hmip code

Mathilde Badoual, Bertrand Travacca

1 Introduction

In this document we present in detail the code for hmip, a heuristic solver for nonlinear mixed integer programming. We aim at solving

$$\begin{aligned} &\min f(x) \\ &\quad x \in [lb, ub] \\ &\quad x_i \in \{lb_i, ub_i\}, \quad i \in I_b \\ &\quad A_{ineq} x \leq b_{ineq} \\ &\quad A_{eq} x = b_{eq} \end{aligned} \tag{1}$$

with f differentiable and with $L-$Lipschitz gradient.

2 hopfield.py

2.1 Class 'HopfieldSolver'

2.1.1 __init__

Instantiate the Hopfield method using the user input or default values. We refer to these as options.

- activation_function activation function that is used in hmip. By default sin function is used. The other options are: tanh, pwl, exp, identity. We denote it ϕ. It is a function from $[lb, ub]$ to $\mathbb{R}^n$.
- inverse_activation_function is the inverse of the activation above, we denote it ϕ^{-1}. This function is defined on $[lb, ub]$.
- proxy_distance_vector is the function $\phi'(\phi^{-1})$ defined on $[lb, ub]$.
- ascent_stop_criterion. When the initialization starts with gradient ascent, the ascent stopping criterion is a value that specifies when to stop gradient ascent. The default value is 0.06 We make sure to have the condition ascent_stop_citerion > absorption_criterion
  using the method utils.adapt_ascent_stop_criterion

- stopping_criterion_type is a type of criterion used to stop the Hopfield method. Not implemented yet.

- absorption_criterion is a value that specifies when to absorb a component x_i to $\{lb_i, ub_i\}$ when $x_i - lb_i$ or $ub_i - x_i$ is smaller than this value. By default the absorption_criterion is None.

- initial_ascent_type is the type of gradient scent used to initialize x. There are two possible type: ascent and binary_neutral_ascent. The former consists in using gradient ascent and the former insures that for $i \in I_b$ we always have $x_{i,0} = \frac{ub_i - lb_i}{2}$

- step_type is the type of step size used for the Hopfield method. There are three possibilities either the step size is constant and specified by the user, the step size is classic (using the formula from the article) or armijo which is adapted to the Hopfield method setting. By default the classic step size is chosen

- direction_type is the type of Hopfield descent method that is chosen. There are four options: classic (the direction is $-\nabla f$), stochastic (a random perturbation around $-\nabla f$), binary consists in solving an optimization problem that gives a descent direction as close as possible to the closest (mixed-)binary point, i.e. . Finally soft_binary consists in solving an optimization problem that gives a descent direction as close as possible to

$$2\frac{\phi(x) - lb}{ub - lb} - 1$$

  By default the direction type is classic.

- max_iterations is the maximum number of iterations used for the Hopfield method

- precision_stopping_criterion is the value used for the stopping criterion. Not implemented yet.

- gamma denoted γ is a value between 0 and 1 that is used to produce a new vector $v = \gamma b + (1 - \gamma)h$ between a vector b and the steepest normalized descent $h := -\nabla f(x)/\|\nabla f(x)\|$. The default value is 0.95.

- theta denoted θ is a value between 0 and $\pi/2$ the maximum deviation angle from the steepest descent $-\nabla f()$. I.e. such that a vector d respects the constraint

$$-\nabla f(x)^\top d \geq -\cos(\theta)\|\nabla f(x)\|\|d\|$$

  The default value is 0.05

- beta denoted $\beta \in \mathbb{R}^n$ is a hyperparameter vector used for the activation function. Given $i \in I$, the choice $\beta_i = 1$ corresponds to an activation $\phi - i$ such that $\max_{q \in \mathbb{R}} \phi_i'(q) = 1$. Taking $\beta_i >> 1$ approximates the projection on the set $\{lb_i, ub_i\}$.

2.1.2 setup_optimization_problem

Creates the dictionary problem corresponding to the optimization problem (1). The key-values of this dictionary are:

- objective_function: f
- gradient: $-\nabla f$
- lb: lb
- ub: ub
- A_eq: A_{eq}
- b_eq: b_{eq}
- A_ineq: A_{ineq}
- b_ineq: b_{ineq}
- binary_indicator: I_b
- smoothness_coef: L
- x_0: x_0
- dim_problem: $|I_b|$
- penalty_eq: ρ_{eq}
- penalty_ineq: ρ_{ineq}

2.2 solve

This is the method developed used to solve (find a candidate solution) the optimization problem. It takes as inputs the problem dictionary described in 2.1.2 and attributes described in 2.1.1.

2.3 _get_dual_variables

When the objective function is convex , take as an input the problem and the options and returns the dual variables of the relaxed problem. We can rewrite the binary relaxed version of (1) equivalently as

$$\begin{aligned} \min_{x,s} & f(x) \\ & A_{ineq}x = b_{ineq} + s \\ & A_{eq}x = b_{eq} \\ & lb \leq x \leq ub \\ & s \leq 0 \end{aligned} \tag{2}$$

The quadratic augmented Lagrangian of this problem with penalty parameters $\rho_{\text{ineq}}, \rho_{\text{eq}}$ reads

$$\begin{aligned}\mathcal{L}(x, s, \lambda_{\text{eq}}, \lambda_{\text{ineq}}) = f(x) + \lambda_{\text{eq}}^{\top}(A_{eq}x - b_{eq}) + \lambda_{\text{ineq}}^{\top}(A_{ineq}x - b_{ineq} - s) \\ + \frac{\rho_{\text{ineq}}}{2}\|A_{ineq}x - b_{ineq} - s\|_2^2 + \frac{\rho_{\text{eq}}}{2}\|A_{eq}x - b_{eq}\|_2^2\end{aligned} \tag{3}$$

_get_dual_variables runs the following algorithm

while *Dual stopping criterion not met* **do**
  while *Primal stopping criterion is not met* **do**
    $[x, s] \leftarrow$
    projection$([x, s] -$ gradient_augmented_lagrangian$(x, s, \lambda_{\text{ineq}}, \lambda_{\text{eq}}))$
  end
  $\lambda_{\text{ineq}} \leftarrow \lambda_{\text{ineq}} + \rho_{\text{ineq}}$inequality_constraint$(x, s)$
  $\lambda_{\text{eq}} \leftarrow \lambda_{\text{eq}} + \rho_{\text{eq}}$equality_constraint$(x, s)$
end

The functions projection, inequality_constraint, equality_constraint and gradient_augmented_lagrangian are described hereafter.

$$\text{problem, options} \longrightarrow \boxed{\text{\_get\_dual\_variables}} \longrightarrow [\lambda_{\text{ineq}}, \lambda_{\text{eq}}]$$

2.3.1 projection

Take as an input a vector $[x, s]$ and outputs $[\min(\max(x, 0), 1), \max(s, 0)]$

$$[x, s] \longrightarrow \boxed{\text{projection}} \longrightarrow [\min(\max(x, 0), 1), \max(s, 0)]$$

2.3.2 inequality_constraint

We transform inequality constraints as equality constraints by rewriting

$$A_{ineq}x \leq b_{ineq}$$

as

$$A_{ineq}x - b_{ineq} - s = 0$$

with

$$s \leq 0$$

hence the function inequality_constraint is defined as

$$(z, s) \longrightarrow \boxed{\text{inequality\_constraint}} \longrightarrow A_{ineq}z - b_{ineq} - s$$

2.3.3 equality_constraint

$$(z, s) \longrightarrow \boxed{\text{equality\_constraint}} \longrightarrow A_{eq}z - b_{eq}$$

2.3.4 gradient_augmented_lagrangian

Computes the gradient of the augmented Lagrangian (3) with respect to (x, s)

$$(x, s, \lambda_{\text{ineq}}, \lambda_{\text{eq}}) \longrightarrow \boxed{\text{gradient\_augmented\_lagrangian}} \longrightarrow \begin{bmatrix} \nabla_x \mathcal{L}(x, s, \lambda_{\text{eq}}, \lambda_{\text{ineq}}) \\ \nabla_s \mathcal{L}(x, s, \lambda_{\text{eq}}, \lambda_{\text{ineq}}) \end{bmatrix}$$

2.4 _hopfield_update

Take as an input the current hidden state x_H, the state (or optimization variable) x, the step size α, the direction d for the next iterate, as well as lb, ub (defined in the problem dictionary) and the activation (options). This function outputs the next iterate for x and

$$(x, x_H, \alpha, d, \text{problem, options}) \longrightarrow \boxed{\text{\_hopfield\_update}} \longrightarrow x_H + \alpha d, \phi(x_H)$$

2.5 _alpha_hop

Takes as an input the problem and options as well as the current optimization variable x, the gradient $\nabla f(x)$ and direction d. This function outputs the next step size α.

$$(\text{problem, options}, x, \nabla f(x), d) \longrightarrow \boxed{\text{\_alpha\_hop}} \longrightarrow \alpha$$

If the direction type is not stochastic then

$$\alpha = \frac{-(\sigma \odot \nabla f(x))^\top d}{L\|\beta \odot d\|_2^2 + 12|\nabla f(x)|^\top (d \odot \beta)^2}$$

If the direction type is stochastic then

$$\alpha \leftarrow \alpha(1 - \frac{1}{\sqrt{k}}) + \frac{1}{\sqrt{k}L}$$

where k is the number of iterations of the hopfield_update and L is the smoothness coefficient for f.

2.6 _compute_x_0

This function takes as input the options and problem (possibly x_0 if it has been specified by the user) to output an initial state x_0.

$$(\text{problem, options}) \longrightarrow \boxed{\text{\_compute\_x\_0}} \longrightarrow x_0$$

This function starts by checking if problem input x_0 is of the type None or if is outside $[lb, ub]$. If that either are true then x_0 is initially

$$x_0 = \frac{ub + lb}{2}$$

The function then consists in variations of projected gradient ascent. Let us denote $\varepsilon :=$ ascent_stop_criterion

while *iterations* $<$ *max_iterations* ***and*** $lb + \varepsilon \leq x_0 \leq ub - \varepsilon$ **do**
  if *initial_ascent_type is ascent* **then**
    $x_0 \leftarrow x_0 + \frac{1}{L}\nabla f(x_0)$
  end
  if *initial_ascent_type is binary_neutral_ascent* **then**
    $x_0 \leftarrow x_0 + \frac{1}{L}\nabla f(x_0) \odot (1 - b)$
  end
  $iterations \leftarrow iterations + 1$
end

2.7 _stoping_criterion_met

Take as an input the options and problem, the number of iterations, x, $\nabla f(x)$ and outputs True if the stopping criterion has been met.

$$(problem, options, x, \nabla f(x)) \longrightarrow \boxed{\text{\_stoping\_criterion\_met}} \longrightarrow \{True, False\}$$

It returns True either if $k >$ max_iterations. If the option stopping_criterion_type is gradient then it returns true if the maximum number of iterations has been reached or $\|\nabla f(x) \odot \phi'(\phi^{-1}(x))\|_2 \leq \varepsilon$. Where $\varepsilon =$precision_stopping_criterion.

2.8 _compute_binary_absorption_mask

Takes as an input the problem x and returns a vector in $u \in \{0,1\}^n$. Such that $u_i = 0$ if $x_i \in \{lb_i, ub_i\}$ and $u_i = 1$ otherwise.

$$(\text{problem}, x) \longrightarrow \boxed{\text{\_compute\_binary\_absorption\_mask}} \longrightarrow u \in \{0,1\}^n$$

2.9 _find_direction

Takes as an input the options and problem, $x, \nabla f(x)$; and outputs d the Hopfield direction for the next iterate.

$$(\text{problem, options}, x, \nabla f(x)) \longrightarrow \boxed{\text{\_find\_direction}} \longrightarrow d$$

We denote $u =$ _compute_binary_absorption_mask(problem, x)

- if direction_type is classic or stochastic, then if absorption_criterion is not None then $d = -\nabla f(x)$. Otherwise, $d = -\nabla f(x) \odot u$. If direction_type is stochastic than we add noise to it. $d \leftarrow d \odot (1 + \varepsilon)$ where ε is a centered random variable.

- if direction_type is binary or soft_binary, we start by creating a vector v such that

$$v = (\phi(x) + \frac{1}{2}(lb - ub)) \odot b$$

if direction_type is soft_binary

$$v = \text{sgn}(x + \frac{1}{2}(lb - ub)) \odot b$$

we then define

$$h = -\nabla f(x)$$

and

$$g = -\phi'(\phi^{-1}(x)) \odot \nabla f(x)$$

if absorption_criterion is not None then

$$b \leftarrow b \odot u$$

$$h \leftarrow h \odot u$$

we then normalize theses arrays

$$b \leftarrow \frac{b}{\|b\|_2},\ h \leftarrow \frac{h}{\|h\|_2},\ g \leftarrow \frac{g}{\|g\|_2}$$

we then define

$$w = \gamma b + (1 - \gamma)h$$

and

$$y = \max\left(0, g^\top w + \arctan(\theta)\sqrt{\|w\|_2^2 - (g^\top w)^2}\right)$$

Finally,

$$d = (w + yg) \odot u$$

2.10 _absorb_solution_to_limits

is a function that takes as an input the problem, options and x and returns a vector $v \in \mathbb{R}^n$, such that the components v_i is lb_i if $|x_i - lb_i| \leq$ absorption_criterion. Similarly, $v_i = ub_i$ if $|x_i - ub_i| \leq$ absorption_criterion. Otherwise $v_i = x\_i$.

$$(\text{problem,options}, x) \longrightarrow \boxed{\text{\_absorb\_solution\_to\_limits}} \longrightarrow v \in \mathbb{R}^n$$

2.11 _inverse_activation

Is the inverse of the activation function ϕ^{-1}

2.12 _activation

Is the activation function ϕ

2.13 _proxy_distance_vector

Is the function $\phi'(\phi^{-1})$

2.14 _inequality_constraint_problem