf{x}_c) + \rho \mathbf{g}_c^T \nabla f(\mathbf{x}_c) - (1-\eta)\rho \mathbf{g}_c^T \nabla f(\mathbf{x}_c) \\
&= f(\mathbf{x}_c) + \eta \rho \mathbf{g}_c^T \nabla f(\mathbf{x}_c),
\end{aligned}$$

where the last inequality holds owing to the assumption $0 \leq \rho \alpha_c M \leq \rho \overline{\alpha} M \leq 2(1 - \eta)$. According to Step 4 of Algorithm GPA, this lemma is valid if $\kappa_1 := \frac{2\beta(1-\eta)}{\overline{\alpha} M}$.

Theorem 3.3 *Suppose that $\{\mathbf{x}_c\}$ is an infinity sequence of iterates generated by GPA. Then, we have*

$$\lim_{c \to \infty} \|\pi_\Delta(\mathbf{x}_c + \nabla f(\mathbf{x}_c)) - \mathbf{x}_c\| = 0.$$

That is to say, every limit point of $\{\mathbf{x}_c\}$ is a critical point.

Proof From (4), Lemmas 3.1 and 3.2, we obtain

$$f(\mathbf{x}_{c+1}) - f(\mathbf{x}_c) \geq \eta \rho_c \mathbf{g}_c^T \nabla f(\mathbf{x}_c) \geq \frac{\eta \rho_c}{\alpha_c} \|\mathbf{g}_c\|^2 \geq \frac{\eta \kappa_1}{\overline{\alpha}} \|\mathbf{g}_c\|^2. \tag{5}$$

Hence, we have

$$\sum_{c=1}^{\infty} \|\mathbf{g}_c\|^2 \leq \frac{\overline{\alpha}}{\eta \kappa_1} \sum_{c=1}^{\infty} [f(\mathbf{x}_{c+1}) - f(\mathbf{x}_c)] \leq \frac{\overline{\alpha}}{\eta \kappa_1} \lambda(G),$$

which means that $\|\mathbf{g}_c\| \to 0$ as $c \to \infty$.

On one hand, if $\alpha_c \geq 1$, we get $\|\mathbf{g}^{\alpha_c}(\mathbf{x}_c)\| \geq \|\mathbf{g}^1(\mathbf{x}_c)\|$ by Lemma 3.1. On the other hand, $\alpha_c < 1$ and hence $\|\mathbf{g}^{\alpha_c}(\mathbf{x}_c)\|/\alpha_c \geq \|\mathbf{g}^1(\mathbf{x}_c)\|$. Hence, we have

$$\|\mathbf{g}_c\| = \|\mathbf{g}^{\alpha_c}(\mathbf{x}_c)\| \geq \min\{\alpha_c, 1\}\|\mathbf{g}^1(\mathbf{x}_c)\| \geq \min\{\underline{\alpha}, 1\}\|\mathbf{g}^1(\mathbf{x}_c)\|. \tag{6}$$

Therefore $\|\mathbf{g}^1(\mathbf{x}_c)\|$ tends to zero. The proof is then completed.

3.1 Further results based on Łojasiewicz gradient inequality

Because the objective function and the constraint set of (3) are semi-algebraic, the following Łojasiewicz gradient inequality is valid [23,6]. The analysis of this subsection is based on work in [2].

Theorem 3.4 (Łojasiewicz property) *Suppose that $\mathbf{x}_*$ is a critical point of (3), i.e., $\mathbf{g}^1(\mathbf{x}_*) = 0$. Then, there exist a neighborhood of $\mathbf{x}_*$ denoted as $\mathbb{U}(\mathbf{x}_*)$, an exponent $\theta \in [1/2, 1)$, and a positive constant C such that the following inequality*

$$\|\mathbf{g}^1(\mathbf{x})\| \geq C|f(\mathbf{x}) - f(\mathbf{x}_*)|^\theta$$

holds for all $\mathbf{x} \in \mathbb{U}(\mathbf{x}_*) \cap \Delta$.

Lemma 3.5 *Let $\mathbf{x}_*$ be a limiting point of $\{\mathbf{x}_c\}$. The initial iterate $\mathbf{x}_0$ is sufficiently close to $\mathbf{x}_*$ in the sense that $\mathbf{x}_0 \in \mathbb{B}(\mathbf{x}_*, \sigma) \cap \Delta$, where $\mathbb{B}(\mathbf{x}_*, \sigma) := \{\mathbf{x} \in \Re^n : \|\mathbf{x} - \mathbf{x}_*\| < \sigma\} \subseteq \mathbb{U}(\mathbf{x}_*)$ and $\sigma \geq \frac{\max\{1, \overline{\alpha}\}}{\eta C(1-\theta)}|f(\mathbf{x}_0) - f(\mathbf{x}_*)|^{1-\theta} + \|\mathbf{x}_0 - \mathbf{x}_*\|$. Then, we have the following two assertions:*

$$\mathbf{x}_c \in \mathbb{B}(\mathbf{x}_*, \sigma) \cap \Delta \qquad for \ c = 0, 1, 2, \ldots,$$

and

$$\sum_{c=0}^{\infty} \|\mathbf{x}_c - \mathbf{x}_{c+1}\| \leq \frac{\max\{1, \overline{\alpha}\}}{\eta C(1-\theta)}|f(\mathbf{x}_0) - f(\mathbf{x}_*)|^{1-\theta}.$$

Proof The conclusions can be proved by induction. It is obvious to see that $\mathbf{x}_0 \in \mathbb{B}(\mathbf{x}_*, \sigma) \cap \Delta$. Next, by supposing that there exists a positive integer ℓ such that

$$\mathbf{x}_c \in \mathbb{B}(\mathbf{x}_*, \sigma) \cap \Delta \qquad \text{for } c = 0, 1, \ldots, \ell,$$

we show $\mathbf{x}_{\ell+1} \in \mathbb{B}(\mathbf{x}_*, \sigma) \cap \Delta$.

Define $\phi(t) := \frac{1}{C(1-\theta)} |t - f(\mathbf{x}_*)|^{1-\theta}$. It is easy to see that $\phi(t)$ is a monotonically decreasing and concave function for $t < f(\mathbf{x}_*)$. Then, for $0 \le c \le \ell$, we have

$$
\begin{aligned}
\phi(f(\mathbf{x}_c)) - \phi(f(\mathbf{x}_{c+1})) &\ge \phi'(f(\mathbf{x}_c))(f(\mathbf{x}_c) - f(\mathbf{x}_{c+1})) \\
&= \frac{-1}{C|f(\mathbf{x}_c) - f(\mathbf{x}_*)|^\theta}(f(\mathbf{x}_c) - f(\mathbf{x}_{c+1})) \\
&\ge \frac{1}{\|\mathbf{g}^1(\mathbf{x}_c)\|}(f(\mathbf{x}_{c+1}) - f(\mathbf{x}_c)) \\
&\ge \frac{1}{\|\mathbf{g}^1(\mathbf{x}_c)\|} \frac{\eta \rho_c}{\alpha_c} \|\mathbf{g}_c\|^2 \\
&\ge \frac{1}{\|\mathbf{g}^1(\mathbf{x}_c)\|} \frac{\eta \rho_c}{\alpha_c} \|\mathbf{g}_c\| \min\{\alpha_c, 1\} \|\mathbf{g}^1(\mathbf{x}_c)\| \\
&= \frac{\eta \min\{\alpha_c, 1\}}{\alpha_c} \|\rho_c \mathbf{g}_c\| \\
&= \eta \min\{1, \alpha_c^{-1}\} \|\mathbf{x}_c - \mathbf{x}_{c+1}\|,
\end{aligned}
$$

where the second inequality is obtained based on Theorem 3.4, the third inequality is deduced from (5), and the fourth inequality is valid because of (6). The above inequalities indicate that

$$\|\mathbf{x}_c - \mathbf{x}_{c+1}\| \le \frac{\max\{1, \overline{\alpha}\}}{\eta} \left(\phi(f(\mathbf{x}_c)) - \phi(f(\mathbf{x}_{c+1})) \right),$$

for $0 \le c \le \ell$. Hence, it holds that

$$\sum_{c=0}^{\ell} \|\mathbf{x}_c - \mathbf{x}_{c+1}\| \le \frac{\max\{1, \overline{\alpha}\}}{\eta} \sum_{c=0}^{\ell} \left(\phi(f(\mathbf{x}_c)) - \phi(f(\mathbf{x}_{c+1})) \right) \le \frac{\max\{1, \overline{\alpha}\}}{\eta} \phi(f(\mathbf{x}_0)). \tag{7}$$

Therefore, we have

$$\|\mathbf{x}_{\ell+1} - \mathbf{x}_*\| \le \|\mathbf{x}_0 - \mathbf{x}_*\| + \sum_{c=0}^{\ell} \|\mathbf{x}_c - \mathbf{x}_{c+1}\| \le \|\mathbf{x}_0 - \mathbf{x}_*\| + \frac{\max\{1, \overline{\alpha}\}}{\eta} \phi(f(\mathbf{x}_0)) < \sigma,$$

which implies that $\mathbf{x}_{\ell+1} \in \mathbb{B}(\mathbf{x}_*, \sigma)$. On the other hand, every iterate is feasible by the mechanism of GPA. Hence, we obtain the first assertion. The second conclusion comes from (7) straightforwardly by setting $\ell \to \infty$.

Theorem 3.6 *Suppose that GPA generates an infinite sequence of iterates $\{\mathbf{x}_c\}$. Then the whole sequence $\{\mathbf{x}_c\}$ converges to a critical point $\mathbf{x}_*$.*

Proof Owing to the compactness of the feasible region Δ, there exists at least a limiting point $\mathbf{x}_*$ of $\{\mathbf{x}_c\}$. From Theorem 3.3, $\mathbf{x}_*$ is a critical point of the optimization problem (3). On the other hand, there is an iteration c_0 such that $\mathbf{x}_{c_0} \in \mathbb{B}(\mathbf{x}_*, \sigma) \cap \Delta$, where σ is specified by Lemma 3.5. By regarding $\mathbf{x}_{c_0}$ as an initial iterate in GPA, we know

$$\sum_{c=0}^{\infty} \|\mathbf{x}_c - \mathbf{x}_{c+1}\| = \sum_{c=0}^{c_0-1} \|\mathbf{x}_c - \mathbf{x}_{c+1}\| + \sum_{c=c_0}^{\infty} \|\mathbf{x}_c - \mathbf{x}_{c+1}\|$$

$$\leq \sum_{c=0}^{c_0-1} \|\mathbf{x}_c - \mathbf{x}_{c+1}\| + \frac{\max\{1, \overline{\alpha}\}}{\eta C(1-\theta)} |f(\mathbf{x}_{c_0}) - f(\mathbf{x}_*)|^{1-\theta} < \infty,$$

where the first inequality is owing to the second assertion of Lemma 3.5. Hence, we claim that the whole sequence $\{\mathbf{x}_c\}$ converges.

3.2 Convergence rate

In this subsection, we analyze the convergence rate of our GPA algorithm based on works in [1,19].

Theorem 3.7 *Suppose GPA generates an infinite sequence of iteration points $\{\mathbf{x}_c\}$ that converges to a critical point $\mathbf{x}_*$. Then, we have the following estimations on convergence rate.*

- *If $\theta = 1/2$, there exist $\gamma > 0$ and $\mu \in (0,1)$ such that*

$$\|\mathbf{x}_c - \mathbf{x}_*\| \leq \gamma \mu^c.$$

- *If $\theta \in (1/2, 1)$, there exist $\gamma_1 > 0$ and $\gamma_2 > 0$ such that*

$$|f(\mathbf{x}_c) - f(\mathbf{x}_*)| \leq \gamma_1 c^{-\frac{1}{2\theta-1}} \qquad and \qquad \|\mathbf{x}_c - \mathbf{x}_*\| \leq \gamma_2 c^{-\frac{1-\theta}{2\theta-1}}.$$

Proof Without loss of generality, we assume $\mathbf{x}_0 \in \mathbb{B}(\mathbf{x}_*, \sigma)$. Define

$$\zeta_c := \sum_{\ell=c}^{\infty} \|\mathbf{x}_\ell - \mathbf{x}_{\ell+1}\| \geq \|\mathbf{x}_c - \mathbf{x}_*\|.$$

From Lemma 3.5 and Theorem 3.4, we have

$$\zeta_c \leq \frac{\max\{1, \overline{\alpha}\}}{\eta C(1-\theta)} |f(\mathbf{x}_c) - f(\mathbf{x}_*)|^{1-\theta}$$

$$= \frac{\max\{1, \overline{\alpha}\}}{\eta C^{1/\theta}(1-\theta)} \left(C|f(\mathbf{x}_c) - f(\mathbf{x}_*)|^{\theta}\right)^{(1-\theta)/\theta}$$

$$\leq \frac{\max\{1, \overline{\alpha}\}}{\eta C^{1/\theta}(1-\theta)} \left(\|\mathbf{g}^1(\mathbf{x}_c)\|\right)^{(1-\theta)/\theta}.$$

It can be obtained from Lemma 3.2 and (6) that

$$\|\mathbf{x}_c - \mathbf{x}_{c+1}\| = \rho_c \|\mathbf{g}_c\| \geq \kappa_1 \min\{\underline{\alpha}, 1\} \|\mathbf{g}^1(\mathbf{x}_c)\|.$$

Hence, by denoting $\kappa_2 := \frac{\max\{1,\overline{\alpha}\}}{\eta C^{1/\theta}(1-\theta)}(\kappa_1 \min\{\underline{\alpha}, 1\})^{-(1-\theta)/\theta}$, we get

$$\zeta_c \leq \kappa_2 \|\mathbf{x}_c - \mathbf{x}_{c+1}\|^{(1-\theta)/\theta} = \kappa_2(\zeta_c - \zeta_{c+1})^{(1-\theta)/\theta}. \tag{8}$$

If $\theta = 1/2$, the inequality (8) means

$$\zeta_{c+1} \leq \frac{\kappa_2 - 1}{\kappa_2}\zeta_c.$$

Hence, the first estimation holds with $\gamma = \zeta_0$ and $\mu = (\kappa_2 - 1)/\kappa_2$.

Next, we consider the case $\theta \in (1/2, 1)$. Let $\varphi(t) := t^{-\theta/(1-\theta)}$ be a decreasing function for $t > 0$. It yields from (8) that

$$\begin{aligned}
\kappa_2^{-\theta/(1-\theta)} &\leq \varphi(\zeta_c)(\zeta_c - \zeta_{c+1}) \\
&= \int_{\zeta_{c+1}}^{\zeta_c} \varphi(\zeta_c)\mathrm{d}t \\
&\leq \int_{\zeta_{c+1}}^{\zeta_c} \varphi(t)\mathrm{d}t \\
&= \frac{1-\theta}{1-2\theta}\left(\zeta_{c+1}^{-(2\theta-1)/(1-\theta)} - \zeta_c^{-(2\theta-1)/(1-\theta)}\right),
\end{aligned}$$

which implies

$$\zeta_{c+1}^{-(2\theta-1)/(1-\theta)} - \zeta_c^{-(2\theta-1)/(1-\theta)} \geq \frac{1-2\theta}{1-\theta}\kappa_2^{-\theta/(1-\theta)} := \kappa_3 > 0.$$

Then, we have

$$\zeta_c^{-(2\theta-1)/(1-\theta)} \geq \kappa_3 + \zeta_{c-1}^{-(2\theta-1)/(1-\theta)} \geq \cdots \geq \kappa_3 c + \zeta_0^{-(2\theta-1)/(1-\theta)},$$

which means

$$\zeta_c \leq \left(\kappa_3 c + \zeta_0^{-(2\theta-1)/(1-\theta)}\right)^{-(1-\theta)/(2\theta-1)} \leq (\kappa_3 c)^{-(1-\theta)/(2\theta-1)}.$$

The last estimation holds by taking $\gamma_2 = \kappa_3^{-(1-\theta)/(2\theta-1)}$.

Let $\xi_c := |f(\mathbf{x}_c) - f(\mathbf{x}_*)|$. From (5), (6), and Theorem 3.4, we have

$$\begin{aligned}
\xi_c - \xi_{c+1} &= f(\mathbf{x}_{c+1}) - f(\mathbf{x}_c) \\
&\geq \frac{\eta\kappa_1}{\overline{\alpha}}\|\mathbf{g}_c\|^2 \\
&\geq \frac{\eta\kappa_1(\min\{\underline{\alpha}, 1\})^2}{\overline{\alpha}}\|\mathbf{g}^1(\mathbf{x}_c)\|^2 \\
&\geq \frac{\eta\kappa_1 C^2(\min\{\underline{\alpha}, 1\})^2}{\overline{\alpha}}\xi_c^{2\theta} := \kappa_4\xi_c^{2\theta}.
\end{aligned}$$

Denote $\chi(t) := t^{-2\theta}$ as a decreasing function for $t > 0$. It is deduced from the above inequality that

$$\kappa_4 \leq \chi(\xi_c)(\xi_c - \xi_{c+1}) = \int_{\xi_{c+1}}^{\xi_c}\chi(\xi_c)\mathrm{d}t \leq \int_{\xi_{c+1}}^{\xi_c}\chi(t)\mathrm{d}t = \frac{1}{2\theta-1}\left(\xi_{c+1}^{-(2\theta-1)} - \xi_c^{-(2\theta-1)}\right).$$

Let $\kappa_5 := (2\theta - 1)\kappa_4 > 0$. We have

$$\xi_c^{-(2\theta-1)} \geq \kappa_5 + \xi_{c-1}^{-(2\theta-1)} \geq \cdots \geq \kappa_5 c + \xi_0^{-(2\theta-1)},$$

which means

$$\xi_c \leq \left(\kappa_5 c + \xi_0^{-(2\theta-1)}\right)^{-1/(2\theta-1)} \leq \left(\kappa_5 c\right)^{-1/(2\theta-1)}.$$

The second estimation is valid if we take $\gamma_1 = \kappa_5^{-1/(2\theta-1)}$.

4 Numerical experiments

To evaluate the performance of the proposed algorithm, we implement GPA in MATLAB and use GPA to compute Lagrangians of small and large scale hypergraphs. In our experiments, parameters are set as follows:

$$\eta = 0.01, \beta = 0.5, \alpha_0 = 1, \underline{\alpha} = 0.001, \text{and } \overline{\alpha} = 1,000.$$

The algorithm terminates if

$$\|\pi_\Delta(\mathbf{x}_c + \nabla f(\mathbf{x}_c)) - \mathbf{x}_c\| \leq 10^{-8}, |f(\mathbf{x}_c) - f(\mathbf{x}_{c-4})| \leq 10^{-8},$$

or the number of iteration exceeds one thousand. For each hypergraph, ten random initial points from the legal weighting set Δ are sampled uniformly. We run the GPA algorithm individually from these starting points, and then choose the best one as our solution. We demonstrate the detailed results in the remainder of this section.

4.1 A toy example

First, we consider the toy 3-graph G_{toy} illustrated in Figure 2. The hypergraph G_{toy} has 12 vertices and 13 edges. We compare our method with function "fmincon" in MATLAB optimization tool, which could run the interior point algorithm (IP), sequence quadratic programming (SQP), and the active set method (AS). By solving the optimization, we find that the Lagrangian of G_{toy} is $\lambda(G_{toy}) = \frac{1}{16}$ and the associated optimal solution is

$$\mathbf{x}_1^* = (\tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, 0, 0, 0, 0, 0, 0, 0, 0)^T,$$
$$\mathbf{x}_2^* = (0, 0, 0, 0, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, 0, 0, 0, 0)^T,$$
$$\mathbf{x}_3^* = (0, 0, 0, 0, 0, 0, 0, 0, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4})^T.$$

It is interesting to see that the Lagrangian always finds the maximal cliques contained in G_{toy}. A clique means a complete sub-hypergraph. We will study complete r-graph in the next subsection.

We report the CPU time of four methods: GPA, IP, SQP, and AS for solving the Lagrangian of G_{toy} in Table 1. It can be seen that GPA is at least seven times faster than fmincon. Since SQP is much faster than IP and AS, we employ fmincon only with SQP in the following experiments.

Table 1 CPU time (second).

Methods	GPA	IP	SQP	AS
time (s)	0.11	3.74	0.82	1.66

4.2 Complete hypergraphs

It is well-known that the Lagrangian of a complete r-graph K_n^r with order n has a closed-form solution

$$\lambda(K_n^r) = \binom{n}{r} \frac{1}{n^r}$$

and the associated solution is $\mathbf{x}^* = (\frac{1}{n}, \ldots, \frac{1}{n})^T$. In this experiment, we examine complete 3-graphs with order n varying from 10 to 1000 and associated sizes ranging from 120 to 166,167,000. We compare the results of GPA and SQP via accuracy of Lagrangian value $|\lambda^* - \hat{\lambda}|$ and the error of optimal solution $\|\mathbf{x}^* - \hat{\mathbf{x}}\|_\infty$, where λ^* and $\mathbf{x}^*$ are the exact Lagrangian value and the associated optimal solution, and $\hat{\lambda}$ and $\hat{\mathbf{x}}$ are computed Lagrangian value and computed optimal solution vector.

Table 2 Performance of complete 3-graphs.

n	m	λ^*	GPA			SQP		
			$\|\lambda^* - \hat{\lambda}\|$	$\|\mathbf{x}^* - \hat{\mathbf{x}}\|_\infty$	time (s)	$\|\lambda^* - \hat{\lambda}\|$	$\|\mathbf{x}^* - \hat{\mathbf{x}}\|_\infty$	time (s)
10	120	0.1200	4.0^{-16}	5.1^{-11}	0.14	6.1^{-16}	2.4^{-8}	0.33
18	816	0.1399	1.0^{-15}	8.2^{-11}	0.02	3.1^{-16}	1.2^{-8}	0.11
32	4,960	0.1514	3.7^{-15}	2.2^{-10}	0.04	0.0	9.5^{-9}	0.17
56	27,720	0.1578	8.8^{-15}	8.9^{-11}	0.32	5.6^{-17}	1.2^{-8}	0.65
100	161,700	0.1617	4.2^{-14}	8.4^{-14}	1.39	2.5^{-16}	9.1^{-9}	3.07
178	924,176	0.1639	8.9^{-14}	6.3^{-14}	9.43	5.6^{-17}	1.0^{-8}	25.96
316	5,209,260	0.1651	1.4^{-13}	2.8^{-10}	77.83	9.4^{-16}	8.6^{-9}	187.91
562	29,426,320	0.1658	9.0^{-13}	2.3^{-11}	465.76	9.0^{-13}	1.2^{-8}	1,030.73
1,000	166,167,000	0.1662	1.5^{-12}	2.6^{-8}	5,384.49	1.5^{-12}	1.3^{-9}	6,983.94

Numerical results are illustrated in Table 2. First, it is straightforward to see that the value of Lagrangian increases monotonously as the number of vertices of a complete 3-graph enlarges. Second, since GPA is a feasible optimization method, GPA obtains better solution errors than SQP. Finally, GPA runs faster than SQP when comparing CPU times.

4.3 Sparse hypergraphs

In this subsection, we focus on 3-graphs defined on a sphere. As illustrated in Figure 3(a), the icosahedron ($\ell = 0$) has 12 vertices and 20 faces. Obviously,

each face is a triangle. To approximate the sphere, we subdivide each triangle of the icosahedron into four triangles and obtain the polyhedron in Figure 3(b) with $\ell = 1$. By recursively subdividing the triangles, we produce polyhedrons in Figure 3(c) and (d) with $\ell = 2$ and 3, respectively. Then, for each ℓ, a sparse 3-graph is formed by using the vertex set and the face set of the ℓth polyhedron.
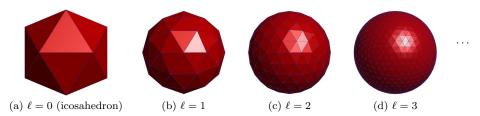

(a) $\ell = 0$ (icosahedron)   (b) $\ell = 1$ (c) $\ell = 2$ (d) $\ell = 3$

Fig. 3 Sparse 3-graphs.

Table 3 Numerical results of sparse 3-graphs.

ℓ	n	m	GPA		SQP	
			$\hat{\lambda}$	time (s)	$\hat{\lambda}$	time (s)
0	12	20	0.037037	0.19	0.037037	0.64
1	42	80	0.037037	0.02	0.037037	0.14
2	162	320	0.037037	0.03	0.037037	0.93
3	642	1,280	0.037037	0.09	0.037037	39.41
4	2,562	5,120	0.037037	0.17	0.037037	813.32
5	10,242	20,480	0.037037	1.02	0.037037	64,759.19
6	40,962	81,920	0.037037	3.13	– –	
7	163,842	327,680	0.037037	16.23	– –	
8	655,362	1,310,720	0.037037	62.99	– –	

Here, "– –" means that CPU times exceeds 24 hours.

The computation results of Lagrangians as well as CPU time are reported in Table 3. No matter how many vertices and edges are involved in the 3-graphs, it seems that all Lagrangian values are about $\hat{\lambda} = 1/27 \approx 0.037037$. However, we did not find a solid proof which covers the Lagrangians of sparse hypergraphs illustrated in Figure 3. We note that these sparse hypergraphs are not regular, i.e., degrees of vertices could be five and six, when $\ell \geq 1$. The CPU time shows that our GPA method is thousands times faster than SQP when we solve the sparse 3-graphs with 2,562 vertices. Furthermore, the GPA method is capable of computing the Lagrangian of hypergraphs with millions of edges, which means GPA is powerful for calculating Lagrangians of large scale hypergraphs.

5 Conclusions

The Lagrangian of a hypergraph reveals special nature of the adjacency tensor of a uniform hypergraph. In this paper, we designed a gradient projection algorithm for computing the Lagrangian of a uniform hypergraph numerically. Global and local convergence of the proposed algorithm was analyzed. Preliminary numerical experiments illustrated that the proposed algorithm is efficient for large scale hypergraphs.

Data Availability All data generated or analyzed during this study are included in this manuscript.

References

1. Attouch, H., Bolte, J.: On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. Math. Program. **116**(1), 5–16 (2009). https://doi.org/10.1007/s10107-007-0133-5

2. Attouch, H., Bolte, J., Redont, P., Soubeyran, A.: Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka–Łojasiewicz inequality. Math. Oper. Res. **35**(2), 438–457 (2010). https://doi.org/10.1287/moor.1100.0449

3. Barzilai, J., Borwein, J. M.: Two-point step size gradientmethods. IMA J. Numer. Anal. **8**, 141–148 (1988). https://doi.org/10.1093/imanum/8.1.141

4. Benson, A.R., Gleich, D.F., Leskovec, J.: Higher-order organization of complex networks. Science **353**, 163–166 (2016). https://doi.org/10.1126/science.aad9029

5. Bertsekas, D.P.: Nonlinear Programming, 3rd ed., Athena Scientific, Belmont (2016).

6. Bolte, J., Daniilidis, A., Lewis A.: The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. SIAM J. Optim. **17**(4), 1205–1223 (2007). https://doi.org/10.1137/050644641

7. Bulò, S.R., Pelillo, M.: A game-theoretic approach to hypergraph clustering. IEEE Trans. Pattern Anal. Mach. Intell. **35**, 1312–1327 (2013). https://doi.org/10.1109/TPAMI.2012.226

8. Chang, J., Chen, Y., Qi, L.: Computing eigenvalues of large scale sparse tensors arising from a hypergraph. SIAM J. Sci. Comput. **38**, A3618–A3643 (2016). https://doi.org/10.1109/10.1137/16M1060224

9. Chang, J., Chen, Y., Qi, L., Yan, H.: Hypergraph clustering using a new Laplacian tensor with applications in image processing. SIAM J. Imaging Sci. **13**(3), 1157–1178 (2020). https://doi.org/10.1137/19M1291601

10. Chang, J., Ding, W., Qi, L., Yan, H.: Computing the p-spectral radii of uniform hypergraphs with applications. J. Sci. Comput. **75**, 1–25 (2018). https://doi.org/10.1007/s10915-017-0520-x

11. Chen, Y., Qi, L., Zhang, X.: The Fiedler vector of a Laplacian tensor for hypergraph partitioning. SIAM J. Sci. Comput. **39**(6), A2508–A2537 (2017). https://doi.org/10.1137/16M1094828

12. Chen, Y., Ye, X.: Projection onto a simplex. arXiv:1101.6081v2, 1–7 (2011). https://doi.org/10.48550/arXiv.1101.6081

13. Cooper, J., Dutle, A.: Spectra of uniform hypergraphs. Linear Algebra Appl. **436**, 3268–3292 (2012). https://doi.org/10.1016/j.laa.2011.11.018

14. Frankl, P., Füredi, Z.: Extremal problems whose solutions are the blowups of the small witt-designs. J. Comb. Theory Ser. A **52**(1), 129–147 (1989). https://doi.org/10.1016/0097-3165(89)90067-8

15. Gruslys, V., Letzter, S., Morrison, N.: Hypergraph Lagrangians I: The Frankl-Füredi conjecture is false. Adv. Math. **365**, 107063 (2020). https://doi.org/10.1016/j.aim.2020.107063

16. Gruslys, V., Letzter, S., Morrison, N.: Lagrangians of hypergraphs II: When colex is best. Isr. J. Math. **242**, 637–662 (2021). https://doi.org/10.1007/s11856-021-2132-2

17. Hager, W.W., Zhang, H.: A new active set algorithm for box constrained optimization. SIAM J. Optim. **17**(2), 526–557 (2006). https://doi.org/10.1137/050635225

18. Hou, J., Pelillo, M., Yuan, H.: Hypergraph matching via game-theoretic hypergraph clustering. Pattern Recognit. **125**, 108526 (2022). https://doi.org/10.1016/j.patcog.2022.108526

19. Hu, S., Li, G.: Convergence rate analysis for the higher order power method in best rank one approximations of tensors. Numer. Math. **140**, 993–1031 (2018). https://doi.org/10.1007/s00211-018-0981-3

20. Hu, S., Peng, Y., Wu, B.: Lagrangian densities of linear forests and Turán numbers of their extensions. J. Combin. Des. **28**, 207–223 (2020). https://doi.org/10.1002/jcd.21687

21. Huang, Y., Dai, Y., Liu, X.: Equipping the Barzilai–Borwein method with the two dimensional quadratic termination property. SIAM J. Optim. **31**, 3068–3096 (2021). https://doi.org/10.1137/21M1390785

22. Koutsoukos, D., Nakandala, S., Karanasos, K., Saur, K., Alonso, G., Interlandi, M.: Tensors: an abstraction for general data processing. Proc. VLDB Endow. **14**(10), 1797–1804 (2021). https://doi.org/10.14778/3467861.3467869

23. Łojasiewicz, S.: Une propriété topologique des sous-ensembles analytiques réels. Les Équations aux Dérivées Partielles, 87–89 (1963).

24. Lu, X., Zhang, X.: A note on Lagrangians of 4-uniform hypergraphs. Ars Combin. **121**, 329–340 (2015).

25. Luo, X., Peng, J., Liang, J.: Directed hypergraph attention network for traffic forecasting. IET Intell. Transp. Syst. **16**, 85–98 (2022). https://doi.org/10.1049/itr2.12130

26. Motzkin, T.S., Straus, E.G.: Maxima for graphs and a new proof of a theorem of Turán. Can. J. Math. **17**, 533–540 (1965). https://doi.org/10.4153/cjm-1965-053-6

27. Qi, L., Luo, Z.: Tensor Analysis: Spectral Theory and Special Tensors, SIAM, Philadelpia (2017). https://doi.org/10.1137/1.9781611974751

28. Sun, Y, Tang, Q., Zhao, C., Peng, Y.: On the largest graph-Lagrangian of 3-graphs with fixed number of edges. J. Optim. Theory Appl. **163**, 57–79 (2014). https://doi.org/10.1007/s10957-013-0519-x

29. Talbot, J.M.: Lagrangians of hypergraphs. Comb., Probab. Comput. **11**(2), 199–216 (2022). https://doi.org/10.1017/s0963548301005053

30. Wang, W., Carreira-Perpiñán, M.Á.: Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. arXiv:1309.1541v1, 1–5 (2013). https://doi.org/10.48550/arXiv.1309.1541

31. Zhang, D., Luo, Y., Yu, Y., Zhao, Q., Zhou, G.: Semi-supervised multi-view clustering with dual hypergraph regularized partially shared non-negative matrix factorization. Sci. China Technol. Sci. **65**, 1349–1365 (2022). https://doi.org/10.1007/s11431-021-1957-3