East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.2023-184.151023

Vol. **15**, No. 1, pp. 1-28
February 2025

An ODE Approach to Multiple Choice Polynomial Programming

Sihong Shao* and Yishan Wu

CAPT, LMAM and School of Mathematical Sciences, Peking University, Beijing 100871, China.

Received 31 July 2023; Accepted (in revised version) 15 October 2023.

Abstract. We propose an ODE approach to solving multiple choice polynomial programming (MCPP) after assuming that the optimum point can be approximated by the expected value of so-called thermal equilibrium as usually did in simulated annealing. The explicit form of the feasible region and the affine property of the objective function are both fully exploited in transforming an MCPP problem into an ODE system. We also show theoretically that a local optimum of the former can be obtained from an equilibrium point of the latter. Numerical experiments on two typical combinatorial problems, MAX-k-CUT and the calculation of star discrepancy, demonstrate the validity of the ODE approach, and the resulting approximate solutions are of comparable quality to those obtained by the state-of-the-art heuristic algorithms but with much less cost. When compared with the numerical results obtained by using Gurobi to solve MCPP directly, our ODE approach is able to produce approximate solutions of better quality in most instances. This paper also serves as the first attempt to use a continuous algorithm for approximating the star discrepancy.

AMS subject classifications: 90C59, 35Q90, 90C09, 90C23
Key words: Pseudo-Boolean optimization, multiple choice constraint, continuous approach, MAX-CUT, star discrepancy.

1. Introduction

We consider the following pseudo-Boolean optimization problem:

$$\begin{aligned} \min_{x\in\{0,1\}^n} \quad & f(x), \\ \text{s.t.} \quad & \sum_{i\in I_j} x_i = 1, \quad j = 1,2,\ldots,m, \end{aligned} \tag{1.1}$$

where f is a polynomial function, x is an n-dimensional Boolean vector, the indices $[n] := \{1,2,\ldots,n\}$ are divided into m disjoint subsets $I_1, I_2, \ldots, I_m$, and the cardinality of each I_j, denoted by $d_j := |I_j|$, must be greater than 1. Then x is accordingly divided into m vectors

*Corresponding author. *Email addresses:* sihong@math.pku.edu.cn (S. Shao), wuyishan@pku.edu.cn (Y. Wu)

$x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ where each $x^{(j)}$ picks out all the entries in I_j of x. The constraints mean that, for each j, there exists exact one element that equals to 1 in the subvector $x^{(j)}$, implying that only one is determined from d_j choices and thus m items chosen from n choices in total. A group of entries of x in a single I_j represents a decision out of finite choices. More precisely, we use multiple choice polynomial programming to call the problem (1.1), which is capable of dealing with various problems in diverse disciplines [2], for example, the MAX-k-CUT [20], star discrepancy [8] problems and SAT [17]. It should be pointed out that studies on integer linear programming problems with multiple choice constraints, termed the multiple choice programming (MCP), can date back to [11]. Since then several methods have been developed — cf. [27], but most of them are not designed for nonlinear objective functions. Although MCPP can be transformed into MCP by defining new variables to represent monomials, an exploration in this direction will not be presented here.

As a typical 0-1 programming problem, MCPP can also be treated with standard mixed integer nonlinear programming solvers, such as Gurobi [10], IBM-CPLEX [14], and SCIP [1], but few of them are designed for general nonlinear programming. For example, to apply Gurobi, one has to reformulate the nonlinear terms in MCPP instances into linear and/or quadratic forms by defining new variables and new constraints, thereby greatly increasing the problem size. This defect becomes even severer in calculating the star discrepancy since the degree of corresponding objective function is nothing but the dimension of underlying space and is usually much greater than 1 (see Eq. (6.13) in Section 6).

Alternatively, continuous approaches to discrete problems has attracted more attention since the Hopfield network — cf. [13], an early ODE approach, was proposed for the 0-1 quadratic programming in 1980s. The Hopfield network has been extended to other combinatorial optimization problems [15,28], where many useful mathematical techniques in dynamical system have been introduced. In the most recent work [22], an ODE approach was proposed through a quartic penalty approximation of the Boolean polynomial program. In line with this, here we propose to use the solutions of the following ODE system:

$$\begin{aligned} &\frac{\mathrm{d}y_i^{(j)}}{\mathrm{d}t} = -y_i^{(j)} + \sigma_i\left(-\Phi^{(j)}(y); 1/T\right), \quad j \in [m], \quad i \in [d_j], \\ &y(0) = y_0 \in [0,1]^n \end{aligned} \tag{1.2}$$

to approximate the solutions of MCPP, where the time-dependent vector $y(t) : [0, +\infty) \to \mathbb{R}^n$ is divided as x in Eq. (1.1) does, the initial data y_0 is required to satisfy the continuous multiple choice constraint: $\sum_{i=1}^{d_j} (y_0)_i^{(j)} = 1$ for all $j \in [m]$, $\Phi^{(j)}$ denotes the partial derivative of f with respect to $x^{(j)}$: $\Phi_i^{(j)} = \partial f / \partial x_i^{(j)}$, T is a positive parameter called temperature, and $\sigma_i(z;\beta) : \mathbb{R}^d \times \mathbb{R}_+ \to (0,1)$ gives the softmax function defined by

$$\sigma_i(z;\beta) = \frac{\exp(\beta z_i)}{\sum_{k=1}^{d} \exp(\beta z_k)} \tag{1.3}$$

with d being the dimension of input argument z. Note in passing that $y_i^{(j)}(t) \in (0,1)$ for arbitrary $t > 0$, i.e. $y(t) : [0, +\infty) \to (0,1)^n$. We are able to prove that the equilibrium points of the ODE system (1.2) represent the local optimum solutions of MCPP (1.1).

Therefore, we just have to numerically integrate (1.2) until we find an equilibrium (usually fast with fewer iterations) which can be rounded into a local solution of MCPP. Along the way, various well-established techniques in numerically solving ODEs can be incorporated to improve the efficiency without sacrificing the solution quality. Actually, the ODE system (1.2) can be regarded as a continuous version of simulated annealing (SA) [17] by starting from a continuous-time Markov chain and exploiting two intrinsic properties of MCPP (1.1). One is that the explicit form of the feasible region is so straightforward that the state space of the Markov chain can be easily determined. The other is that the objective function of MCPP, $f(x^{(1)}, x^{(2)}, \ldots, x^{(m)})$, is affine with respect to $x^{(j)}$ for all $j \in [m]$ (otherwise we can modify it equivalently), with which the definition of transition rate can be significantly simplified. It should be noted that Eq. (1.2) may recover the ODE system used in [23] for the unconstrained binary quadratic programming — cf. Example 2.1.

We apply the proposed ODE approach into two typical NP-hard problems: MAX-k-CUT with $k = 2, 3, 4, 5$ and the star discrepancy calculation, and two state-of-the-art heuristic algorithms: the multiple operator heuristic (MOH) [20] and the improved threshold accepting (TA_improved) [8] methods are employed to as the reference of the solution quality, respectively. For MAX-k-CUT problems, the test bed is G-Set[†]. More than half of the ratios between the best cut values achieved by our ODE approach and those by MOH are above 0.99. For star discrepancy calculation, the test set is a group of good lattice point (GLP) sets adopted in [29]. The ratios between the best lower bounds achieved by our ODE approach and those by TA_improved are at least 0.91. More important, in both problems the ODE approach requires much fewer iterations compared to MOH or TA_improved for each trial while the cost of each iteration is linear to the problem size across these three methods. Its iteration steps over MOH's for MAX-k-CUT are about $1/10^5$ to $1/10^4$, and about $1/10^2$ to $1/10$ over TA_improved's for calculating the star discrepancy.

To further demonstrate the capability of the proposed ODE approach, the numerical results obtained by using Gurobi to directly solve the MCPP instances are adopted as comparison for which the time limit of Gurobi is set to be the very runtime of the ODE approach. A detailed numerical comparison shows that, (1) for MAX-k-CUT with $k = 3, 4, 5$, in more than 80% instances, the solutions produced by the ODE approach are better than or as good as Gurobi's; (2) for the star discrepancy problem, in about 17% instances, Gurobi does not even provide a feasible solution under the time limitation, and in the remaining instances, it provides no better solutions than our ODE approach.

The paper is organized as follows. In Section 2 we show how transform the MCPP problem (1.1) into the ODE system (1.2) step by step starting from the well-known assumption (2.2) of simulated annealing. In Section 3 we prove that a local optimum of MCPP (1.1) can be indeed obtained via an equilibrium point of the ODE system (1.2). After that, Section 4 details the procedure of finding the equilibrium points by numerically integrating Eq. (1.2). In Sections 5 and 6, we apply the proposed ODE approach into the MAX-k-CUT problem and approximating the star discrepancy, respectively. The paper is concluded in Section 7 with a few remarks.

[†]Available at https://web.stanford.edu/ yyye/yyye/Gset/

2. The ODE Approach: A Continuous Version of Simulated Annealing

Under the multiple choice constraint in Eq. (1.1), $x^{(j)}$ must be a standard unit vector in $\mathbb{R}^{d_j}$. Let $B_d = \{e_i^d\}_{i=1}^d \subset \mathbb{R}^d$ collects the standard base of $\mathbb{R}^d$ where e_i^d denotes the d-dimensional unit vector the i-th entry of which equals 1 and the rests are zeros. The multiple choice constraint is then equivalent to $x^{(j)} \in B_{d_j}, j = 1, 2, \ldots, m$. Therefore, the feasible region for MCPP is

$$\mathscr{X} := B_{d_1} \times B_{d_2} \times \cdots \times B_{d_m}. \tag{2.1}$$

We should mention that it is d_j, the cardinality of each I_j, that determines the structure of the feasible region, rather than the particular division, and a compact notation, $B_d^m := \mathscr{X}$, will be used in Sections 5 and 6 when $d_1 = \cdots = d_m = d$. Then MCPP (1.1) becomes $\min_{x\in\mathscr{X}} f(x)$ starting from which we explain how regard the ODE system (1.2) as a continuous version of SA. The well-known SA method [17] approaches "thermal equilibrium" by generating a Markov chain such that its stationary distribution is the Boltzmann distribution, namely

$$\Pr(X = x) \propto \exp\big(-E(x)/T\big), \tag{2.2}$$

where $X \in \mathscr{X}$ is a random variable, and $E(x)$ denotes the energy at state x. We set $E(x) := f(x)$ for MCPP, adopt a similar approach as SA but in a continuous, ODE-type way, by defining a continuous-time Markov chain, and then calculate the dynamics of expected value through the forward equation.

To define a random variable of a continuous-time Markov chain, we need to determine two things: the state space of the variable and the transition rate between states. For MCPP, the former is nothing but the feasible region (2.1). Let X_t with $t \geqslant 0$ be the random variable. The state change of X_t is allowed to happen only in a single B_j. That is, X_t changes between $x = (x^{(1)}, x^{(2)}, \ldots, x^{(m)})$ and x', where there exists $j \in [m]$ and $i' \in [d_j]$ such that $x' = (x^{(\bar{j})}, e_{i'}^{d_j})$. Here $x^{(\bar{j})} := (x^{(1)}, \ldots, x^{(j-1)}, x^{(j+1)}, \ldots, x^{(m)})$ denotes the $(n - d_j)$-dimensional vector by deleting $x^{(j)}$, and $(x^{(\bar{j})}, \xi) := (x^{(1)}, \ldots, x^{(j-1)}, \xi, x^{(j+1)}, \ldots, x^{(m)})$ the n-dimensional vector by replacing $x^{(j)}$ with $\xi \in \mathbb{R}^{d_j}$. We are ready to determine the transition rate. Before that, we would like to assume that $f(x^{(1)}, x^{(2)}, \ldots, x^{(m)})$ is affine with respect to $x^{(j)}$ after fixing $x^{(\bar{j})}$ for all $j \in [m]$

$$f(x) = x^{(j)} \cdot \Phi^{(j)}\big(x^{(\bar{j})}\big) + f\big((x^{(\bar{j})}, 0)\big), \tag{2.3}$$

where the dot $\cdot$ gives the standard inner product, and the second RHS term is independent of $x^{(j)}$. Eq. (2.3) can be easily verified if noting:

(1) Every $(x_i^{(j)})^k$ in f can be replaced by $x_i^{(j)}$ because the power of 0 or 1 is equal to itself.

(2) Every monomial with divisor $x_i^{(j)} x_{i'}^{(j)}$, $i \neq i'$, vanishes.

(3) The partial derivative $\Phi^{(j)}(x)$ is independent of $x^{(j)}$ and thus $\Phi^{(j)}(x^{(\bar{j})}) = \Phi^{(j)}(x)$ is defined unambiguously.

For $x = (x^{(\bar{j})}, e_i^{d_j})$, $x' = (x^{(\bar{j})}, e_{i'}^{d_j})$, $i \neq i'$, according to Eq. (2.3), we have

$$f(x) = f\big((x^{(\bar{j})}, 0)\big) + \Phi_i^{(j)}(x^{(\bar{j})}), \quad f(x') = f\big((x^{(\bar{j})}, 0)\big) + \Phi_{i'}^{(j)}(x^{(\bar{j})}),$$

and

$$-f(x') + f(x) = -\Phi_{i'}^{(j)}(x^{(\bar{j})}) + \Phi_i^{(j)}(x^{(\bar{j})}). \tag{2.4}$$

Let $q(x \to x')$ denote the transition rate from state x to x'. It also means (x, x')-entry of the Q-matrix over $\mathbb{R}^{\mathscr{X}\times\mathscr{X}}$. The detailed balance condition reads

$$\exp\big(-f(x)/T\big)q\big(x \to x'\big) = \exp\big(-f(x')/T\big)q\big(x' \to x\big),$$

or

$$\frac{q(x \to x')}{q(x' \to x)} = \exp\big((-f(x') + f(x))/T\big) \tag{2.5}$$

for non-zero entries of q, which implies $x \neq x'$. To satisfy Eq. (2.5), combining it with Eq. (2.4), we define

$$q(x \to x') = \begin{cases} \sigma_{i'}\big(-\Phi^{(j)}(x^{(\bar{j})}); 1/T\big), & x = \big(x^{(\bar{j})}, e_i^{d_j}\big), \quad x' = \big(x^{(\bar{j})}, e_{i'}^{d_j}\big), \quad i \neq i', \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

Using $x'^{(\bar{j})} = x^{(\bar{j})}$, we have

$$q(x' \to x) = \sigma_i\Big(-\Phi^{(j)}(x'^{(\bar{j})}); 1/T\Big) = \sigma_i\Big(-\Phi^{(j)}(x^{(\bar{j})}); 1/T\Big),$$

and then Eq. (2.5) can be readily verified.

Next, let us calculate the dynamics of expected value. We denote the probability distribution at time t by $p(x;t) := \Pr(X_t = x)$. Then it satisfies the forward equation

$$\frac{\mathrm{d}p(x;t)}{\mathrm{d}t} = \sum_{x'\in\mathscr{X}, x'\neq x} q(x' \to x)p(x';t) - \sum_{x'\in\mathscr{X}, x'\neq x} q(x \to x')p(x;t). \tag{2.7}$$

Let $\mu : \mathbb{R}^n \to \mathbb{R}$ be an arbitrary vector function. From Eq. (2.7), we have

$$\begin{aligned} \frac{\mathrm{d}\langle\mu(X_t)\rangle}{\mathrm{d}t} &= \sum_{x\in\mathscr{X}} \mu(x)\frac{\mathrm{d}p(x;t)}{\mathrm{d}t} \\ &= \sum_{x'\in\mathscr{X}} p(x';t) \sum_{x\neq x'} \mu(x)q(x' \to x) - \sum_{x\in\mathscr{X}} p(x;t) \sum_{x'\neq x} \mu(x)q(x \to x') \\ &= \left\langle \sum_{x\neq X_t} (\mu(x) - \mu(X_t))q(X_t \to x) \right\rangle, \end{aligned} \tag{2.8}$$

where the term in the summation only counts when both $\mu(x) \neq \mu(X_t)$ and $x \neq X_t$ hold. Let $m_t = \langle X_t\rangle \in [0,1]^n$ be the expected value of X_t. Then $(m_t)_i^{(j)} = \sum_{x\in\mathscr{X}} x_i^{(j)} p(x;t)$ which

assigns μ to $(\cdot)_i^{(j)} : x \mapsto x_i^{(j)}$ where there are only few xs satisfies these two conditions. Recalling the definition of q in Eq. (2.6), $x \neq x'$ if and only if x and x' differs in only one group I_k. In addition, we need $x_i^{(j)} \neq (x')_i^{(j)}$ then k must be j. Therefore, $x = ((X_t)^{(\bar{j})}, e_{i'}^{d_j})$ for some $e_{i'}^{d_j} \neq (X_t)^{(j)}$. Accordingly, from Eqs. (2.8) and (2.6), we obtain

$$\begin{aligned}
\frac{\mathrm{d}(m_t)_i^{(j)}}{\mathrm{d}t} &= \left\langle \sum_{1\leqslant i'\leqslant d_j, e_{i'}^{d_j}\neq (X_t)^{(j)}} \left(x_i^{(j)} - (X_t)_i^{(j)}\right) q(X_t \to x) \right\rangle \\
&= \left\langle \sum_{1\leqslant i'\leqslant d_j, e_{i'}^{d_j}\neq (X_t)^{(j)}} \left(x_i^{(j)} - (X_t)_i^{(j)}\right) \sigma_{i'}\left(-\Phi^{(j)}(x); 1/T\right) \right\rangle \\
&= \left\langle \sum_{1\leqslant i'\leqslant d_j} \left(x_i^{(j)} - (X_t)_i^{(j)}\right) \sigma_{i'}\left(-\Phi^{(j)}(x^{(\bar{j})}); 1/T\right) \right\rangle \\
&= \left\langle \sum_{1\leqslant i'\leqslant d_j} x_i^{(j)} \sigma_{i'}\left(-\Phi^{(j)}((X_t)^{(\bar{j})}); 1/T\right) \right\rangle \\
&\quad - \left\langle \sum_{1\leqslant i'\leqslant d_j} (X_t)_i^{(j)} \sigma_{i'}\left(-\Phi^{(j)}(x^{(\bar{j})}); 1/T\right) \right\rangle \\
&= \left\langle \sigma_i\left(-\Phi^{(j)}(X_t); 1/T\right) \right\rangle - \left\langle (X_t)_i^{(j)} \right\rangle \qquad (2.9) \\
&\approx \sigma_i\left(-\Phi^{(j)}(m_t); 1/T\right) - (m_t)_i^{(j)}, \qquad (2.10)
\end{aligned}$$

where we have applied $x^{(\bar{j})} = (X_t)^{(\bar{j})}$, and $x_i^{(j)} \neq 0$ if and only if $i' = i$ in the first RHS term of Eq. (2.9), $\sum_{1\leqslant i'\leqslant d_j} \sigma_{i'}(-\Phi^{(j)}(x); 1/T) = 1$ in the second RHS term of Eq. (2.9), as well as a rough approximation $\langle \sigma_i(-\Phi^{(j)}(X_t); 1/T)\rangle \approx \sigma_i(-\Phi^{(j)}(m_t); 1/T)$ in the first RHS term of Eq. (2.10). The ODE system (1.2) is manifest in Eq. (2.10) after replacing m_t with $y(t) \in (0,1)^n$. That is, $y(t)$ is also an approximation of the dynamics of the expected value of a continuous-time Markov chain, and by searching for its efficient numerical approximations, we are able to approximate the Markov chain efficiently. Moreover, its equilibrium serves as the stable distribution of the Markov chain, namely, the Boltzmann distribution (2.2). When the temperature goes to zero, the distribution approaches the uniform distribution on the ground states, which may give the optimum solution. Therefore, the equilibrium gives a reasonable approximation of the optimum, the proof of which is left for Section 3.

Finally, $X_t \in \mathscr{X}$ means that the multiple choice constraint $\sum_{i=1}^{d_j} (X_t)_i^{(j)} = 1$ for all $j \in [m]$ holds with probability 1 and so does m_t. In fact, we claim that $y(t)$ defined in Eq. (1.2) also satisfies such constraint when the initial data $y(0)$ does, which can be readily seen from the sum of Eq. (1.2),

$$\frac{\mathrm{d}}{\mathrm{d}t} S_j(t) = -S_j(t) + 1 \quad \Leftrightarrow \quad S_j(t) = \left(S_j(0) - 1\right)\mathrm{e}^{-t} + 1,$$

where $S_j(t) = \sum_{i=1}^{d_j} y_i^{(j)}(t)$.

Example 2.1. The ODE approach is applicable in the unconstrained situation where the multiple choice constraint in Eq. (1.1) is absent. By introducing extra n Boolean variables $x_{n+1},\dots,x_{2n}$, the unconstrained problem $\min_{x\in\{0,1\}^n} f(x)$ can be reformulated into the following MCPP:

$$\begin{aligned}\min_{x\in\{0,1\}^{2n}} \quad & f(x_1,x_2,\dots,x_n),\\ \text{s.t.} \quad & x_i + x_{i+n} = 1, \quad i = 1,2,\dots,n,\end{aligned}$$

where the index set $[2n]$ are divided into n disjoint subsets: $I_j = \{j, j+n\}, d_j = 2$, $j = 1,2,\dots,n$, and $\Phi_1^{(j)}(x) = \partial f(x)/\partial x_j$, $\Phi_2^{(j)}(x) = 0$. Then from Eq. (1.2), we have that y_j (i.e., $y_1^{(j)}$) satisfies

$$\frac{\mathrm{d}y_j}{\mathrm{d}t} = -y_j + \frac{1}{2}\left(\tanh\left(-\frac{1}{2T}\frac{\partial f(y)}{\partial x_j}\right)+1\right), \quad j = 1,2,\dots,n. \tag{2.11}$$

As

$$y_1^{(j)}(t) + y_2^{(j)}(t) = y_j(t) + y_{j+n}(t) = 1 \quad \text{for} \quad t \geqslant 0,$$

we do not have to care about the dynamics of extra variable $y_{j+n}(t)$. Implementing a linear transformation $z_j = 2y_j - 1$ in Eq. (2.11) yields a ODE system for $z \in [-1,1]^n$

$$\frac{\mathrm{d}z_j}{\mathrm{d}t} = -z_j + \tanh\left(-\frac{1}{2T}\frac{\partial f((z+1)/2)}{\partial x_j}\right), \quad j = 1,2,\dots,n,$$

which was also mentioned in [23] for the unconstrained binary quadratic programming.

3. Numerical Analysis

Let $\bar{y}(T)$ be an equilibrium point of the ODE system (1.2) under the parameter T, namely,

$$\bar{y}(T)^{(j)} = \sigma\left(-\Phi^{(j)}(\bar{y}(T)); 1/T\right) \quad \text{for all} \quad j \in [m]. \tag{3.1}$$

We claim that $\bar{y}(T)$ approximates a local solution of Eq. (1.1) when T is sufficiently small. This can be formally explained as follows. As β approaches $+\infty$, the limit of softmax function (1.3), denoted by $\hat{\sigma}(z) = (\hat{\sigma}_i(z))$, is a kind of hard max

$$\hat{\sigma}_i(z) = \lim_{\beta\to+\infty} \sigma_i(z;\beta) = \begin{cases} 1/r, & z_i = \max\{z_1,z_2,\dots,z_d\},\\ 0, & \text{otherwise},\end{cases} \tag{3.2}$$

where $r = |\{i \mid z_i = \max\{z_1,z_2,\dots,z_d\}\}|$ denotes the number of maximal entries of z. It can be easily observed that the range of $\hat{\sigma}(z)$, denoted by $\bar{B}_d$, is a discrete set, i.e.,

$$\bar{B}_d = \left\{ \frac{1}{|A|}\chi^A \in \mathbb{R}^d \;\middle|\; \chi_i^A = \begin{cases} 1, & i \in A,\\ 0, & i \notin A,\end{cases} \quad A \subseteq [d],\ A \neq \varnothing \right\}.$$

If the limit $\hat{y} := \lim_{T\to 0^+} \bar{y}(T)$ exists, then Eq. (3.1) formally yields

$$\hat{y}^{(j)} = \hat{\sigma}\left(-\Phi^{(j)}(\hat{y})\right) \tag{3.3}$$

for all $j \in [m]$. Thus $\hat{y}^{(j)} \in \bar{B}_{d_j}$ and

$$\hat{y} \in \bar{\mathscr{X}} := \bar{B}_{d_1} \times \bar{B}_{d_2} \times \cdots \times \bar{B}_{d_m}.$$

Combining the Eqs. (3.2) and (3.3) shows that the nonzero entries in $\hat{y}^{(j)}$ correspond to the minimal entries of $\Phi^{(j)}(\hat{y})$. Hence, $\hat{y}^{(j)}$ minimizes the MCPP objective function $f((\hat{y}^{(\bar{j})}, \xi))$ for $\xi \in \mathbb{R}^{d_j}$ after fixing $\hat{y}^{(\bar{j})}$ in view of its affine property given in Eq. (2.3). Actually, given Definition 3.1, we are able to prove that $\hat{y}$ is indeed a local optimum of MCPP (1.1) in $\bar{\mathscr{X}}$, cf. Proposition 3.1.

Definition 3.1 (Local Optimality)**.** *We call $x \in \bar{\mathscr{X}}$ a local optimum of f in $\bar{\mathscr{X}}$, if for any $x' \in \bar{\mathscr{X}}$ for which exists $j_0 \in [m]$ such that $(x')^{(\bar{j}_0)} = x^{(\bar{j}_0)}$, we have $f(x') \geqslant f(x)$.*

Proposition 3.1. *The n-dimensional vector $\hat{y}$ defined in Eq.* (3.3) *is a local optimum of MCPP* (1.1) *in $\bar{\mathscr{X}}$.*

Proof. Let $y' \in \bar{\mathscr{X}}$ satisfy $(y')^{(\bar{j}_0)} = \hat{y}^{(\bar{j}_0)}$. It follows from the Eqs. (3.2), (3.3) that for all i such that $\hat{y}_i^{(j_0)} \neq 0$, we have

$$\hat{y}_i^{(j_0)} = \frac{1}{r}, \quad \Phi_i^{(j_0)}(\hat{y}^{(\bar{j}_0)}) = \min_{1\leqslant k\leqslant d_{j_0}} \Phi_k^{(j_0)}(\hat{y}^{(\bar{j}_0)}), \tag{3.4}$$

where r equals the number of nonzero entries of $\hat{y}^{(j_0)}$. Combining Eqs. (2.3) and (3.4) yields

$$\begin{aligned} f(y') - f(\hat{y}) &= (y')^{(j_0)} \cdot \Phi^{(j_0)}\left((y')^{(\bar{j}_0)}\right) - \hat{y}^{(j_0)} \cdot \Phi^{(j_0)}(\hat{y}^{(\bar{j}_0)}) \\ &= (y')^{(j_0)} \cdot \Phi^{(j_0)}(\hat{y}^{(\bar{j}_0)}) - \min_{1\leqslant k\leqslant d_{j_0}} \Phi_k^{(j_0)}(\hat{y}^{(\bar{j}_0)}) \geqslant 0, \end{aligned}$$

where the fact $(y')^{(j_0)} \in \bar{B}_{d_{j_0}}$ is used in the last inequality. □

It should be noted that Eq. (3.3) does not hold in general since $\sigma(\cdot;\beta)$ does not converge to $\hat{\sigma}$ uniformly. However, in numerical experiments, the equilibriums $\bar{y}(T)$, which may not be close to a Boolean vector, are always very close to vectors in $\bar{\mathscr{X}}$ for sufficiently small T. Therefore, we can reasonably assume that $\bar{y}$ is an approximation of $\hat{y}$ and that $\sigma(\cdot;1/T)$ is an approximation of $\hat{\sigma}$. Then Eq. (3.1) approximates Eq. (3.3). In fact, the theoretical results exists and in Proposition 3.2 we give the analytical condition for the closeness between $\bar{y}$ and $\bar{\mathscr{X}}$ under which we can claim that Eq. (3.3) holds. Thus, by Proposition 3.1, $\hat{y}$ is a local optimum. Before that, we give some notations and definitions.

Let L be the maximal Lipschitz constant of all $\Phi_i^{(j)}$ with $j \in [m]$, $i \in [d_j]$. Namely, we have

$$\left|\Phi_i^{(j)}(x) - \Phi_i^{(j)}(x')\right| \leqslant L\|x - x'\|_\infty$$

for all $x, x' \in [0,1]^n$ and all $j \in [m]$, $i \in [d_j]$. Moreover, in order to measure the closeness of different $\Phi_i^{(j)}$ in $\bar{\mathscr{X}}$, we let

$$\hat{d} := \max\{d_1, d_2, \ldots, d_m\},$$

and define the minimal gap of Φ as

$$\mathrm{g} := \min\left\{ \left| \Phi_{i'}^{(j)}(x) - \Phi_i^{(j)}(x) \right| \,\middle|\, \Phi_{i'}^{(j)}(x) \neq \Phi_i^{(j)}(x),\ x \in \bar{\mathscr{X}},\ j \in [m],\ i', i \in [d_j] \right\}. \tag{3.5}$$

Proposition 3.2. *Let $\bar{y} = \bar{y}(T)$ be an equilibrium of the ODE system* (1.2)*, and assume that there exists $\hat{y} \in \bar{\mathscr{X}}$, whose distance to $\bar{y}$, denoted by $\varepsilon := \|\hat{y} - \bar{y}\|_\infty > 0$, satisfies*

$$\hat{d}\varepsilon < \frac{1}{2}, \quad \frac{\varepsilon}{\ln(1/(\hat{d}\varepsilon) - 1)} < \frac{T}{2L}, \quad T \ln \frac{1+\hat{d}\varepsilon}{1-\hat{d}\varepsilon} + 2L\varepsilon < \mathrm{g}. \tag{3.6}$$

Then the Eq. (3.3) *holds and thus $\hat{y}$ must be a local optimum of f in $\bar{\mathscr{X}}$.*

Proof. Notice that conditions (3.6) imply

$$\hat{d}\varepsilon < 1, \quad \ln\left(1/(\hat{d}\varepsilon) - 1\right) > 0, \quad T \ln\left(1/(\hat{d}\varepsilon) - 1\right) > 2L\varepsilon.$$

In order to obtain Eq. (3.3), according to the definition of softmax function given in Eq. (1.3), it suffices to show that for every $j \in [m]$ we have

$$\begin{aligned} \Phi_i^{(j)}(\hat{y}) &= \min_{1 \leqslant k \leqslant d_j} \Phi_k^{(j)}(\hat{y}), \quad \text{if} \quad i \in A_j, \\ \Phi_i^{(j)}(\hat{y}) &> \min_{1 \leqslant k \leqslant d_j} \Phi_k^{(j)}(\hat{y}), \quad \text{if} \quad i \notin A_j, \end{aligned} \tag{3.7}$$

where A_j collects the nonzero entries of $\hat{y}^{(j)}$ and $r = |A_j|$.

First, we prove that $\Phi_{i_1}^{(j)}(\hat{y}) = \Phi_{i_2}^{(j)}(\hat{y})$ for all $i_1, i_2 \in A_j$. Using the fact that $\bar{y}$ is an equilibrium, we arrive at

$$\ln\left(\bar{y}_{i_1}^{(j)} / \bar{y}_{i_2}^{(j)}\right) = \frac{1}{T}\left(-\Phi_{i_1}^{(j)}(\bar{y}) + \Phi_{i_2}^{(j)}(\bar{y})\right). \tag{3.8}$$

On the other hand, the distance between $\hat{y}$ and $\bar{y}$ provides a limit for every entry of $\bar{y}^{(j)}$. By $|\hat{y}_i^{(j)} - \bar{y}_i^{(j)}| \leqslant \|\hat{y} - \bar{y}\|_\infty = \varepsilon$, we get

$$\begin{aligned} \bar{y}_i^{(j)} &\in [1/r - \varepsilon, 1/r + \varepsilon], \quad \text{if} \quad i \in A_j, \\ \bar{y}_i^{(j)} &\in [0, \varepsilon], \qquad\qquad\quad \text{if} \quad i \notin A_j. \end{aligned} \tag{3.9}$$

Combining the Eqs. (3.6), (3.8), (3.9) and using $r \leqslant \hat{d}$ gives

$$\left|\Phi_{i_1}^{(j)}(\bar{y}) - \Phi_{i_2}^{(j)}(\bar{y})\right| = T\left|\ln\left(\bar{y}_{i_1}^{(j)} / \bar{y}_{i_2}^{(j)}\right)\right| \leqslant T \ln \frac{1/r + \varepsilon}{1/r - \varepsilon} \leqslant T \ln \frac{1+\hat{d}\varepsilon}{1-\hat{d}\varepsilon} \quad \text{for all} \quad i_1, i_2 \in A_j,$$

so that

$$\begin{aligned}\left|\Phi_{i_1}^{(j)}(\hat{y})-\Phi_{i_2}^{(j)}(\hat{y})\right| &\leqslant \left|\Phi_{i_1}^{(j)}(\hat{y})-\Phi_{i_1}^{(j)}(\bar{y})\right|+\left|\Phi_{i_1}^{(j)}(\bar{y})-\Phi_{i_2}^{(j)}(\bar{y})\right|+\left|\Phi_{i_2}^{(j)}(\hat{y})-\Phi_{i_2}^{(j)}(\bar{y})\right| \\ &\leqslant L\varepsilon + T\ln\frac{1+\hat{d}\varepsilon}{1-\hat{d}\varepsilon}+L\varepsilon < \mathrm{g},\end{aligned}$$

which directly implies $\Phi_{i_1}^{(j)}(\hat{y}) = \Phi_{i_2}^{(j)}(\hat{y})$ for all $i_1, i_2 \in A_j$, because g defined in Eq. (3.5) gives the minimal distance between two different $\Phi_i^{(j)}$s in $\bar{\mathscr{X}}$.

In order to verify Eq. (3.7), the rest is to prove that $\Phi_{i'}^{(j)}(\hat{y}) > \Phi_{i_1}^{(j)}(\hat{y})$ for all $i' \notin A_j$. According to Eqs. (3.6), (3.8) and (3.9) and the fact that $r \leqslant \hat{d}$, we have

$$\Phi_{i'}^{(j)}(\bar{y})-\Phi_{i_1}^{(j)}(\bar{y}) = T\ln\left(\bar{y}_{i_1}^{(j)}/\bar{y}_{i'}^{(j)}\right) \geqslant T\ln\frac{1/r-\varepsilon}{\varepsilon} \geqslant T\ln\left(\frac{1}{\hat{d}\varepsilon}-1\right) > 2L\varepsilon,$$

and then

$$\begin{aligned}\Phi_{i'}^{(j)}(\hat{y})-\Phi_{i_1}^{(j)}(\hat{y}) &= \left(\Phi_{i'}^{(j)}(\bar{y})-\Phi_{i_1}^{(j)}(\bar{y})\right)+\left(\Phi_{i'}^{(j)}(\hat{y})-\Phi_{i'}^{(j)}(\bar{y})\right)-\left(\Phi_{i_1}^{(j)}(\hat{y})-\Phi_{i_1}^{(j)}(\bar{y})\right) \\ &> 2L\varepsilon-\left|\Phi_{i'}^{(j)}(\hat{y})-\Phi_{i'}^{(j)}(\bar{y})\right|-\left|\Phi_{i_1}^{(j)}(\hat{y})-\Phi_{i_1}^{(j)}(\bar{y})\right| \\ &\geqslant 2L\varepsilon-L\varepsilon-L\varepsilon = 0.\end{aligned}$$

The proof is complete. □

Remark 3.1. There is a more concise condition for ε, viz.

$$\varepsilon < \min\left\{\frac{1}{4\hat{d}}, \frac{T}{2L}, \frac{\mathrm{g}}{3\hat{d}T+2L}\right\},$$

from which the three conditions given in Eq. (3.6) can be readily derived by direct relaxations. That is, for a given T, a sufficiently small ε guarantees an equilibrium point of the ODE system (1.2) gives a local optimum of MCPP (1.1) in $\bar{\mathscr{X}}$. However, we cannot theoretically assure that ε will be small enough as T approaches 0 at the current stage.

4. Implementation

The biggest benefit of transforming MCPP (1.1) into the ODE system (1.2) is that various numerical ODE solvers and related techniques come into play for complex combinatorial problems. According to Proposition 3.2, we need to find the equilibrium of Eq. (1.2) quickly rather than to know how we approach it. That is, we do not have to know the dynamics evolution quite precisely. To this end, we adopt variable time step forward Euler scheme for accelerating the calculation. Besides, also inspired by SA, we use an infinite temperatures sequence (mostly it is decreasing) $T_1, T_2, \ldots$, and numerically solve Eq. (1.2) with $T := T_s$, $s = 1, 2, \ldots$, until we find an equilibrium $\bar{y}(T_s)$. Except for the first numerical integration with $T = T_1$, the initial value for the ODE system (1.2) with $T = T_s$ can be set to be the

previous equilibrium $\bar{y}(T_{s-1})$. When the distance $\varepsilon := \min_{y\in\mathscr{X}} \|y-\bar{y}(T)\|_\infty$ is less than a prescribed accuracy tolerance ε_0, we stop evolving the ODE system and turn to rounding procedure.

It should be noted that there is a suitable range for T_1 due to the following reasons:

(1) T_1 should not be too large or the diversity of solutions may be insufficient. To see this, consider the extreme case by formally letting $T=+\infty$ and replacing $1/T$ by 0 in Eq. (3.1). We get $\bar{y}(+\infty)^{(j)} = \mathbf{1}_{d_j}/d_j$, where $\mathbf{1}_{d_j}$ is a d_j-dimensional vector with every entry identical to 1. This uninformative equilibrium is independent of the initial point y_0. A similar phenomenon happens when T_1 is large enough, in which the numerical experiments show that we can only find one equilibrium $\bar{y}(T_1)$ for different y_0s (and it is close to $\bar{y}(+\infty)$). Then the subsequent evolutions of y will be similar. Sometimes the method even produces the same solutions.

(2) T_1 should not be too small either. Otherwise the ODE will act like a greedy local search. This phenomenon is accessible noticing that the SA method is more like greedy algorithm when the temperature becomes smaller and so does the ODE approach.

In practice, it is not necessary to determine the range precisely. To find a reasonable assign for T_1, we first choose a small value then double it repeatedly until $2T_1$ leads to uninformative equilibriums while T_1 does not. Thus the above two requirements are satisfied.

4.1. Choice of the initial data

As mentioned in Section 1, the initial data of ODE y_0 should satisfy the so-called continuous multiple choice constraint

$$\sum_{i=1}^{d_j} (y_0)_i^{(j)} = 1 \quad \text{for all} \quad j\in[m],$$

i.e. $(y_0)^{(j)}$ belongs to the standard (d_j-1)-simplex

$$\Delta^{d_j-1} := \left\{ z\in\mathbb{R}^{d_j} \;\middle|\; z\geqslant 0, \sum_{i=1}^{d_j} z_i = 1 \right\}.$$

Therefore, we generate $(y_0)^{(j)}$s by sampling from a distribution on this simplex and then combine $(y_0)^{(j)}, j=1,2,\ldots,m$ into one y_0. In practice, we choose a symmetric Dirichlet distribution with the concentration parameter 0.01. It is more likely to produce values, most entries of which are close to 0. This may lead to a larger expected distance between each pair of y_0s and thus a more thorough exploration of the solution space of the ODE system.

4.2. Variable time step

The forward Euler (FE) scheme for an autonomous system $\mathrm{d}y/\mathrm{d}t = F(y)$ reads

$$y^k = y^{k-1} + h^{k-1}F^{k-1}, \tag{4.1}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is a vector function, y^k denotes the numerical approximation of $y(t)$ at time t^k, $h^{k-1} := t^k - t^{k-1}$ is the time step, and $F^k := F(y^k)$. To estimate the local truncation error of Eq. (4.1), which is dominated by $(h^{k-1})^2$ when h^{k-1} is small enough, we adopt a method similar to the Richardson extrapolation [24]. Let $(h^{k-1})^2 c$ denote the error. Assuming that the vector c keeps almost unchanged for some successive time steps and $h^{k-1} = h^{k-2}$ for even k, we have another approximation to $y(t^k)$, viz.

$$\tilde{y}^k := y^{k-2} + (h^{k-1} + h^{k-2})F^{k-2} = y^{k-2} + 2h^{k-1}F^{k-2},$$

the error of which is $4(h^{k-1})^2 c$, while the method (4.1) has the error

$$(h^{k-1})^2 c + (h^{k-2})^2 c = 2(h^{k-1})^2 c,$$

if staring from the same y^{k-2}. Therefore, in order to estimate the truncation error in practice, we can use $(\tilde{y}^k - y^k)/2 \approx (h^{k-1})^2 c$. Specifically, we adopt a tolerance $\Theta > 0$ for $\theta^k := \|\tilde{y}^k - y^k\|_2$ and a step adjust ratio $\rho > 1$. Since the error for the k-th step is proportion to the square of step size h^{k-1} and can be well approximated by $\theta^k/2$, we are able to maintain it within a specific range according to Θ and ρ by a two-way adjustment of h^k as follows. Reduce the time step $h^{k+1} = h^k = h^{k-1}/\rho$ when $\theta^k > \Theta\rho^2$, which may avoid the error increase by decreasing the time step; increase the time step $h^{k+1} = h^k = \rho h^{k-1}$ when $\theta^k < \Theta/\rho^2$, which may exploit the maximum large time step allowed by a small error. If the errors make a small change in successive two steps, then θ^k will not be far from $[\Theta/\rho^2, \rho^2\Theta]$. Thus, in doing so, the errors can be controlled around a desired accuracy of $\Theta/2$, and few of them will be outside the interval $[\Theta/2\rho^2, \rho^2\Theta/2]$.

4.3. Rounding procedure

Proposition 3.2 requires $\varepsilon = \|\hat{y} - \bar{y}(T)\|_\infty$ to be small enough. It is natural to think of letting $\hat{y}$ reach the minimum among $\bar{\mathscr{X}}$. However, a naive implementation of this procedure is impractical since each B_{d_j} contains $2^{d_j} - 1$ elements and thus $|\bar{\mathscr{X}}| = \prod_{j=1}^m (2^{d_j} - 1)$. Instead, we only generate an appropriate $\hat{y}$ as follows. For each $\bar{y}^{(j)}$, let $\eta_j = \max_{i\in[d_j]}\{\bar{y}_i^{(j)}\}$ and $r_j = \lfloor 1/\eta_j + 1/2 \rfloor$. Then consider the largest r_j entries of $\bar{y}^{(j)}$ and record the indices in set $A_j \subseteq [d_j]$. This always can be done after noting $\eta_j \geqslant 1/d_j$. Let $\hat{y}^{(j)} = \chi^{A_j}/r_j$ for each $j \in [m]$ then $\hat{y}^{(j)} \in \bar{B}_{d_j}$ as $|A_j| = r_j \leqslant d_j$, and we achieve $\hat{y} \in \bar{\mathscr{X}}$. It should be pointed out that if there exists a r'_j-element set $A'_j \subseteq [d_j]$ such that $\|\bar{y}^{(j)} - \chi^{A'_j}/r'_j\|_\infty$ is small enough, $1/\eta_j$ will be close to r'_j, thus we choose the right $r_j = r'_j$. Moreover, the largest r_j entries of $\bar{y}^{(j)}$ will be close to $1/r'_j$, which form A'_j and are also chosen by A_j. This explains why $\hat{y}^{(j)}$ may serve as a good approximation for $\bar{y}^{(j)}$.

At the final step, we have to get a Boolean vector x from $\hat{y}$. First, we let $x \leftarrow \hat{y}$. Then sequentially change $x^{(j)}$ into a standard unit vector. This process should be done greedily. More precisely, we choose and $i \in [d_j]$ such that minimize $\Phi_i^{(j)}(x^{(\bar{j})})$ and let $x^{(j)} \leftarrow e_i^{d_j}$. This will not increase $f(x)$. When x is locally optimal, the procedure ends. Thus $f(x)$ is no larger than $f(\hat{y})$.

4.4. Cost analysis

The main cost of each FE step in the method (4.1) is calculating $F(y)$, and according to Eq. (1.2), the cost of $F(y)$ consists of calculating $\Phi(y)$, m softmax functions with $\Phi(y)^{(j)}$ as input and $\mathcal{O}(n)$ basic operations. The softmax functions requires another $\mathcal{O}(n)$ operations. Therefore, the computational complexity of the gradient function may dominate the cost and it is necessary to analyze it respectively for different problems in practice.

5. Application to MAX-k-CUT

As a classical graph optimization problem, the MAX-k-CUT problem wants a k-division of the vertex set V for a given edge-weighted graph $G = (V, E, W)$ such that the sum of weights across any two subsets is maximized. Let $V = \{v_i\}_{i=1}^{|V|}$, $W = (w_{ij})_{|V|\times|V|}$ and V_1, $V_2, \ldots, V_k$ denote the subsets after division. Precisely, MAX-k-CUT maximizes the following cut function:

$$\mathrm{cut}(V_1, V_2, \ldots, V_k) := \sum_{v_i \in V_r, v_j \in V_s, r \neq s, i<j} w_{ij}.$$

It reduces to the famous MAX-CUT problem for $k = 2$, one of the 21 Karp's NP-complete problems [16]. MAX-k-CUT is widely studied and there are a number of algorithms for finding its approximating solutions, including the heuristic methods, see e.g. [20] and reference therein, and continuous algorithms [4, 9, 26].

We will use the proposed ODE approach to solve the MAX-k-CUT problem and need to transform it into MCPP at the first step. This can be readily implemented by setting $n \leftarrow k|V|$, $m \leftarrow |V|$, $d_j \leftarrow k$ for any $j \in [m]$ in Eqs. (1.1) and (2.1), and the feasible region for MAX-k-CUT turns out to be $B_k^{|V|}$. Accordingly, the belonging of each vertex v_i is mapped to an element in B_k as follows: $v_i \in V_r$ if and only if $x^{(i)} = e_r^k$; v_i and v_j belong to the same subset if and only if $(x^{(i)})^\top x^{(j)} = 1$. That is, the corresponding MCPP objective function becomes

$$f(x) = -\mathrm{cut}(V_1, V_2, \ldots, V_k) = -\sum_{i<j} w_{ij} \left(1 - (x^{(i)})^\top x^{(j)}\right). \tag{5.1}$$

The minus sign before the cut function converts it into a minimization problem to fit the form of (1.1). For the sake of comparison, we still regard $-f$ as the result cut value in the rest of the paper. Then our ODE approach can be used directly to solve the MAX-k-CUT problem. We implement it with MATLAB R2021a and run it on AMD Ryzen 1950X with 64GB RAM.

Due to the differences in programming languages, implementation and platforms, making a fair comparison of runtime is quite difficult. Therefore, we focus on the comparison of solution quality where the cut values produced by MOH [20] are adopted as the reference. Tables 1-5 present the numerical results for $k = 2, 3, 4, 5$ on G-Set which includes 71 randomly generated graph with 800 to 20000 vertices. So there are 4×71 instances in all. We run 100 independent trials for each instance and record the best cut value among all the results in Tables 1-4, see the columns headed by "ODE". The initial value of each

trial is generated randomly as described in Section 4.1. The trials are simply parallelized with MATLAB's `parfor`. Other parameters are $T_1 = 3$, $T_s = \gamma^{s-1} T_1, s = 2, 3, \ldots$, $\gamma = 0.95$, $\varepsilon_0 = 1 \times 10^{-3}$, $\Theta = 1 \times 10^{-6} n$ and $\rho = 1.1$. T_1 is determined according to the routine described in Section 4, and $\{T_s\}$ is a geometric progression since it is a simple sequence converging to zero. For comparison, Tables 1-4 also calculate the ratio between the best cut value achieved by our ODE approach in those 100 runs and that by MOH for each instance, and one can find that, for all four problems, such ratios are larger than 0.99 for at least 38 instances. For the remaining graphs of G-Set, the ODE method performs diversely.

For $k = 2$, the ratios are at least 0.96 and in only one instance is less than 0.97 (see G64 in Table 1). For $k = 3$, the ratios are all above 0.97. However, there are 5 instances for $k = 4$ the ratios of which are in $(0.95, 0.97]$ (see G18, G19, G20, G56, G61 in Table 3), and there are 6 ratios in this interval for $k = 5$ (see G19, G20, G21, G40, G56, G61 in Table 4). The worst solution quality happens in solving the MAX-5-CUT problem and the ratio decreases to 0.9472 for G18 and it is the only one less than 0.95 in Table 4. For $k = 2, 4, 5$, the ratios reach 0.98 in at least 56 instances, but only 46 ratios reach 0.98 for $k = 3$.

In order to further show the overall performance in the solution quality achieved by the ODE approach, for all $(4 \times 71) \times 100$ trials, we calculate the ratios between the cut values of the ODE approach and the reference values obtained from MOH, and plot the histogram of all these ratios in Fig. 1. It can be observed that in more than 97% trials, the ratios are greater than 0.95, thereby implying that the approximate cuts produced by the ODE approach are of comparable quality to MOH. Moreover, we compare our ODE approach's solutions with those obtained by using Gurobi 10.0.1 to directly solve MCPP in the same time duration (see the column Gurobi in Tables 1-4). We find that, for $k = 2$, in more than 70% instances, ODE's solutions are better than or as good as Gurobi's, and the proportion gets larger and reaches 80% for $k = 3, 4, 5$.

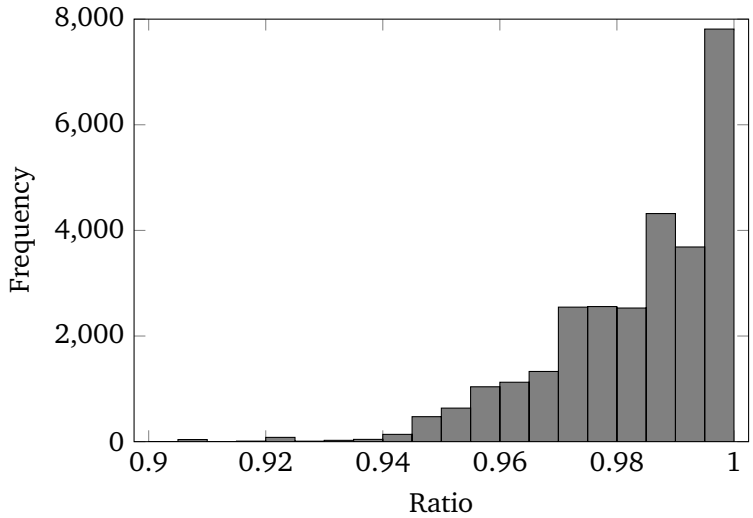

Figure 1: Quality check: Histogram of the ratio between the cut values of the ODE approach and the reference values produced by MOH for all $4 \times 71 \times 100$ trials of MAX-k-CUT.

Table 1: Numerical cut values for MAX-CUT.

Graph	ODE	MOH	Ratio	Gurobi	Graph	ODE	MOH	Ratio	Gurobi
G1	11623	11624	0.9999	11624	G37	7629	7691	0.9919	7535
G2	11612	11620	0.9993	11620	G38	7624	7688	0.9917	7543
G3	11621	11622	0.9999	11622	G39	2355	2408	0.9780	2250
G4	11646	11646	1.0000	11646	G40	2342	2400	0.9758	2198
G5	11626	11631	0.9996	11631	G41	2333	2405	0.9701	2166
G6	2173	2178	0.9977	2178	G42	2429	2481	0.9790	2307
G7	1999	2006	0.9965	2006	G43	6645	6660	0.9977	6225
G8	2005	2005	1.0000	2005	G44	6640	6650	0.9985	6173
G9	2045	2054	0.9956	2038	G45	6638	6654	0.9976	6147
G10	1993	2000	0.9965	1992	G46	6640	6649	0.9986	6282
G11	554	564	0.9823	564	G47	6644	6657	0.9980	6200
G12	548	556	0.9856	556	G48	6000	6000	1.0000	6000
G13	576	582	0.9897	582	G49	6000	6000	1.0000	6000
G14	3049	3064	0.9951	3015	G50	5880	5880	1.0000	5880
G15	3032	3050	0.9941	2993	G51	3823	3848	0.9935	3773
G16	3027	3052	0.9918	2989	G52	3825	3851	0.9932	3769
G17	3030	3047	0.9944	2994	G53	3829	3850	0.9945	3781
G18	979	992	0.9869	918	G54	3820	3852	0.9917	3782
G19	891	906	0.9834	853	G55	10211	10299	0.9915	9906
G20	931	941	0.9894	869	G56	3954	4016	0.9846	3851
G21	914	931	0.9817	880	G57	3414	3494	0.9771	3494
G22	13325	13359	0.9975	11180	G58	19096	19288	0.9900	18928
G23	13337	13344	0.9995	12368	G59	5911	6087	0.9711	5716
G24	13325	13337	0.9991	11008	G60	14089	14190	0.9929	13989
G25	13321	13340	0.9986	12572	G61	5678	5798	0.9793	5590
G26	13308	13328	0.9985	11132	G62	4766	4868	0.9790	4872
G27	3325	3341	0.9952	2515	G63	26799	27033	0.9913	26545
G28	3286	3298	0.9964	2174	G64	8460	8747	0.9672	8239
G29	3379	3405	0.9924	2497	G65	5444	5560	0.9791	5562
G30	3389	3413	0.9930	2442	G66	6208	6360	0.9761	6364
G31	3299	3310	0.9967	2546	G67	6802	6942	0.9798	6950
G32	1384	1410	0.9816	1410	G70	9482	9544	0.9935	9546
G33	1362	1382	0.9855	1382	G72	6840	6998	0.9774	7008
G34	1362	1384	0.9841	1384	G77	9712	9928	0.9782	9940
G35	7628	7686	0.9925	7544	G81	13724	14036	0.9778	14058
G36	7612	7680	0.9911	7538					

Table 2: Numerical cut values for MAX-3-CUT.

Graph	ODE	MOH	Ratio	Gurobi	Graph	ODE	MOH	Ratio	Gurobi
G1	15158	15165	0.9995	14863	G37	9968	10052	0.9916	9893
G2	15160	15172	0.9992	14912	G38	9959	10040	0.9919	9892
G3	15167	15173	0.9996	14899	G39	2837	2903	0.9773	2369
G4	15176	15184	0.9995	14889	G40	2808	2870	0.9784	2357
G5	15187	15193	0.9996	14827	G41	2803	2887	0.9709	2343
G6	2631	2632	0.9996	2323	G42	2897	2980	0.9721	2445
G7	2401	2409	0.9967	2078	G43	8571	8573	0.9998	8303
G8	2420	2428	0.9967	2104	G44	8543	8571	0.9967	8286
G9	2464	2478	0.9944	2138	G45	8543	8566	0.9973	8273
G10	2401	2407	0.9975	2047	G46	8545	8568	0.9973	8279
G11	653	669	0.9761	669	G47	8548	8572	0.9972	8260
G12	645	660	0.9773	661	G48	6000	6000	1.0000	6000
G13	670	686	0.9767	684	G49	6000	6000	1.0000	6000
G14	3979	4012	0.9918	3939	G50	6000	6000	1.0000	6000
G15	3943	3984	0.9897	3930	G51	4992	5037	0.9911	4958
G16	3951	3991	0.9900	3924	G52	4996	5040	0.9913	4952
G17	3936	3983	0.9882	3924	G53	5004	5039	0.9931	4958
G18	1176	1207	0.9743	1027	G54	4993	5036	0.9915	4952
G19	1050	1081	0.9713	872	G55	12329	12429	0.9920	12047
G20	1097	1122	0.9777	931	G56	4610	4752	0.9701	3850
G21	1089	1109	0.9820	929	G57	3985	4083	0.9760	4101
G22	17080	17167	0.9949	16577	G58	24977	25195	0.9913	24016
G23	17130	17168	0.9978	16587	G59	7097	7262	0.9773	5921
G24	17117	17162	0.9974	16604	G60	16946	17076	0.9924	16726
G25	17125	17163	0.9978	16631	G61	6668	6853	0.9730	5641
G26	17103	17154	0.9970	16600	G62	5566	5685	0.9791	5717
G27	3963	4020	0.9858	3338	G63	35001	35322	0.9909	33661
G28	3921	3973	0.9869	3325	G64	10217	10443	0.9784	8587
G29	4036	4106	0.9830	3453	G65	6341	6490	0.9770	6539
G30	4063	4119	0.9864	3489	G66	7241	7416	0.9764	7474
G31	3963	4003	0.9900	3270	G67	7904	8086	0.9775	8148
G32	1618	1653	0.9788	1653	G70	9999	9999	1.0000	9999
G33	1583	1625	0.9742	1627	G72	8010	8192	0.9778	8256
G34	1573	1607	0.9788	1609	G77	11332	11578	0.9788	11666
G35	9961	10046	0.9915	9876	G81	15985	16321	0.9794	16290
G36	9954	10039	0.9915	9911					

Table 3: Numerical cut values for MAX-4-CUT.

Graph	ODE	MOH	Ratio	Gurobi	Graph	ODE	MOH	Ratio	Gurobi
G1	16789	16803	0.9992	16443	G37	11018	11117	0.9911	10967
G2	16798	16809	0.9993	16395	G38	11015	11108	0.9916	10970
G3	16799	16806	0.9996	16375	G39	2935	3006	0.9764	2449
G4	16800	16814	0.9992	16436	G40	2898	2976	0.9738	2487
G5	16798	16816	0.9989	16413	G41	2906	2983	0.9742	2423
G6	2739	2751	0.9956	2439	G42	3014	3092	0.9748	2563
G7	2497	2515	0.9928	2114	G43	9353	9376	0.9975	9089
G8	2513	2525	0.9952	2159	G44	9358	9379	0.9978	9086
G9	2574	2585	0.9957	2120	G45	9360	9376	0.9983	9112
G10	2497	2510	0.9948	2114	G46	9355	9378	0.9975	9100
G11	661	677	0.9764	677	G47	9372	9381	0.9990	9073
G12	652	664	0.9819	665	G48	6000	6000	1.0000	6000
G13	680	690	0.9855	690	G49	6000	6000	1.0000	6000
G14	4396	4440	0.9901	4363	G50	6000	6000	1.0000	6000
G15	4354	4406	0.9882	4346	G51	5518	5571	0.9905	5500
G16	4375	4415	0.9909	4353	G52	5536	5584	0.9914	5494
G17	4360	4411	0.9884	4325	G53	5528	5574	0.9917	5520
G18	1207	1261	0.9572	1031	G54	5527	5579	0.9907	5503
G19	1080	1121	0.9634	894	G55	12498	12498	1.0000	12498
G20	1119	1168	0.9580	934	G56	4722	4931	0.9576	4000
G21	1133	1155	0.9810	923	G57	4036	4112	0.9815	4148
G22	18739	18776	0.9980	18170	G58	27650	27885	0.9916	26567
G23	18736	18777	0.9978	18155	G59	7362	7539	0.9765	6090
G24	18740	18769	0.9985	18211	G60	17148	17148	1.0000	17148
G25	18742	18775	0.9982	18135	G61	6842	7110	0.9623	5850
G26	18734	18767	0.9982	18101	G62	5642	5743	0.9824	5793
G27	4159	4201	0.9900	3433	G63	38758	39083	0.9917	37386
G28	4098	4150	0.9875	3439	G64	10605	10814	0.9807	8884
G29	4245	4293	0.9888	3474	G65	6430	6534	0.9841	6608
G30	4261	4305	0.9898	3548	G66	7352	7474	0.9837	7553
G31	4131	4171	0.9904	3376	G67	8026	8155	0.9842	8256
G32	1641	1669	0.9832	1679	G70	9999	9999	1.0000	9999
G33	1602	1638	0.9780	1644	G72	8123	8264	0.9829	8358
G34	1595	1616	0.9870	1623	G77	11493	11674	0.9845	11827
G35	11017	11111	0.9915	10957	G81	16208	16470	0.9841	16670
G36	11023	11108	0.9923	10955					

Table 4: Numerical cut values for MAX-5-CUT.

Graph	ODE	MOH	Ratio	Gurobi	Graph	ODE	MOH	Ratio	Gurobi
G1	17695	17703	0.9995	17326	G37	11562	11603	0.9965	11354
G2	17693	17706	0.9993	17364	G38	11552	11601	0.9958	11493
G3	17691	17701	0.9994	17390	G39	2944	3022	0.9742	2479
G4	17687	17709	0.9988	17341	G40	2884	2986	0.9658	2384
G5	17697	17710	0.9993	17326	G41	2925	2986	0.9796	2475
G6	2763	2781	0.9935	2317	G42	3037	3109	0.9768	2574
G7	2495	2533	0.9850	2183	G43	9747	9770	0.9976	9487
G8	2516	2535	0.9925	2121	G44	9742	9772	0.9969	9509
G9	2574	2601	0.9896	2226	G45	9747	9771	0.9975	9471
G10	2503	2526	0.9909	2147	G46	9742	9774	0.9967	9481
G11	667	677	0.9852	677	G47	9745	9775	0.9969	9471
G12	654	662	0.9879	665	G48	6000	6000	1.0000	6000
G13	680	689	0.9869	690	G49	6000	6000	1.0000	6000
G14	4611	4639	0.9940	4568	G50	6000	6000	1.0000	6000
G15	4580	4606	0.9944	4548	G51	5802	5826	0.9959	5753
G16	4588	4613	0.9946	4546	G52	5802	5837	0.9940	5761
G17	4575	4603	0.9939	4514	G53	5807	5829	0.9962	5707
G18	1201	1268	0.9472	1013	G54	5806	5830	0.9959	5723
G19	1083	1132	0.9567	918	G55	12498	12498	1.0000	12498
G20	1132	1172	0.9659	937	G56	4733	4971	0.9521	3866
G21	1115	1162	0.9596	963	G57	4049	4111	0.9849	4148
G22	19513	19553	0.9980	18989	G58	29019	29105	0.9970	28017
G23	19513	19558	0.9977	18883	G59	7443	7566	0.9837	6191
G24	19507	19555	0.9975	19063	G60	17148	17148	1.0000	17148
G25	19506	19554	0.9975	18989	G61	6866	7188	0.9552	5946
G26	19504	19552	0.9975	18949	G62	5650	5744	0.9836	5793
G27	4178	4236	0.9863	3449	G63	40692	40786	0.9977	39540
G28	4117	4182	0.9845	3429	G64	10732	10896	0.9849	8809
G29	4255	4327	0.9834	3548	G65	6428	6540	0.9829	6608
G30	4268	4340	0.9834	3563	G66	7364	7476	0.9850	7553
G31	4157	4211	0.9872	3461	G67	8035	8165	0.9841	8256
G32	1644	1670	0.9844	1679	G70	9999	9999	1.0000	9999
G33	1612	1638	0.9841	1644	G72	8145	8266	0.9854	8358
G34	1589	1615	0.9839	1623	G77	11516	11687	0.9854	11827
G35	11547	11605	0.9950	11477	G81	16227	16501	0.9834	16670
G36	11543	11601	0.9950	11416					

Table 5: Approximate total steps and time in seconds on G-Set used by running the ODE approach once for MAX-k-CUT from a given initial data.

k	$\lvert V\rvert$	$\lvert E\rvert$	Number of steps	Time	Time per step
2	8×10^2	2×10^4	3.0×10^4	1.3	4.3×10^{-5}
	2×10^3	2×10^4	1.0×10^5	4.0	4.0×10^{-5}
	1×10^4	2×10^4	9.0×10^4	15	1.7×10^{-4}
	2×10^4	4×10^4	1.5×10^5	30	2.0×10^{-4}
3	8×10^2	2×10^4	9.1×10^4	7.8	8.6×10^{-5}
	2×10^3	2×10^4	1.7×10^5	28	1.6×10^{-4}
	1×10^4	2×10^4	1.1×10^5	55	5.0×10^{-4}
	2×10^4	4×10^4	1.2×10^5	84	7.0×10^{-4}
4	8×10^2	2×10^4	1.2×10^5	20	1.5×10^{-4}
	2×10^3	2×10^4	1.4×10^5	28	2.0×10^{-4}
	1×10^4	2×10^4	1.0×10^6	60	6.0×10^{-4}
	2×10^4	4×10^4	1.2×10^5	90	7.5×10^{-4}
5	8×10^2	2×10^4	1.1×10^5	13	1.2×10^{-4}
	2×10^3	2×10^4	1.5×10^5	30	2.0×10^{-4}
	1×10^4	2×10^4	1.2×10^5	70	5.5×10^{-4}
	2×10^4	4×10^4	1.3×10^5	86	6.6×10^{-4}

It should be noted that, given an initial data, the ODE approach is deterministic, and there is no heuristic operations dedicated to MAX-k-CUT since the implementation and the running parameters strictly follow the general guidelines detailed in Section 4. Hence, we may conclude that, the proposed ODE approach can produce relatively good results without any ad hoc designs.

Next, we are going to show the efficiency of our ODE approach. Let $w_{\text{tot}} = \sum_{i<j} w_{ij}$ represent the total weight of all edges and $P = (p_{ij}) \in \{0,1\}^{k\times|V|}$ a matrix satisfying $p_{ij} = x_i^{(j)}$. It is easy to check that

$$f(x) = -w_{\text{tot}} + \frac{1}{2}\text{Trace}(PWP^\top),$$

yielding $\partial f/\partial P = PW$. Here we use the denominator layout, thereby implying that the complexity of calculating ∇f is $\mathcal{O}(k|E|)$. To obtain this order we have used that W is a sparse matrix with $2|E|$ nonzero elements and calculating each row of PW requires $\mathcal{O}(|E|)$ operations. By the observation in Section 4.4, we deduce that the time complexity of each FE step is $\mathcal{O}(k(|E|+|V|))$.

Table 5 reports the approximate total steps and time needed by running the ODE approach once and shows clearly that the computing cost is not expensive compared to MOH. In MOH, each iteration step has a time complexity of $\mathcal{O}(|E|+k|V|)$ and the total iteration steps may be up to $10^6 \sim 10^8$ [20].

6. Application to the Star Discrepancy

For a positive integer d, vectors $a, b \in \mathbb{R}^d$ and relation $\triangle \in \{<, >, \leqslant, \geqslant\}$, let $a \triangle b$ mean that the relation holds for every entry — i.e. $a_i \triangle b_i$ for all $i \in [d]$. For $a < b$, we define the closed interval $[a, b]$ by

$$[a, b] := \left\{u \in \mathbb{R}^d \mid u \geqslant a \text{ and } u \leqslant b\right\}.$$

The half-open and open cases can be similarly expressed. The volume for an interval denotes as $\mathrm{vol}([a, b]) := \prod(b_i - a_i)$. If $a = 0$, we write it simply as $\mathrm{vol}(b)$. To represent a series of vector with indices, we let those indices appear as superscripts of a single symbol. The subscripts are left for denoting the entries of these vectors. Given an N-point set

$$U = \left\{u^1, \ldots, u^N\right\} \subset [0,1)^d, \quad u = (u_1, \ldots, u_d) \in [0,1]^d,$$

we let

$$A(u; U) = |[0, u) \cap U|, \quad D(u; U) = \mathrm{vol}(u) - \frac{1}{N} A(u; U). \tag{6.1}$$

The star discrepancy of the N-point set U is defined by

$$d^*_\infty(U) = \sup_{u \in [0,1]^d} |D(u; U)|, \tag{6.2}$$

which measures the uniformity of U, and has been widely used in high dimensional numerical integration [12,18] as well as in high dimensional statistics such as number-theoretical methods [6] and density estimation [19, 25]. However, calculating $d^*_\infty(U)$ admits NP-hardness [7] and only a few algorithms are developed, including the exact algorithm [3], a threshold-accepting algorithm [29] and its variation, dubbed the TA_improved algorithm [8]. In this section, we will apply the proposed ODE approach for MCPP to approximate $d^*_\infty(U)$. To the best of our knowledge, this is the first attempt to use a continuous algorithm for approximating the star discrepancy.

For $j \in [d]$, we define

$$\Gamma_j(U) = \{u^i_j \mid i = 1, 2, \ldots, N\}, \quad \bar{\Gamma}_j(U) = \Gamma_j(U) \cup \{1\}, \quad \underline{\Gamma}_j(U) = \Gamma_j(U) \cup \{0\},$$

and set

$$\Gamma(U) := \Gamma_1(U) \times \cdots \times \Gamma_d(U).$$

The terms $\bar{\Gamma}(U)$ and $\underline{\Gamma}(U)$ are defined analogously. It can be readily observed that on each open sub-domain of $[0,1]^d$ divided by the grids in $\Gamma(U)$, $A(u; U)$ keeps unchanged and thus $|D(u; U)|$ reaches its maximum at one of the extreme points of $\mathrm{vol}(u)$, i.e. either the lower left or upper right corner. Since $\underline{\Gamma}(U)$ and $\bar{\Gamma}(U)$ respectively collect all the lower left and upper right corners in $[0,1]^d$, an equivalent form for the star discrepancy in Eq. (6.2) was obtained in [21]

$$d^*_\infty(U) = \max\left\{\max_{u \in \bar{\Gamma}(U)} D(u; U), \max_{u \in \underline{\Gamma}(U)} \bar{D}(u; U)\right\}, \tag{6.3}$$

where

$$\bar{A}(u;U)=|[0,u]\cap U|,\quad \bar{D}(u;U)=\frac{1}{N}\bar{A}(u;U)-\mathrm{vol}(u).$$

We should mention that, $\underline{\Gamma}(U)$, the feasible region for maximizing $\bar{D}(u;U)$, can be slightly narrowed to $\Gamma(U)$ due to the following reasons. For each $j\in[d]$, if $0\in\Gamma_j(U)$, then $\underline{\Gamma}_j(U)$ equals $\Gamma_j(U)$. Otherwise, $\bar{A}(u;U)=0$ if $u_j=0$, thus $\bar{D}(u;U)=0<d^*_\infty(U)$, indicating that we can drop out that kind of u when maximizing $\bar{D}(u;U)$. In either case, we can modify the Cartesian product expression of $\Gamma(U)$ by using $\Gamma_j(U)$ instead of $\underline{\Gamma}_j(U)$. Therefore, from Eq. (6.3), we achieve a more concise form for calculating the star discrepancy,

$$d^*_\infty(U)=\max\Big\{\max_{u\in\bar{\Gamma}(U)}D(u;U),\max_{u\in\Gamma(U)}\bar{D}(u;U)\Big\}. \tag{6.4}$$

The equivalent form (6.4) transforms the star discrepancy into two optimization problems on discrete sets with which we are able to obtain MCPP forms.

Let us start from the first optimization problem $\max_{u\in\bar{\Gamma}(U)}D(u;U)$ in Eq. (6.4). Inspired by the Cartesian-product structure of $\bar{\Gamma}(U)$, the feasible region of corresponding MCPP instance should be B^d_{N+1} since the cardinality of $\bar{\Gamma}_j$ is no larger than $N+1$. That is, the key parameters which shape the MCPP problem in Eqs. (1.1) and (2.1) are: $n\leftarrow(N+1)d$, $m\leftarrow d$, $d_j\leftarrow N+1$ for any $j\in[m]$. However, mapping $\bar{\Gamma}(U)$ into B^d_{N+1} and defining an objective function over B^d_{N+1} that represents $D(u;U)$ need meticulous designs we are about to state below.

For $j\in[d]$, sort $\bar{\Gamma}_j(U)$ into $\bar{u}_{1j}\leqslant\bar{u}_{2j}\leqslant\cdots\leqslant\bar{u}_{Nj}<\bar{u}_{(N+1)j}=1$, which also sorts $\Gamma_j(U)$ in the same order for $\Gamma_j(U)=\{\bar{u}_{1j},\ldots,\bar{u}_{Nj}\}$. For $i\in[N+1]$, let $\bar{u}_{\sigma_{ij}j}:=u^i_j$ record the order of u^i_j and $\{\sigma_{ij}\,|\,i\in[N+1]\}$ be the corresponding permutation of $[N+1]$. When some elements of $\Gamma_j(U)$ are identical, such permutation may not be unique, and it is viable to choose one of them arbitrarily. Let $x\in B^d_{N+1}$. For each $j\in[d]$, there is a unique entry of $x^{(j)}$, denoted by $x^{(j)}_{s_j}$ with $s_j\in[N+1]$, that equals 1. Then we have a surjection from $x\in B^d_{N+1}$ to $u\in\bar{\Gamma}(U)$

$$x^{(j)}\to s_j\to u_j=\bar{u}_{s_jj} \tag{6.5}$$

for all $j\in[d]$. Let

$$v(x):=\prod_{j=1}^{d}\sum_{i=1}^{N+1}x^{(j)}_i\bar{u}_{ij},\quad \alpha(x):=\sum_{i=1}^{N}\prod_{j=1}^{d}\sum_{k=\sigma_{ij}+1}^{N+1}x^{(j)}_k,\quad \delta(x):=v(x)-\frac{1}{N}\alpha(x), \tag{6.6}$$

all of which are affine with respect to each $x^{(j)}$. In fact, $v(x),\alpha(x)$ and $\delta(x)$ act as $\mathrm{vol}(u)$, $A(u;U)$ and $D(u;U)$, respectively. For $x\in B^d_{N+1}$, it should be noticed that the inner sum can be replaced by the logical disjunction "$\vee$" and the product can be replaced by the logical conjunction "$\wedge$" (regarding 0-1 variables as boolean variables) , namely,

$$\alpha(x)=\sum_{i=1}^{N}\bigwedge_{j=1}^{d}\bigvee_{k=\sigma_{ij}+1}^{N+1}x^{(j)}_k,$$

since $\sum_{k=\sigma_{ij}+1}^{N+1}x^{(j)}_k\in\{0,1\}$.

Proposition 6.1. *Let $u \in \bar{\Gamma}(U)$ be the image of $x \in B^d_{N+1}$ under the surjection* (6.5)*. Then we have*

$$\mathrm{vol}(u) = \nu(x), \tag{6.7}$$

$$A(u;U) \leqslant \alpha(x), \tag{6.8}$$

$$D(u;U) \geqslant \delta(x). \tag{6.9}$$

Moreover, $D(u;U)$ and $\delta(x)$ admit a deeper connection — viz.

$$\max_{u\in\bar{\Gamma}(U)} D(u;U) = \max_{x\in B^d_{N+1}} \delta(x). \tag{6.10}$$

Proof. The Eq. (6.7) immediately follows from the relation $\sum_{i=1}^{N+1} x_i^{(j)} \bar{u}_{ij} = u_j$. Now we express $|[0,u) \cap \{u^i\}|$ by x. For each $i \in [N]$, we have

$$\begin{aligned} u^i \in [0,u) \ &\Leftrightarrow\ u^i_j = \bar{u}_{\sigma_{ij} j} < u_j = \bar{u}_{s_j j} \quad \text{for all} \quad j \in [d], \\ &\Rightarrow\ \sigma_{ij} < s_j \qquad\qquad\quad \text{for all} \quad j \in [d], \\ &\Leftrightarrow \bigvee_{k=\sigma_{ij}+1}^{N+1} x_k^{(j)} = 1 \qquad \text{for all} \quad j \in [d]. \end{aligned} \tag{6.11}$$

Thus

$$|[0,u) \cap \{u^i\}| \leqslant \bigwedge_{j=1}^{d} \bigvee_{k=\sigma_{ij}+1}^{N+1} x_k^{(j)},$$

and summing the above inequalities for $i \in [N]$ leads to Eq. (6.8). Combining Eqs. (6.1) and (6.6)-(6.8), we arrive at the inequality (6.9). It follows that $\max_{u\in\bar{\Gamma}(U)} D(u;U) \geqslant \max_{x\in B^d_{N+1}} \delta(x)$ since the mapping (6.5) from B^d_{N+1} to $\bar{\Gamma}(U)$ is surjective.

To prove Eq. (6.10), we only have to show that

$$\max_{u\in\bar{\Gamma}(U)} D(u;U) \leqslant \max_{x\in B^d_{N+1}} \delta(x).$$

This follows from the fact that for each $u \in \bar{\Gamma}(U)$, there exists a preimage of u, $\hat{x} = \hat{x}(u) \in B^d_{N+1}$, such that $A(u;U) = \alpha(\hat{x}(u))$. The $\hat{x}$ is defined by letting s_j equal the minimal index that $u_j = \bar{u}_{s_j j}$ holds and $\hat{x}^{(j)}_{s_j} = 1, j \in [d]$. Thus $\bar{u}_{(s_j-1)j} < \bar{u}_{s_j j}$ whenever $s_j > 1$, otherwise contradicting to the minimality of s_j. When $\sigma_{ij} < s_j$, we then have

$$\sigma_{ij} \leqslant s_j - 1 \ \Rightarrow\ \bar{u}_{\sigma_{ij} j} \leqslant \bar{u}_{(s_j-1)j} < \bar{u}_{s_j j}.$$

Therefore, the "⇒" in Eq. (6.11) becomes "⇔", which results in the equivalence between $|[0,u) \cap \{u^i\}|$ and $\bigwedge_{j=1}^{d} \bigvee_{k=\sigma_{ij}+1}^{N+1} \hat{x}_k^{(j)}$, rather than an inequality. Hence, we get $A(u;U) = \alpha(\hat{x}(u))$. Let u^* reaches the maximum of $D(u;U)$. Then

$$\max_{u\in\bar{\Gamma}(U)} D(u;U) = D(u^*;U) = \delta(\hat{x}(u^*)) \leqslant \max_{x\in B^d_{N+1}} \delta(x).$$

The proof is complete. □

Eq. (6.10) gives an MCPP form for the first optimization problem in Eq. (6.4) and the same treatment can be also applied to the second one. The key parameters which shape the MCPP problem in Eqs. (1.1) and (2.1) become $n \leftarrow Nd$, $m \leftarrow d$, $d_j \leftarrow N$ for any $j \in [m]$, and we need the following functions defined on B_N^d:

$$\bar{\nu}(x') := \prod_{j=1}^{d} \sum_{i=1}^{N} (x')_i^{(j)} \bar{u}_{ij}, \quad \bar{\alpha}(x') := \sum_{i=1}^{N} \prod_{j=1}^{d} \sum_{k=\sigma_{ij}}^{N} (x')_k^{(j)}, \quad \bar{\delta}(x') := \frac{1}{N}\bar{\alpha}(x') - \bar{\nu}(x'),$$

to present the MCPP form

$$\max_{u \in \Gamma(U)} \bar{D}(u;U) = \max_{x' \in B_N^d} \bar{\delta}(x'). \tag{6.12}$$

The proof of this is similar to that of Eq. (6.10) and is omitted. In a word, according to Eqs. (6.10) and (6.12), the star discrepancy is now determined by two MCPP problems

$$d_\infty^*(U) = \max\Big\{ \max_{x \in B_{N+1}^d} \delta(x), \max_{x' \in B_N^d} \bar{\delta}(x') \Big\}, \tag{6.13}$$

which can be solved approximately by the proposed ODE approach in a straightforward manner.

Compared to the quadratic objective function of MAX-k-CUT problem in Eq. (5.1), both $\delta(x)$ and $\bar{\delta}(x')$ in Eq. (6.13) are of degree of d and it may be more complicated to store and calculate these function values as well as their gradients when $d > 2$. For example, calculating the gradients of δ and $\bar{\delta}$ may involve high computational cost if handling it inappropriately. It forms the main cost in using FE scheme (4.1) to solve Eq. (6.13), thereby requiring careful optimization. More precisely, it is easy to check that for $i \in [N+1]$, $j \in [d]$,

$$\begin{aligned} \frac{\partial \nu(x)}{\partial x_i^{(j)}} &= \bar{u}_{ij} \prod_{j' \in [d]\setminus\{j\}} \sum_{i'=1}^{N+1} x_{i'}^{(j')} \bar{u}_{i'j'}, \\ \frac{\partial \alpha(x)}{\partial x_i^{(j)}} &= \sum_{\substack{i' \in [N] \\ \sigma_{i'j} < i}} \prod_{j' \in [d]\setminus\{j\}} \sum_{k=\sigma_{i'j'}+1}^{N+1} x_k^{(j')}. \end{aligned} \tag{6.14}$$

The directly computation of the sums and products in $\partial\alpha(x)/\partial x_i^{(j)}$ requires at least $\mathcal{O}(N^2 d)$ operations. Since there are $(N+1)d$ entries, the complexity of $\nabla\delta(x)$ will be at least $\mathcal{O}(N^3 d^2)$, which is barely acceptable. However, we would like to point out that the complexity can be limited to $\mathcal{O}(Nd)$ according to the following procedure. First, it should be observed that given U, Eq. (6.14) can be rewritten as

$$\begin{aligned} \frac{\partial \nu(x)}{\partial x_i^{(j)}} &= \bar{u}_{ij} \prod_{j' \in [d]\setminus\{j\}} b_{j'}(x), \\ \frac{\partial \alpha(x)}{\partial x_i^{(j)}} &= \sum_{\substack{i' \in [N] \\ \sigma_{i'j} < i}} \prod_{j' \in [d]\setminus\{j\}} z_{i'j'}(x), \end{aligned}$$

where the vector function $b : [0,1]^n \to \mathbb{R}^d$ and the matrix function $z : [0,1]^n \to \mathbb{R}^{N\times d}$ do not depend on i and j. Thus b and z can be calculated in advance. Noting that $z_{i'j'}(x) = \sum_{k>\sigma_{i'j'}} x_k^{(j')}$ is a partial sum of the entries $x^{(j')}$ in the reverse order, we can compute each column of z in a group within a cost of only $\mathcal{O}(N)$. Therefore, z can be computed at the cost $\mathcal{O}(Nd)$. To get $g_{i'j}(x) := \prod_{j'\in[d]\setminus\{j\}} z_{i'j'}(x)$ fast, we replace the multiplication by $\prod_{j'\in[d]} z_{i'j'}(x)/z_{i'j}(x)$, whose numerator can be reused for fixed i'. Therefore, the complexity of computing g from z is also $\mathcal{O}(Nd)$. To get $\nabla\alpha$ from g, the partial sum trick can also be applied in the summation $\sum_{i':\sigma_{i'j}<i} g_{i'j}(x)$, albeit $\sigma_{\cdot j}$ determining the order of entries in $g_{\cdot j}$. Hence, the total time cost is $\mathcal{O}(Nd)$. In the same spirit, we are able to get $\nabla\nu$ with cost of $\mathcal{O}(Nd)$. By these means, the complexity of $\nabla\delta$ is limited to $\mathcal{O}(Nd)$. This is also true for $\nabla\bar{\delta}$. Since $n = \mathcal{O}(Nd)$ for these two MCPP instances, the cost of a FE step is also $\mathcal{O}(Nd)$ by the observation in Section 4.4.

The GLP sets used in [29] are adopted for test in this work, and include 30 small sets with N ranging from 28 to 487 and d from 4 to 6 as well as 6 large sets with N from around 2000 to 5000 and d from 6 to 11. The parameters for our ODE approach are $T_1 = 1\times10^{-4}$, $T_s = \gamma^{s-1}T_1, s = 2,3,\ldots,$ $\gamma = 0.95$, $\varepsilon_0 = 1\times10^{-3}$, $\Theta = 1\times10^{-6}Nd$ and $\rho = 1.1$. The parameters are similar to those in MAX-k-CUT (see Section 5). Numerical values of the star discrepancy obtained by re-running the ODE approach 100 times are listed in Table 6, where those obtained by TA_improved are directly copied from [8].

We list the ratios between the best discrepancy values achieved by our ODE method and by TA_improved in Table 6. Note that the ODE method gives the same value as TA_improved did, except for 4 out of 30 small instances — cf. $(N,d) = (312,4),(376,4),(487,4),(73,6)$ in Table 6. For these four instances, the ratios are at least 0.98, and for the last 6 large instances, the ratios are at least 0.91.

The histogram of the ratios between the discrepancy values of the ODE approach and the reference values obtained from TA_improved for 36×100 trials is plotted in Fig. 2. In more than 85% trials, the ratios are greater than 0.9. Compared to TA_improved's 100000 iterations per trial used in [8] and the same $\mathcal{O}(Nd)$ complexity per iteration, our method requires much less computational resource overall while producing solutions with similar quality. We also record the approximate typical runtime and steps for different sizes of sets in Table 7. It can be seen there that the time consumed per iteration is roughly proportional to Nd.

Since Gurobi 10.0.1 can only deal with quadratic objective functions and constraints, we need to reformulate Eq. (6.13). For the $\nu(x)$ part of $\delta(x)$, the procedure is straightforward. We create d variables $\mu_j, j = 1,2,\ldots,d$ and constrain them by $\mu_j = \sum_{i=1}^{N+1} x_i^{(j)}\bar{u}_{ij}$. Then the product of μ_j is modeled in a standard way as follows. We create $\tau_j, j = 2,3,\ldots,d$, satisfying $\tau_j = \mu_1\mu_2\cdots\mu_j$, which can be done by quadratic constraints $\tau_2 = \mu_1\mu_2$, $\tau_j = \tau_{j-1}\mu_j$, $j = 3,4,\ldots,d$. For the $\alpha(x)$ part, we need $\mathcal{O}(Nd)$ variables representing the subsum terms. The products in $\alpha(x)$ are modelled by Gurobi's standard general constraints "AND". $\bar{\delta}(x)$ is handled in a similar way.

The values obtained by Gurobi limited to the same runtime of ODE approach are also listed in Table 6. Since in some instances, Gurobi fails to produce any feasible solutions,

Table 6: Numerical values of the star discrepancy on GLP sets. Values in the columns headed by "TA" are directly copied from [8] as reference and obtained there by the TA_improved algorithm. Values in the columns headed by "Gurobi" are either the objective values or the quotients of runtimes (in parentheses). A dash "-" means Gurobi fails to provide a competitive solution compared to the ODE method after running for more than 20 hours.

N	d	ODE	TA	Ratio	Gurobi	N	d	ODE	TA	Ratio	Gurobi
145	4	0.0731	0.0731	1.0000	0.0731	28	6	0.5360	0.5360	1.0000	0.5360
255	4	0.1093	0.1093	1.0000	0.1016	29	6	0.2532	0.2532	1.0000	0.2532
312	4	0.0617	0.0618	0.9974	0.0595	35	6	0.3431	0.3431	1.0000	0.3431
376	4	0.0752	0.0753	0.9979	0.0682	50	6	0.3148	0.3148	1.0000	0.3148
388	4	0.1297	0.1297	1.0000	0.0989	61	6	0.1937	0.1937	1.0000	0.1937
442	4	0.0620	0.0620	1.0000	0.0424	73	6	0.1467	0.1485	0.9876	0.1406
448	4	0.0548	0.0548	1.0000	0.0178	81	6	0.2500	0.2500	1.0000	0.2498
451	4	0.0271	0.0271	1.0000	0.0146	88	6	0.2658	0.2658	1.0000	0.2658
471	4	0.0286	0.0286	1.0000	0.0225	90	6	0.1992	0.1992	1.0000	0.1605
487	4	0.0413	0.0413	0.9995	0.0138	92	6	0.1635	0.1635	1.0000	0.1631
102	5	0.1216	0.1216	1.0000	0.1160	2129	6	0.0246	0.0254	0.9685	(1.1)
122	5	0.0860	0.0860	1.0000	0.0791	3997	7	0.0251	0.0254	0.9882	(8.3)
147	5	0.1456	0.1456	1.0000	0.1107	3997	8	0.0242	0.0254	0.9528	(29)
153	5	0.1075	0.1075	1.0000	0.0871	3997	9	0.0387	0.0387	1.0000	-
169	5	0.0755	0.0755	1.0000	0.0710	4661	10	0.0256	0.0272	0.9412	-
170	5	0.0860	0.0860	1.0000	0.0771	4661	11	0.0259	0.0283	0.9152	-
195	5	0.1574	0.1574	1.0000	0.1193						
203	5	0.1675	0.1675	1.0000	0.1260						
235	5	0.0786	0.0786	1.0000	0.0606						
236	5	0.0582	0.0582	1.0000	0.0466						

Table 7: Approximate total steps and time in milliseconds used by the ODE approach for approximating the star discrepancy in Eq. (6.13).

N	d	Time	Number of steps	Time per step
1.0×10^2	4	4.0×10^1	8.0×10^2	5.0×10^{-2}
2.5×10^2	4	6.0×10^1	8.0×10^2	6.0×10^{-2}
5.0×10^2	4	1.0×10^2	1.5×10^3	7.0×10^{-2}
5.0×10^2	6	6.0×10^2	5.0×10^3	1.2×10^{-1}
2.0×10^3	6	3.2×10^3	8.0×10^3	4.0×10^{-1}
4.0×10^3	10	1.3×10^4	1.3×10^4	1.0×10^0

we let it run until it finds a solution no worse than our approach's, then we record the runtime (as there are two maximum problems in one instance, we choose the one with larger objective value found by ODE approach). The columns headed by "Gurobi" in Table 6 show the results by two means. That is, the numbers without parentheses repre-

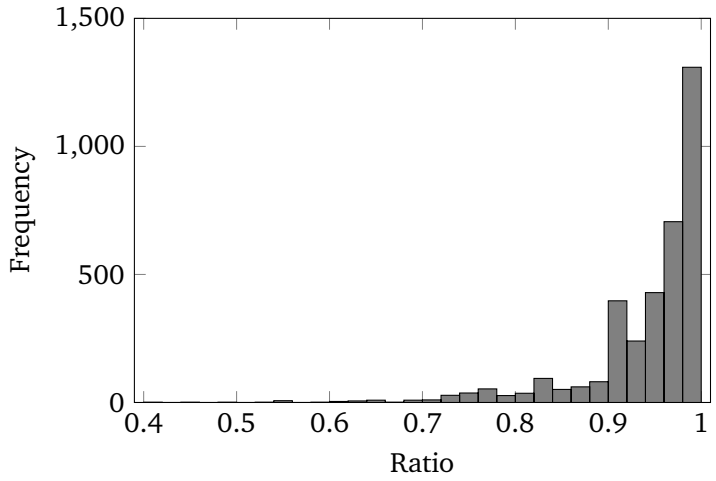

Figure 2: Quality check: Histogram of the ratios between the discrepancy values of the ODE approach and the reference values produced by TA_improved for all 36×100 trials.

sent the objective function values found by Gurobi under the limitation of runtime, while the ones in parentheses represent the quotients of runtimes of Gurobi and our method (total time for two problems). In the first 30 instances, Gurobi manages to find feasible solutions, but the objective values are all less than or equal to our method's. For $(N,d) = (3997,9),(4661,10),(4661,11)$, Gurobi fails to provide a competitive solution compared to ODE method after running for more than 20 hours (therefore, we leave a dash there).

7. Conclusion and Discussion

We proposed an ODE approach for multiple choice polynomial programming (MCPP) and demonstrated its validity via both theoretical analysis and numerical experiments. It fully exploits a connection between the discrete MCPP problem and the continuous ODE system through revealing the relation between local optima of the MCPP and equilibriums of the ODE. The resulting solutions of MCPP instances representing two specific problems are relatively good compared to dedicated algorithms', and are mostly competitive compared to Gurobi's. We are conducting analysis on the existence of equilibrium points and the realizability of conditions that ensures the local optimality, and trying more advanced numerical techniques for evolving an ODE to its equilibrium points. We are going to extend the proposed ODE approach to some kinds of mixed integer programming problems with more constraints rather than multiple choice. On the other hand, although the polytope of unconstrained pseudo-boolean optimization has been thoroughly studied [5], there is very limited research on the polytope of MCPP, and thus accelerating the ODE approach with the aid of polyhedral property and/or cutting-plane method is also a subject of future research. Moreover, we hope that our preliminary attempt in this work may inspire more new connections between discrete data world and continuous math field.

Acknowledgements

This research was supported by the National Key R&D Program of China (Grant Nos. 2020AAA0105200, 2022YFA1005102) and by the National Natural Science Foundation of China (Grant Nos. 12325112, 12288101, 11822102). S.S. is partially supported by the Beijing Academy of Artificial Intelligence.

References

[1] K. Bestuzheva et al., *The SCIP optimization suite 8.0*, Tech. Rep. 21-41, ZIB, Takustr. 7, 14195 Berlin (2021).
[2] E. Boros and P. L. Hammer, *Pseudo-boolean optimization*, Discrete Appl. Math. **123**, 155–225 (2002).
[3] P. Bundschuh and Y. Zhu, *A method for exact calculation of the discrepancy of low-dimensional finite point sets I*, Abh. Math. Semin. Univ. Hambg. **63**, 115–133 (1993).
[4] S. Burer, R.D.C. Monteiro and Y. Zhang, *Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs*, SIAM J. Optim. **12**, 503–521 (2002).
[5] A. Del Pia and A. Khajavirad, *A polyhedral study of binary polynomial programs*, Math. Oper. Res. **42**, 389–410 (2017).
[6] K.-T. Fang and Y. Wang, *Number-Theoretic Methods in Statistics*, Chapman and Hall (1993).
[7] M. Gnewuch, A. Srivastav and C. Doerr, *Finding optimal volume subintervals with k points and calculating the star discrepancy are NP-hard problems*, J. Complex. **25**, 115–127 (2009).
[8] M. Gnewuch, M. Wahlström and C. Winzen, *A new randomized algorithm to approximate the star discrepancy based on threshold accepting*, SIAM J. Numer. Anal. **50**, 781–807 (2011).
[9] M.X. Goemans and D.P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM **42**, 1115–1145 (1995).
[10] Gurobi Optimization, LLC., *Gurobi optimizer reference manual*, https://www.gurobi.com.
[11] W.C. Healy, Jr., *Multiple choice programming (a procedure for linear programming with zero-one variables)*, Oper. Res. **12**, 122–138 (1964).
[12] E. Hlawka, *Funktionen von beschränkter variatiou in der theorie der gleichverteilung*, Ann. di Mat. Pura ed Appl. **54**, 325–333 (1961).
[13] J.J. Hopfield, *Neurons with graded response have collective computational properties like those of two-state neurons*, Proc. Natl. Acad. Sci. USA **81**, 3088–3092 (1984).
[14] IBM ILOG CPLEX, *User's manual for CPLEX*, https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer.
[15] A. Jagota, *Approximating maximum clique with a Hopfield network*, IEEE Trans. Neural Netw. **6**, 724–735 (1995).
[16] R.M. Karp, *Reducibility among combinatorial problems*, in: *Complexity of Computer Computations*, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger (Eds), pp. 85–103, The IBM Research Symposia Series, Springer US (1972).
[17] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, *Optimization by simulated annealing*, Science **220**, 671–680 (1983).
[18] J.F. Koksma, *A general theorem from the theory of uniform distribution modulo 1*, Mathematica B (Zutphen) **11**, 7–11 (1942).
[19] D. Li, K. Yang and W.H. Wong, *Density estimation via discrepancy based adaptive sequential partition*, in: *Advances in Neural Information Processing Systems, 29*, Curran Associates, Inc. (2016).

[20] F. Ma and J.-K. Hao, *A multiple search operator heuristic for the max-k-cut problem*, Ann. Oper. Res. **248**, 365–403 (2017).
[21] H. Niederreiter, *Discrepancy and convex programming*, Ann. di Mat. Pura ed Appl. **93**, 89–97 (1972).
[22] Y.-S. Niu and R. Glowinski, *Discrete dynamical system approaches for boolean polynomial optimization*, J. Sci. Comput. **92**, 46 (2022).
[23] C. Peterson and J.R. Anderson, *Neural networks and NP-complete optimization problems: A performance study on the graph bisection problem*, Complex Systems **2**, 58–59 (1988).
[24] L.F. Richardson and R.T. Glazebrook, *The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam*, Philos. Trans. Royal Soc. A **210**, 307–357 (1911).
[25] S. Shao and Y. Xiong, *SPADE: Sequential-clustering particle annihilation via discrepancy estimation*, arXiv:2005.05129 (2020).
[26] S. Shao, D. Zhang and W. Zhang, *A simple iterative algorithm for maxcut*, J. Comput. Math. **42**, 1277–1304 (2024).
[27] S. Singh and Sonia, *Multi-choice programming: an overview of theories and applications*, Optimization **66**, 1713–1738 (2017).
[28] J. Wang, *An improved discrete Hopfield neural network for max-cut problems*, Neurocomputing **69**, 1665–1669 (2006).
[29] P. Winker and K.-T. Fang, *Application of threshold-accepting to the evaluation of the discrepancy of a set of points*, SIAM J. Numer. Anal. **34**, 2028–2042 (1997).