Dual Hopfield Methods for Large-Scale Mixed-Integer Programming

Bertrand Travacca [1], Scott Moura [1,2]

***Abstract*— We present a novel heuristic first order method for large-scale mixed-integer programs, more specifically we focus on mixed-integer quadratically constrained quadratic programs. Our method builds on Lagrangian relaxation techniques and Hopfield Neural Networks. For illustration, we apply this method to an economic load dispatch problem and compare with two convex approximation techniques.**

I. Introduction

This paper addresses the problem of efficient algorithms to solve large-scale mixed-integer quadratic programs using a novel heuristic method. We begin with the problem formulation.

A. Problem formulation

Consider the following mixed-integer quadratically constrained quadratic program (MIQCQP) given by

$$\begin{array}{lll} \min & f_0(x) & \\ \text{s. to:} & f_j(x) \leq 0 & j \in J \\ & x_i \in \{0,1\} & i \in I_b \\ & x_i \in [0,1] & i \in I_c \end{array} \tag{1}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $J = \{1, \cdots, m\}$ (m is the number of constraints), I_b and I_c are complementary subsets of $I = \{1, \cdots, n\}$, with $|I_b| := n_b > 0$. We consider all the functions to be quadratic

$$f_j(x) = \frac{1}{2} x^T P_j x + q_j^T x + r_j, \quad j \in \{0\} \cup J$$

where $P_j \in \mathbb{R}^{n \times n} \in S^n(\mathbb{R})$ (set of symmetric matrices of size n), $q_j \in \mathbb{R}^n$ and $r_j \in \mathbb{R}$ are given problem parameters. We denote θ the parameters of the problem

$$\theta = \{P_0, ..., P_m\} \cup \{q_0, ..., q_m\} \cup \{r_0, ..., r_m\}$$

Non-convexity arises from: (i) P_j for $j \in \{0\} \cup J$ are possibly indefinite (i.e. the minimum eigenvalue is strictly negative), and (ii) the binary constraints.

*This work was supported by TOTAL SA

[1] Energy, Controls, and Applications Lab (eCAL), Civil & Environmental Engineering, University of California, Berkeley, USA. Corresponding author: bertrand.travacca@berkeley.edu

[2] Tsinghua-Berkeley Shenzhen Institute (TBSI), University of California, Berkeley, USA.

Note that $x_i \in \{0,1\}$ is equivalent to the two indefinite quadratic inequalities

$$\begin{cases} x_i(x_i - 1) & \leq 0 \\ -x_i(x_i - 1) & \leq 0 \end{cases}$$

Similarly, for $i \in I$, $x_i \in [0,1]$ can be rewritten as: $x_i - 1 \leq 0$ and $-x_i \leq 0$. All in all, (1) can be recast as

$$\begin{array}{lll} \min & f_0(x) & \\ \text{s. to:} & f_j(x) \leq 0 & j \in \tilde{J} \end{array} \tag{2}$$

with $|\tilde{J}| = m + 2n$ and appropriate $f_j,\ j > m$.

B. The Tractability Problem

MIQCQPs occur across a multitude of applications, including power systems, transportation, logistics, and more. For instance, discrete time Hybrid Systems can be approximated as a mixed logical dynamic systems by transforming Boolean relations into polyhedral constraints using conjunctive normal form and transforming events into linear mixed integer relations [1]. If the objective to minimize is quadratic convex, the optimal control for the Hybrid System can be formulated as a MIQP (mixed integer quadratic programs) with convex objective. Other applications consist of Boolean least squares, two way partitioning problems (e.g. maximum cut) [2] or 3-satisfiability problem [3]. Unfortunately, MIQCQPs are NP-hard in general: they contain Mixed-Integer Linear Programs (MILPs) as a special case. In fact, finding polynomial times algorithms for this class of problems is highlighted as one of the Millennium Prize Problems: $P \stackrel{?}{=} NP$. Our purpose is much more humble, we only aim at producing heuristic algorithms to find candidate solutions to MIQCQPs.

Global methods refer to methods that exactly solve (1). Global methods are often based on the branch-and-bound framework or branch-and-cut [4]. Nevertheless these methods are characterized by a complexity that grows exponentially with n, and are therefore limited as the problem grows in size. In the present article we focus on methods for large scale problems, and we will therefore not consider global methods in our analysis.

C. Literature review

Examples of metaheuristics to solve combinatorial optimization problems include simulated annealing [5], tabu search [6], genetic algorithms [7], particle swarm optimization [8] and more. Refer to [9] for a comprehensive review of these methods. Another approach consists in approximating (1) with a convex problem. We briefly present three convex relaxation techniques from which our method builds upon, or takes inspiration from. These include: (i) binary relaxation in the convex quadratic case [10], [11], (ii) the so-called Lagrangian relaxation technique [12], [3], and (iii) semidefinite relaxation [13], [14], [3]. We introduce each technique in detail in Section II. Other convex procedures have also been used, e.g. the convex-concave approach [15], [16], [3], and alternating direction method of multipliers (ADMM) [3], [17].

Meanwhile, Hopfield neural networks (HNNs) were introduced in [18], [19], and were initially used in machine learning as a way to memorize the state of data (also known as content addressable memory) [20]. The principal methods for training Hopfield networks on data use variants of Hebbian rule [21]. A different but relevant application of HNNs involves using its structure and dynamics (cf. Section III-A, Eqn. (5)) to find a candidate solution to an optimization problem [22], [23], [24]. In this case the weights of the network are not obtained via training. Instead, they are defined by the parameters θ of the optimization problem [25]. In fact, we show in Section III-A that the corresponding method is closely related to projected gradient descent. HNNs were used for combinatorial programming for the first time in [22], with application to the traveling salesman problem. Since then, HNNs have been used in combinatorial problems such as clustering [26], vertex cover [27], linear and nonlinear programming [28], and knapsack problems [29]. We refer the reader to [30], [31] for a comprehensive review of HNNs applications. This paper advances our understanding of HNNs to solve MIQCQPs efficiently by proposing a novel Dual Hopfield method.

D. Paper Organization

Section II details existing convex approximation methods. Section III details existing Hopfield methods and then proposes a novel method that extends to the broader class of MIQCQP problems using duality. Section IV applies the proposed dual Hopfield method to an economic load dispatch problem and compares it with a semidefinite relaxation method.

II. Existing Convex Approximation Methods

A. Binary relaxation in the convex quadratic setting

If $P_j \succeq 0$, for $j \in J$, then a natural relaxation of (1) is to replace, for $i \in I_b$, the binary constraints $x_i \in \{0, 1\}$, by (convex) the convex constraint $x_i \in [0, 1]$. The relaxed problem is a convex quadratic constrained quadratic program and can be solved efficiently using off the shelf solvers. The optimal value provides a lower bound to (1). If x^r denotes a solution to this relaxed problem, then $x_i^r \in [0, 1]$ can be interpreted as the probability that the i^{th} agent is equal to one [10], alternatively x^r can be projected back to $\{0, 1\}$ as a final step.

B. Lagrangian relaxation

The *Lagrangian* of (2) is given by

$$\mathcal{L}(x, \lambda) = f_0(x) + \sum_{j \in \tilde{J}} \lambda_j f_j(x)$$

The *dual function*, $g(\lambda) = \min_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda)$ can be computed analytically, and the dual problem can then be cast as a semidefinite program (SDP) [3]. Hence, this method uses weak duality (see [2]). In general, the complexity for solving SDPs is typically between $\mathcal{O}(n^5)$ and $\mathcal{O}(n^8)$ [32], hence for $n \geq 10^2$, this method can become relatively slow.

C. Semidefinite relaxation

Another relaxation for MIQCQPs uses a semidefinite relaxation (SDR) technique called *lifting*. A new variable $X = xx^T$ is introduced and (1) is rewritten as

$$\begin{aligned} \min \quad & \frac{1}{2}\text{Tr}(P_0 X) + q_0^T x + r_0 \\ \text{s. to:} \quad & \frac{1}{2}\text{Tr}(P_j X) + q_j^T x + r_j \leq 0, \ j \in J \\ & X_{i,i} = x_i, \quad i \in I_b \\ & x_i \in [0, 1], \quad i \in I_c \\ & X = xx^T \end{aligned} \tag{3}$$

This problem is equivalent to (1). The equality constraint $X = xx^T$ makes this problem non-convex. SDR relaxation replaces this intractable constraint by the convex inequality $X \succeq xx^T$. Using Schur complement

$$X \succeq xx^T \iff \begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0$$

Therefore the relaxed version of (3) is an SDP. Generally speaking, the complexity is similar to that of Lagrangian relaxation described previously. It is remarkable that SDR is the biconjugate of problem (1) and the dual of the Lagrangian relaxed problem, to which there is no duality gap when Slater's conditions hold.

D. Metrics for Approximation Methods

Next we describe example metrics for evaluating the performance of approximation methods, which will be used in the case study in Section IV. Consider a "candidate point", $x \in \mathbb{R}^n$, obtained from a heuristic method that aims at being close to the set of optimal

solutions denoted by $\mathcal{X}^*$ (non empty by hypothesis). Finding a *feasible* point to (1) is in most cases a NP-hard problem itself. Therefore, we often do not require a candidate to be feasible. To assess the distance of x to the feasible set $\mathcal{X}$, we define a constraint violation function $v : x \in \mathbb{R}^n \to v(x) \in \mathbb{R}^q$, $q \geq 1$. For instance [3] propose the *maximum constraint violation* ($q = 1$)

$$v(x) := \max_{j \in \bar{J}} f_j(x)_+$$

where $z_+ := \max(0, z)$. Another choice could be the 2-dimensional *average constraint violation*

$$v(x) := \begin{bmatrix} \frac{1}{m} \sum_{j \in J} f_j(x)_+ \\ \\ \frac{1}{n_b} \sum_{i \in I_b} d_{\{0,1\}}(x_i) \end{bmatrix} \in \mathbb{R}^2$$

where $d_{\{0,1\}}(z) := \begin{cases} |1 - z| \text{ if } z \geq 0.5 \\ |z| \text{ if } z < 0.5 \end{cases}$

Generally speaking, the choice of v should be tailored for a particular application. Often, we can use a real-valued trade-off criterion $C(x) = f_0(x) + \gamma^T v(x)$ where $\gamma \in \mathbb{R}^q$ is a hyperparameter corresponding to the user's preferences.

Definition 1. *Given a specific optimization problem, a choice of v, parameters $\theta \sim \mathcal{D}_\theta$, where $\mathcal{D}_\theta$ denotes the distribution from which the parameters of the problem are drawn, we say that heuristic method 1 is on average better than heuristic method 2 if*

$$\mathbb{E}_\theta \begin{bmatrix} f_0(x_1) \\ v(x_1) \end{bmatrix} \geq \mathbb{E}_\theta \begin{bmatrix} f_0(x_2) \\ v(x_2) \end{bmatrix}$$

where $\geq$ denotes the elementwise comparison operator. Alternatively, if a trade-off criterion is defined, if

$$\mathbb{E}_\theta[C(x_1)] \geq \mathbb{E}_\theta[C(x_2)]$$

Remark 1. *Definition 1 does not define an ordered set without a trade-off criterion.*

Given a candidate solution from a heuristic method, local methods can be used to improve it. Local methods can, for instance, be rule based. This framework is referred to as *suggest and improve* in [3]. In this paper we focus exclusively on the suggest step, i.e. finding a candidate point.

III. Dual Hopfield method

A. Hopfield methods

Consider the following unconstrained MIQP

$$\begin{aligned} \min_{0 \leq x \leq 1} \quad & f_0(x) \\ \text{s. to:} \quad & x_i \in \{0, 1\}, \ i \in I_b \end{aligned} \tag{4}$$

The HNN corresponding to (4) involves an autonomous nonlinear dynamic system [22], [31] given by

$$\begin{aligned} \dot{x}_H(t) &= -\nabla f_0(x(t)) \\ x(t) &= \sigma(x_H(t)) \\ x(0) &\in (0, 1)^n \end{aligned} \tag{5}$$

where x and $x_H \in \mathbb{R}^n$ are respectively referred to as the state and hidden state of the system. The function $\sigma : \mathbb{R}^n \to [0, 1]$ is defined element-wise as

$$\sigma : x \in \mathbb{R}^n \mapsto (\sigma_1(x_1), ..., \sigma_n(x_n)) \in \mathbb{R}^n$$

where all functions σ_i, $i \in I$ are surjective, monotonically increasing, continuous everywhere and differentiable almost everywhere. In machine learning, these class of functions are referred to as *activation functions* (e.g. piecewise linear, logistic, $tanh$), a terminology that we will adopt. In the remainder of the paper, only piecewise linear activation will be considered, let $i \in I$

$$\sigma(x_i) = \begin{cases} 0, & x_i < \frac{1}{2} - \frac{1}{2\beta_i} \\ \beta_i(x_i - \frac{1}{2}) + \frac{1}{2}, & \frac{1}{2} - \frac{1}{2\beta_i} \leq x_i \leq \frac{1}{2} + \frac{1}{2\beta_i} \\ 1, & x_i > \frac{1}{2} + \frac{1}{2\beta_i} \end{cases}$$

where β_i is a hyperparameter idiosyncratic to the i-th constraint. If $\beta_i = 1$, then the activation function is the convex projection on the segment $[0, 1]$. If $\beta \to \infty$, σ_i asymptotically approximates the (non-convex) projection on $\{0, 1\}$, we denote it $\sigma_{i,\infty}$. Figure 1 displays σ_i for different choices of β_i.

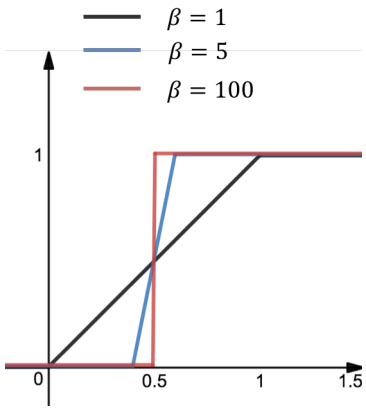

Fig. 1. Piecewise linear activation for $\beta = 1, 5, 100$.

We now state and prove a central theorem to the HNN heuristic,

Theorem 1 (Decreasing objective function)**.** *The objective function f_0 decreases along the unique state trajectory $x(t)$ of the HNN (5).*

Proof: Picard-Lindelöf theorem ([33], Ch.3) applies to (5), and therefore this dynamical system is well-posed.

That is, there exist a unique (time) trajectory for $x(t)$ with initial condition x_0. Consequently, we have

$$\frac{d}{dt}f_0(x(t)) = -[\sigma'(x_H(t))\odot\nabla f_0(x(t))]^T\nabla f_0(x(t)) \leq 0$$

where $\odot$ denotes the elementwise product.

As a corollary, because the state is restricted in the compact $[0,1]^n$, there exist $f_0^\dagger > -\infty$ such that $f_0(x(t)) \to f_0^\dagger$. We also denote $x^\dagger \in [0,1]^n$ a point such that $f_0(x^\dagger) = f_0^\dagger$ and

$$x^\dagger \in \{x | \forall i \in I,\ x_i \in \{0,1\} \text{ or } \nabla f_0(x)_i = 0\}$$

We refer to the explicit time discretization of HNN as *Hopfield method*, method presented in algorithm 1.

Algorithm 1 Hopfield method [discrete time]

Initialize $x_0 \in (0,1)^n$
for $k = 0, 1, ...$
  Step 1, $x_H^{k+1} = x_H^k - \alpha^k \nabla f_0(x^k)$
  Step 2, $x^{k+1} = \sigma(x_H^{k+1})$
until stopping criterion is met

Remark 2. *In algorithm 1. and subsequent ones, we do not discuss an explicit stopping criterion or choice of step size α^k. A stopping criterion could involve a given number of iterations, or a condition of the type $\|f_0(x^k) - f_0(x^{k+1})\| \leq \varepsilon$, $\varepsilon > 0$ given. The step size choice can be a constant or follow other rules, e.g. as in [34].*

We show that this algorithm is closely related to projected gradient descent algorithm used in the same setting as Section II-A, where f_0 is a convex and $x_i \in \{0,1\}$ is relaxed to $x_i \in [0,1]$, for $i \in I_b$.

Algorithm 2 Projected gradient descent

Initialize $x_0 \in (0,1)^n$
for $k = 0, 1, ...$
  Step 1, $x_H^{k+1} = x^k - \alpha^k \nabla f_0(x^k)$
  Step 2, $x^{k+1} = \sigma_{\beta=1}(x_H^{k+1})$
until stopping criterion is met

The main difference between the Hopfield method (Alg 1.) and projected gradient descent (Alg 2.) is Step 1. Specifically, the hidden state x_H is used as a temporary variable in Alg 2. and does not have dynamics. The proof of convergence for projected gradient descent relies on the 1-Lipschitz property of convex projections. In Algorithm 2 a choice $\beta_i > 1$ could therefore compromise convergence. If we seek to approximate the projection on $\{0,1\}$, then we can use the Hopfield method.

B. Limitations of traditional uses of Hopfield methods

Traditionally Hopfield methods have been used for programs with a quadratic objective, linear equality constraints and binary constraints [22], [23], [24]

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{s. to:} \quad & Ax = b \\ & x_i \in \{0,1\}, \quad i \in I \end{aligned} \tag{6}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. In the literature, the linear equality constraints are relaxed and included as a penalty term. This is referred to as *penalty approach*

$$\begin{aligned} \min_x \quad & f_0(x) + \frac{\rho}{2}\|Ax - b\|_2^2 \\ \text{s. to:} \quad & x_i \in \{0,1\}, \quad i \in I \end{aligned} \tag{7}$$

where $\rho > 0$ is a hyperparameter tuned by the practitioner. A variant of what we call Hopfield method (cf. Alg 1 in Section III-A) is then applied to (7).

The main drawback of this heuristic is that the penalty approach does not ensure that a finite ρ exists such that problems (6) and (7) yield equivalent solutions, even in the strictly convex case with binary relaxation, i.e. $P_0 \succ 0$ and $x \in [0,1]^n$. For example, in the penalty approach presented in [35] (Section 3.1.1), the penalty is taken asymptotically to infinity. Increasing ρ in an unbounded fashion creates numerical instability and generally slows down convergence. To tune ρ, trial and error approaches such as the Sequential Unconstrained Maximization Technique (SUMT) [36] have been proposed. A variety of penalty approaches have been used for Hopfield methods. For example, in [37], one hyperparameter per scalar linear equality $\rho_j > 0$, for $j \in J$ needs to be tuned. In [38] the penalty is scaled by a matrix, via a technique called the "subspace approach" and only one hyperparameter is needed. Nevertheless, these methods never entirely solve the aforementioned issues.

C. Lagrangian relaxation

Our method uses Lagrangian relaxation, similar to Section II-B, but without including the integer and binary constraints in the Lagrangian

$$\mathcal{L}(x,\lambda) = f_0(x) + \sum_{j\in J}\lambda_j f_j(x)$$

The dual function is then given by

$$\begin{aligned} g(\lambda) = \min_{0\leq x\leq 1} \quad & \mathcal{L}(x,\lambda) \\ \text{s. to:} \quad & x_i \in \{0,1\}, \quad i \in I_b \end{aligned} \tag{8}$$

Given λ, we denote by $x^\star(\lambda) = \arg\min_x \mathcal{L}(x,\lambda)$ an optimal solution to (8) and we denote by $\mathcal{X}$ its feasible set. Slater's condition does not hold, and therefore a zero duality gap is not guaranteed (weak duality). Nevertheless the following bounds hold

$$g(\lambda^\star) = \max_{\lambda \geq 0} g(\lambda) \leq f_0^* \leq f_0(x^\star(\lambda^\star))$$

where f_0^* is the optimal value of the primal problem (1) and $\lambda^\star \in \operatorname{argmax}_{\lambda \geq 0} g(\lambda)$. Because $\mathcal{L}(x, \lambda)$ is linear in λ, the dual function $g(\lambda)$ is concave. Namely, $g(\lambda)$ is the pointwise minimum of linear functions, and is therefore concave in λ. This guarantees the existence of a supergradient set $\partial g(\lambda)$ for any $\lambda \geq 0$. Unfortunately, Danskin's Theorem hypotheses do not hold. Nevertheless, we can have access to one element of the supergradient set of g at λ as follows. Given any $\lambda \geq 0$, and $\tilde{\lambda} \geq 0$

$$\begin{aligned} g(\tilde{\lambda}) &= \min_{x \in X} \mathcal{L}(x, \tilde{\lambda}) \\ &= \min_{x \in X} \mathcal{L}(x, \lambda) + \sum_{j \in J} (\tilde{\lambda} - \lambda) f_j(x) \\ &\geq g(\lambda) + \sum_{j \in J} (\tilde{\lambda} - \lambda) f_j(x^\star(\lambda)) \end{aligned}$$

Hence, $\sum_{j \in J} f_j(x^\star(\lambda)) \in \partial g(\lambda)$ and the dual ascent with step size $\gamma^k > 0$ (Algorithm 3) can be applied to iteratively compute $\lambda^\star$.

Algorithm 3 *Dual gradient ascent*

Initialize $\lambda_0 \geq 0$
for $k = 0, 1, ...$
  Step 1, $x^\star(\lambda^k) \in \operatorname{argmin}_{x \in \mathcal{X}} \mathcal{L}(x, \lambda^k)$
  Step 2, $\lambda^{k+1} = \lambda^k + \gamma^k \sum_{j \in J} f_j(x^\star(\lambda^k))$
until stopping criterion is met

Convergence speed can be improved via acceleration methods introduced by Nesterov [39]. They introduce momentum into the iterations by using information from the gradient at consecutive iterations. An accelerated method for dual gradient ascent would involve the following update at *Step 2* of Algorithm 3

$$\lambda^{k+1} = \lambda^k + \frac{k-1}{k+2}(\lambda^k - \lambda^{k-1}) + \gamma^k \sum_{j \in J} f_j(x^*(\lambda^k))$$

where we highlight in grey the difference with classical dual gradient ascent. In our particular setting there is no proof of convergence for Nesterov accelerated ascent since g is generally not smooth and not differentiable as in [40] or *simple* as in [41]. In our Dual Hopfield heuristic, we will approximate $\sum_{j \in J} f_j(x^\star(\lambda^k))$ by $\sum_{j \in J} f_j(x^\dagger(\lambda^k))$ – the objective value to which Hopfield method corresponding to the optimization problem (8) converges to.

D. Dual ascent via Hopfield method

The dual ascent via Hopfield (and its accelerated counterpart in grey) is detailed in algorithm 4:

Accessing the value of the dual function $g(\lambda)$ with $\lambda \geq 0$ corresponds to an optimization problem similar to (4). Hopfield dual ascent involves approximating the dual function and its minimizer $x^*(\lambda^*)$ with $x^\dagger(\lambda^\dagger)$ – one of the key idea of this paper.

Algorithm 4 *(Accelerated) Dual Hopfield Method*

Initialize $\lambda_0 \geq 0$
for $k = 0, 1, ...$
  Step 1, Hopfield method
  Initialize $x_0 \in (0, 1)^n$
  for $k = 0, ...$
    $x_H^{k+1} = x_H^k - \alpha^k \nabla_x \mathcal{L}(x^k, \lambda^k)$
    $x^{k+1} = \sigma(x_H^{k+1})$
  until stopping criterion is met
  $x^\dagger(\lambda^k) = x^k$
  Step 2, dual ascent
  $\lambda^{k+1} = \lambda^k + \frac{k-1}{k+2}(\lambda^k - \lambda^{k-1}) + \gamma^k \sum_{j \in J} f_j(x^\dagger(\lambda^k))$
until stopping criterion is met
$\lambda^\dagger = \lambda^k$
$x^\dagger(\lambda^\dagger) = x^k$

The structure of this algorithm is similar to classical dual ascent. Hence, depending on the structure of matrices P_j, $j \in J$ (e.g. separability), in algorithm 4, the computation at *Step 1* can be distributed among *agents*, $i \in I$. Due to its similarity to dual ascent, it is also possible to produce versions of our method that relates directly to the method of multipliers (MM) by introducing the augmented Lagrangian $\mathcal{L}_\rho$

$$\mathcal{L}_\rho(x, \lambda) := \mathcal{L}(x, \lambda) + \frac{\rho}{2} \|Ax - b\|_2^2$$

where $Ax = b$ collects all the equality constraints of problem (1) (i.e. $\{j \in J : P_j = 0\}$). The MM version of Algorithm 4 simply consists in replacing $\mathcal{L}$ by $\mathcal{L}_\rho$. Finally, using the same augmented Lagrangian, an ADMM version of our algorithm can be built. These features could potentially be added to provide convergence robustness as explained in [17].

IV. Case Study: Economic Load Dispatch

Hopfield methods using the penalty approach (cf. Section III-B) have been used to solve economic load dispatch problem in [42], [43], [44]. In our case study we include start-up and shut-down costs, which results in a non-convex quadratic equality constraint to test the proposed dual Hopfield method.

A. Problem formulation

Consider a closed electricity system with n generators. At an initial time $t = 0$, a given generator i is either 'on' ($x_i^0 = 1$) or 'off' ($x_i^0 = 0$). The vector $x^0 \in \{0, 1\}^n$ collects all these initial on/off states. Similarly $y^0 \in \mathbb{R}_+^n$ is a vector gathering all the generator power output magnitudes at $t = 0$. For the next time step ($t = 1$), the dispatcher simultaneously decides (i) which generators

to turn 'on' or 'off' and (ii) a power level for each generator that will be in the 'on' state. For generator i, we denote the 'on' decision by $x_{i,\circ} \in \{0,1\}$ with $x_{i,\circ} = 1$ corresponding to a start-up request. Similarly, for generator i, the 'off' decision is denoted $x_{i,|} \in \{0,1\}^n$. Vector $y \in \mathbb{R}^n$ is the power level decision vector. The on/off state $x \in \{0,1\}^n$ at $t = 1$ is given by,

$$x = x_\circ - x_| + x^0 \tag{9}$$

Moreover, we consider that it is not possible for the economic dispatcher to turn on a generator that was already on or to turn off a generator that was already off. This translates to,

$$\begin{aligned} x_\circ^T x^0 &= 0 \\ x_|^T (1 - x^0) &= 0 \end{aligned} \tag{10}$$

These two conditions imply that a generator cannot be simultaneously turned on and off (i.e. $x_|^T x_\circ = 0$).

We additionally assume that each generator i in the 'on' state has power output limits. The sum off all the power outputs (i.e. electricity supply) must be equal to demand $d \in \mathbb{R}_+$ at $t = 1$. Moreover, only generators that are on can output power. These constraints are given by

$$\underline{y}_i \leq y_i \leq \overline{y}_i \tag{11}$$

$$x^T y = d \tag{12}$$

Each generator has a marginal cost of production $c_i \in \mathbb{R}_+$ as well as a switching costs $c_{\circ,i}$ (start-up cost) and $c_{|,i} \in \mathbb{R}_+$ (shut-down cost). The objective function for the economic dispatcher is assumed linear

$$f_0(y, x_\circ, x_|) := c^T y + c_\circ^T x_\circ + c_|^T x_|$$

In summary, the economic dispatch problem is given by the nonlinear mixed integer problem:

$$\begin{aligned} \min_{y \in \mathbb{R}^n_+, x_\circ, x_| \in \{0,1\}^n} \quad & f_0(y, x_\circ, x_|) \\ \text{s. to:} \quad & (9), (10), (11), (12) \end{aligned} \tag{13}$$

Without the constraint (12) this problem would be a MILP. A straightforward approximation of this problem consists in solving the following linear problem (LP)

$$\begin{aligned} \min_{y \in \mathbb{R}^n} \quad & c^T y \\ \text{s. to:} \quad & y^T 1 = d, \qquad \underline{y} \leq y \leq \overline{y} \end{aligned} \tag{14}$$

and then computing $x_\circ$, $x_|$ as a post-process step (LPp method). Naturally, with LPp the 'on' and 'off' switching process is not optimized. Nevertheless, by construction, LPp method remarkably achieves to find a feasible point to the original problem (13). Hence, it is always possible to find a feasible solution in polynomial time to problem (13) (provided the LP (14) is feasible).

An alternative approximation method is SDR. A semidefinite relaxation of (13) consists in solving the convex problem (15) derived in the Appendix.

B. Results

The problem parameters θ are generated from independent uniform distributions. We use MATLAB to code the proposed dual Hopfield method and use the CVX toolbox [45] to solve the SDR problem and the linear program (14). Interested readers may refer to [46] for the code and the choice of parameters for the uniform distribution $\mathcal{D}_\theta$. We define the electricity demand constraint violation $v_D(x) := \frac{1}{d}|x^T y - d|$ and the binary constraint violation

$$v_B(x) := \frac{1}{2n}\Big(\|x_\circ - \sigma_\infty(x_\circ)\|_1 + \|x_| - \sigma_\infty(x_|)\|_1\Big)$$

Finally, we define the trade-off criterion

$$C(x) := f_0(x) + \gamma v_B(x)$$

with $\gamma = 1.5 \cdot 10^6$, a choice that we will justify hereafter. We run 100 simulations (i.e. draw randomly 100 times $\theta \sim \mathcal{D}_\theta$). We then compute the empirical means. Results are summarized in the following table,

Criteria	unit	DH	ADH	SDR	LPp
$\overline{f_0}$	10^6 \$	1.331	1.37	1.156	1.614
$\overline{v_D}$	%	0.96	0.98	60	0
$\overline{v_B}$	%	0	0.055	0.001	0
$\overline{C}$	10^6 \$	1.346	1.384	2.056	1.614

Acronyms DH and ADH refer respectively to the dual Hopfield method and its accelerated counterpart. Using paired sample t-tests with 95% confidence, all differences seen in the empirical mean results are *significant*, i.e. null hypothesis can be rejected, with the exception of $\overline{v_D}$ between DH and ADH. Note that the binary constraint violations are small for each method. This is why we do not include it in the trade-off criterion. The value of γ corresponds to the magnitude of the average total cost for electricity. Using this trade-off criterion, the dual Hopfield method is the best performing method (cf. Definition 1). Most notably it performs 50% better than SDR and almost 20% better than LPp. The accelerated dual ascent slightly under-performs (less than 3% off). However, ADH exhibits better convergence properties as shown in Fig. 2.

V. Conclusion

This paper introduces a novel heuristic first order method, the dual Hopfield method, for large-scale MIQC-QPs. We have shown how the Hopfield method is closely related to gradient descent, where the hidden state has its own dynamics. In turn, the Hopfield method

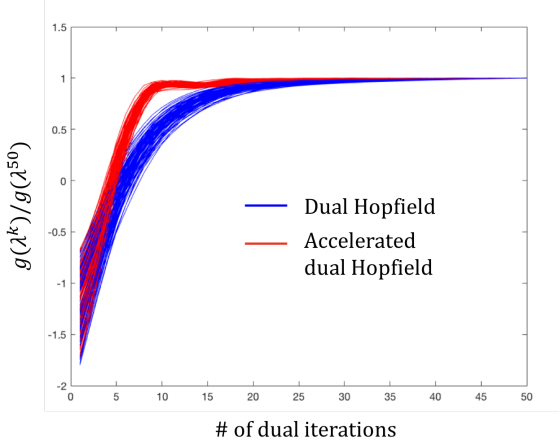

Fig. 2. Juxtaposition of the 100 convergence profiles of the dual Hopfield method (blue) and its accelerated counterpart (red)

allows one to approximate the binary projection while guaranteeing convergence and a decreasing objective function value at each iteration. In order to adapt the Hopfield method to the MIQCQP setting, we used weak duality and approximated the dual function using the Hopfield method. This approach allows one to avoid the drawbacks of the penalty approach previously used in the literature, while also extending the heuristic to a larger class of problems. Contrary to SDR, our method is well suited to large scale problems and is amenable to distributed optimization techniques.

Finally, we applied the dual Hopfield method to an economic dispatch problem with start-up and shut-down costs. We demonstrated that dual Hopfield methods yield higher performing solutions than SDR or LP approximations.

APPENDIX

The SDR of problem (13) reads

$$\begin{aligned}
\min_{Z,x_\circ,x_\shortmid,y,z}\ & c^T y + c_\circ^T x_\circ + c_\shortmid^T x_\shortmid \\
\text{s. to: } & x_\circ, x_\shortmid \in [0,1]^n, \quad \underline{y} \le y \le \overline{y} \\
& x_\circ^T x^0 = 0, \quad x_\shortmid^T (1 - x^0) = 0 \\
& Z \succeq 0, \quad \begin{bmatrix} Z & z \\ z^T & 1 \end{bmatrix} \succeq 0, \quad z = \begin{bmatrix} x_\circ \\ x_\shortmid \\ y \end{bmatrix} \\
& \forall i \in \{1, ..., 2n\}, \ Z_{i,i} = z(i) \\
& \mathrm{Tr}(PZ) + q^T z = d
\end{aligned} \tag{15}$$

where, $P = \frac{1}{2}\begin{bmatrix} 0 & 0 & I \\ 0 & 0 & -I \\ I & -I & 0 \end{bmatrix}$ and $q = \begin{bmatrix} 0 \\ 0 \\ x^0 \end{bmatrix}$

REFERENCES

[1] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[2] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[3] J. Park and S. Boyd, "General heuristics for nonconvex quadratically constrained quadratic programming," *arXiv preprint arXiv:1703.07870*, 2017.

[4] C. Bliek1ú, P. Bonami, and A. Lodi, "Solving mixed-integer quadratic programming problems with ibm-cplex: a progress report," in *Proceedings of the twenty-sixth RAMP symposium*, 2014, pp. 16–17.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[6] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & operations research*, vol. 13, no. 5, pp. 533–549, 1986.

[7] J. Holland, "Adaptation in natural and artificial systems. mi," *Ann Arbor: University of Michigan Press*, 1975.

[8] J. Kennedy and R. Eberhart, "Pso optimization," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 4. IEEE Service Center, Piscataway, NJ, 1995, pp. 1941–1948.

[9] G. A. E.-N. A. Said, A. M. Mahmoud, and E.-S. M. El-Horbaty, "A comparative study of meta-heuristic algorithms for solving quadratic assignment problem," *arXiv preprint arXiv:1407.4863*, 2014.

[10] P. Raghavan and C. D. Tompson, "Randomized rounding: a technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.

[11] L. Lovász, "On the ratio of optimal integral and fractional covers," *Discrete mathematics*, vol. 13, no. 4, pp. 383–390, 1975.

[12] N. Z. Shor, "Quadratic optimization problems," *Soviet Journal of Computer and Systems Sciences*, vol. 25, no. 6, pp. 1–11, 1987.

[13] L. Lovász, "On the shannon capacity of a graph," *IEEE Transactions on Information theory*, vol. 25, no. 1, pp. 1–7, 1979.

[14] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

[15] T. Lipp and S. Boyd, "Variations and extension of the convex–concave procedure," *Optimization and Engineering*, vol. 17, no. 2, pp. 263–287, 2016.

[16] X. Shen, S. Diamond, Y. Gu, and S. Boyd, "Disciplined convex-concave programming," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 1009–1014.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[18] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[19] ——, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[20] H. Wang, B. Raj, and E. P. Xing, "On the origin of deep learning," *arXiv preprint arXiv:1702.07800*, 2017.

[21] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[22] J. J. Hopfield and D. W. Tank, ""neural" computation of decisions in optimization problems," *Biological cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

[23] R. L. Wang, Z. Tang, and Q. P. Cao, "A learning method in hopfield neural network for combinatorial optimization problem," *Neurocomputing*, vol. 48, no. 1-4, pp. 1021–1024, 2002.

[24] A. H. Gee and R. W. Prager, "Polyhedral combinatorics and neural networks," *Neural Computation*, vol. 6, no. 1, pp. 161–180, 1994.

[25] D. Tank and J. Hopfield, "Simple'neural'optimization networks: An a/d converter, signal decision circuit, and a linear programming circuit," *IEEE transactions on circuits and systems*, vol. 33, no. 5, pp. 533–541, 1986.

[26] B. Kamgar-Parsi and B. Kamgar-Parsi, "Clustering taxonomic data with neural networks," in *Proceedings of the International Neural Network Conference—Washington DC*, vol. 1, 1990, pp. 277–80.

[27] J. Ramanujam and P. Sadayappan, "Optimization by neural networks," in *IEEE International Conference on Neural Networks*, vol. 2, 1988, pp. 325–332.

[28] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 5, pp. 554–562, 1988.

[29] M. Ohlsson, C. Peterson, and B. Söderberg, "Neural networks for optimization problems with inequality constraints: the knapsack problem," *neural computation*, vol. 5, no. 2, pp. 331–339, 1993.

[30] C.-K. Looi, "Neural network methods in combinatorial optimization," *Computers & Operations Research*, vol. 19, no. 3-4, pp. 191–208, 1992.

[31] K. A. Smith, "Neural networks for combinatorial optimization: a review of more than a decade of research," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 15–34, 1999.

[32] L. Vandenberghe, V. R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh, "Interior-point algorithms for semidefinite programming problems derived from the kyp lemma," in *Positive polynomials in control*. Springer, 2005, pp. 195–238.

[33] S. Sastry, *Nonlinear systems: analysis, stability, and control*. Springer Science & Business Media, 2013, vol. 10.

[34] Y.-x. Yuan, "Step-sizes for the gradient method," *AMS IP Studies in Advanced Mathematics*, vol. 42, no. 2, p. 785, 2008.

[35] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.

[36] G. P. McCormick, "The projective sumt method for convex programming," *Mathematics of operations research*, vol. 14, no. 2, pp. 203–223, 1989.

[37] X.-Y. Wu, Y.-S. Xia, J. Li, and W.-K. Chen, "A high-performance neural network for solving linear and quadratic programming problems," *IEEE transactions on neural networks*, vol. 7, no. 3, pp. 643–651, 1996.

[38] S. Aiyer and F. Fallside, *A Subspace Approach to Solving Combinatorial Optimization Problems with Hopfield Networks*. University of Cambridge, Department of Engineering, 1990.

[39] Y. Nesterov, "A method of solving a convex programming problem with convergence rate o (1/k2)," in *Soviet Mathematics Doklady*, vol. 27, no. 2, 1983, pp. 372–376.

[40] W. Su, S. Boyd, and E. Candes, "A differential equation for modeling nesterov's accelerated gradient method: Theory and insights," in *Advances in Neural Information Processing Systems*, 2014, pp. 2510–2518.

[41] Y. Nesterov *et al.*, "Gradient methods for minimizing composite objective function," 2007.

[42] T. Yalcinoz and M. Short, "Neural networks approach for solving economic dispatch problem with transmission capacity constraints," *IEEE Transactions on Power Systems*, vol. 13, no. 2, pp. 307–313, 1998.

[43] J. Park, Y. Kim, I. Eom, and K. Lee, "Economic load dispatch for piecewise quadratic cost function using hopfield neural network," *IEEE transactions on power systems*, vol. 8, no. 3, pp. 1030–1038, 1993.

[44] T. D. King, M. El-Hawary, and F. El-Hawary, "Optimal environmental dispatching of electric power systems via an improved hopfield neural network model," *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1559–1565, 1995.

[45] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2008.

[46] B. Travacca, "Hopfield methods and boltzmann machines, code for conference on decision and control 2018," https://github.com/bertravacca/Hopfield-Methods-Boltzmann-Machines, GitHub, 2018.