ParaQapNB : Massively Parallelized DNN-based Branch-and-bound QAP solver

*Koichi Fujii, Naoki Ito, Sunyoung Kim,
Masakazu Kojima, Yuji Shinano, Kim-Chuan Toh

NTT DATA Mathematical Systems Inc., FAST RETAILING CO., LTD.,
Ewha W. University, Chuo University, Zuse Institute Berlin,
National University of Singapore

2nd UG workshop, 01 October 2021
16 December 2021 revised

Summary : DNN-based Branch-and-bound for the Quadratic Assignment Problem

Motivation

- Quadratic assignment problems still remain as one of the most difficult combinatorial problems
- Recent conic relaxation technique updates the lower bounds of (open) quadratic assignment problem

Goal

Solve all open instances of QAPLIB

Our Results

Our DNN-based branch-and-bound solver solved two open instances (tai30a, sko42).

Summary: Quadratic Assignment Problem

- n facilities $i = 1, \ldots, n$: f_{ij} : flow from facility i to facility j.
- n locations $k = 1, \ldots, n$: $d_{k\ell}$: distance from loc k to loc ℓ.
- Assign each facility to each location to minimize flow $\times$ distance.

$n = 4$, Flow f_{ij}				
	j			
i	1	2	3	4
1	0	5	4	2
2	4	0	2	5
3	2	3	0	4
4	1	5	3	0

Distance $d_{k\ell}$				
	ℓ			
k	1	2	3	4
1	0	6	7	8
2	6	0	9	6
3	7	9	0	8
4	8	6	8	0

Assignment				
i	1	2	3	4
k	k_1	k_2	k_3	k_4
	$\|\|$	$\|\|$	$\|\|$	$\|\|$
	4	1	3	2

QAP : minimize the total cost $= \sum_{i,\, j} f_{ij} d_{k_i k_j}$

subject to $k_1, \ldots, k_n$: a permutation of $1, \ldots, n$

- $n!$ feasible solutions; $40! = 8.2e47$ — too large combination

Summary: DNN relaxation

Polynomial optimization problem (POP) with non-negative variables

$\min_x \{f_0(x) \mid f_i(x) = 0 \ (i = 1, 2, \ldots, m), x \geq 0\}$

- 0-1 binary quadratic optimization problem
- Optimal power flow, sensor network localization, ...

Doubly non-negative (DNN) relaxation

$$\min_{\boldsymbol{X}} \left\{ \langle \boldsymbol{A}_0, \ \boldsymbol{X} \rangle \ \middle| \ \begin{array}{l} \langle \boldsymbol{A}_i, \ \boldsymbol{X} \rangle = \boldsymbol{b} \ (i = 1, 2, ..., m), \ \boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2 \\ \mathbb{K}_1 : \text{semidefinite cone}, \ \mathbb{K}_2 : \text{polyhedral cone} \end{array} \right\}$$

SDP relaxation + non-negative constraints

- better lower bounds than SDP
- very large $O(n^2)$ non-negative constraints

Summary: Newton-bracketing method

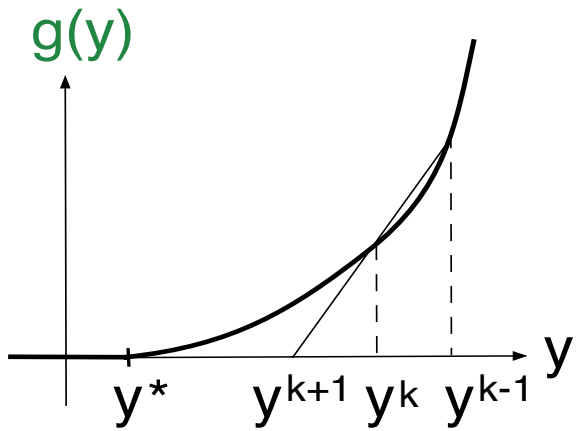
- ζ^* : The optimal value of QAP
- y^* : The optimal value of Lagrangian DNN relaxation; $y^* \leq \zeta^*$.

Newton-bracketing method

Lagrangian DNN relaxation problem can be regarded as minimization problem of convex function $g : \mathbb{R} \to \mathbb{R}_+$.

- Newton method (+ secant method)
  $y^* \leftarrow y^{r+1} \leq y^r \leq \cdots \leq y^0$.
- (Valid) lower bound
  $\ell^1 \leq \cdots \leq \ell^r \leq \ell^{r+1} \to y^*$

$y^* \in [\ell^r, y^r]$, $|y^r - \ell^r| \to 0$.



Agenda

1. Lagrangian DNN relaxation of QAP
2. DNN Optimization and Newton-bracketing method
3. QapNB: DNN-based Branch-and-bound solver for QAP
4. ParaQapNB: parallelized DNN-based branch-and-bound solver for QAP

Lagrangian DNN relaxation of QAP

QAP(01-QOP)

$$\zeta^* := \min \left\{ \langle \boldsymbol{B} \otimes \boldsymbol{A},\ \boldsymbol{x}\boldsymbol{x}^T \rangle \;\middle|\; \begin{array}{l} \boldsymbol{x} \in \{0,1\}^m \\ (\boldsymbol{I} \otimes \boldsymbol{e}^T)\boldsymbol{x} = (\boldsymbol{e}^T \otimes \boldsymbol{I})\boldsymbol{x} = \boldsymbol{e} \end{array} \right\}.$$

where $\boldsymbol{A} \times \boldsymbol{B}$ denotes Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{B}$.

$\Downarrow$ Replace $\begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^T$ by $\boldsymbol{X} \in \mathbb{K}_1 \equiv \mathbb{S}_+^{1+m}$ with $X_{pq} \geq 0$ $(0 \leq p, q \leq m)$.

$$\eta^* = \min \left\{ \langle \boldsymbol{Q},\ \boldsymbol{X} \rangle : \begin{array}{l} \boldsymbol{X} \in \mathbb{K}_1,\ X_{00} = 1,\ \langle \boldsymbol{H}^1,\ \boldsymbol{X} \rangle = 0, \\ X_{pq} \geq 0, X_{p0} = X_{pp}\ (1 \leq p, q \leq m) \end{array} \right\} \leq \zeta^*,$$

where $\boldsymbol{H}^1 \in \mathbb{K}_1 \equiv \mathbb{S}_+^{1+m}$ (semidefinite cone).

Lagrangian DNN relaxation of QAP

$$\eta^* = \min\left\{ \langle \boldsymbol{Q}, \boldsymbol{X}\rangle : \begin{array}{l} \boldsymbol{X} \in \mathbb{K}_1,\ X_{00} = 1,\ \langle \boldsymbol{H}^1, \boldsymbol{X}\rangle = 0, \\ X_{pq} \geq 0, X_{p0} = X_{pp}\ (1 \leq p, q \leq m) \end{array} \right\} \leq \zeta^*,$$

ここで $\boldsymbol{H}^1 \in \mathbb{K}_1 \equiv \mathbb{S}^{1+m}_+$ (semidefinite cone).

$$\Downarrow \quad \begin{array}{l} \boldsymbol{H}^0 \in \mathbb{S}^{1+m}_+;\ H^0_{00} = 1,\ H^0_{pq} = 0\ ((p,q) \neq (0,0)) \Rightarrow X_{00} = \langle \boldsymbol{H}^0, \boldsymbol{X}\rangle, \\ \mathbb{K}_2 \equiv \{\boldsymbol{X} \in \mathbb{S}^{1+m} : X_{pq} \geq 0, X_{0p} = X_{p0} = X_{pp}\ (1 \leq p, q \leq m)\} \text{ (poly.)} \end{array}$$

$$\eta^* = \min\left\{ \langle \boldsymbol{Q}, \boldsymbol{X}\rangle : \boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2, \langle \boldsymbol{H}^0, \boldsymbol{X}\rangle = 1, \langle \boldsymbol{H}^1, \boldsymbol{X}\rangle = 0 \right\}.$$

$$\Downarrow \quad \begin{array}{l} \langle \boldsymbol{H}^1, \boldsymbol{X}\rangle \geq 0,\ \forall \boldsymbol{X} \in \mathbb{K}_1 \equiv \mathbb{S}^{1+m}_+, \\ \boldsymbol{Q}^\lambda \equiv \boldsymbol{Q} + \lambda \boldsymbol{H}^1\ (\lambda \geq 0) \quad \swarrow \text{ a penalty term} \\ \langle \boldsymbol{Q}^\lambda, \boldsymbol{X}\rangle \equiv \langle \boldsymbol{Q}, \boldsymbol{X}\rangle + \lambda \langle \boldsymbol{H}^1, \boldsymbol{X}\rangle \geq \langle \boldsymbol{Q}, \boldsymbol{X}\rangle\ \forall \boldsymbol{X} \in \mathbb{S}^{1+m}, \lambda \geq 0. \end{array}$$

Lagrangian DNN relaxation of 01-QOP:

$$\text{P: } \eta^\lambda = \min\left\{ \langle \boldsymbol{Q}^\lambda, \boldsymbol{X}\rangle : \boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2, \langle \boldsymbol{H}^0, \boldsymbol{X}\rangle = 1\ (X_{00} = 1) \right\}$$

$$\zeta^* \text{ (QAP opt. val)} \geq \eta^* \geq \eta^\lambda \to \eta^* \text{ as } 0 \leq \lambda \to \infty.$$

Lagrangian DNN relaxation of QAP

QAP(01-QOP)

$$\zeta^* := \min \left\{ \langle \boldsymbol{B} \otimes \boldsymbol{A},\ \boldsymbol{x}\boldsymbol{x}^T \rangle \;\middle|\; \begin{array}{l} \boldsymbol{x} \in \{0,1\}^m \\ (\boldsymbol{I} \otimes \boldsymbol{e}^T)\boldsymbol{x} = (\boldsymbol{e}^T \otimes \boldsymbol{I})\boldsymbol{x} = \boldsymbol{e} \end{array} \right\}.$$

$$\Downarrow$$

Lagrangian DNN relaxation of QAP

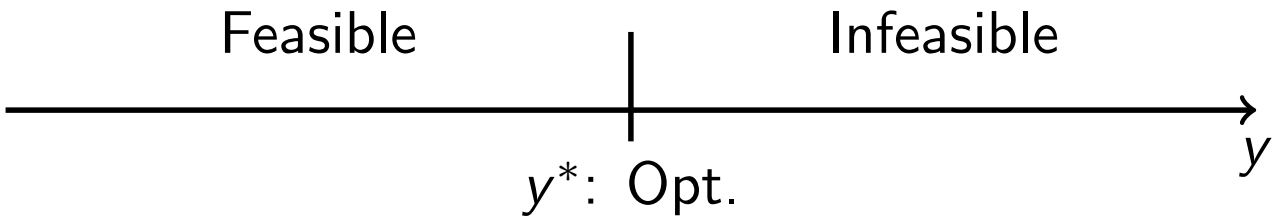
$$\text{P: } \eta^\lambda = \min \left\{ \langle \boldsymbol{Q}^\lambda,\ \boldsymbol{X} \rangle : \ \boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2, \langle \boldsymbol{H}^0,\ \boldsymbol{X} \rangle = 1\ (X_{00} = 1) \right\}.$$

$$\mathbb{K}_1 \quad : \quad \text{semidefinite cone} \tag{1}$$

$$\mathbb{K}_2 \quad := \quad \left\{ \boldsymbol{X} \in \mathbb{S}^{1+m} : \begin{array}{l} \boldsymbol{X}_{\alpha\beta} \geq 0 \text{ (nonnegative)} \\ \boldsymbol{X}_{0\alpha} = \boldsymbol{X}_{\alpha 0} = \boldsymbol{X}_{\alpha\alpha} \end{array} \right\} \tag{2}$$

DNN Optimization

$$\text{P: } \eta^\lambda = \min\left\{\langle \boldsymbol{Q}^\lambda,\ \boldsymbol{X}\rangle : \boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2, \langle \boldsymbol{H}^0,\ \boldsymbol{X}\rangle = 1\ (X_{00}=1)\right\}$$
$$\Updownarrow \text{ Strong duality}$$
$$\text{D: } y^* = \max\{y \in \mathbb{R} : \underbrace{\boldsymbol{Q}^\lambda - \boldsymbol{H}^0 y}_{\boldsymbol{G}(y)} \in (\mathbb{K}_1 \cap \mathbb{K}_2)^* \equiv \mathbb{K}_1^* + \mathbb{K}_2{}^*\} = \eta^\lambda,$$



BP method ([Kim, Kojima, & Toh, '16], [Ito, et al., '18]) : Bisection method

Newton-bracketing method ([Kim, Kojima, & Toh, '20]) : Netwton method (+ secant method).

DNN Optimization

How to judge if $\boldsymbol{G}(y_0) \in \mathbb{K}_1^* + \mathbb{K}_2^*$? $\Rightarrow$ solve regression model

$$
\begin{aligned}
f^* &= \min_{\boldsymbol{Y}_1, \boldsymbol{Y}_2} \{ \|\boldsymbol{G} - (\boldsymbol{Y}_1 + \boldsymbol{Y}_2)\|^2 \mid \boldsymbol{Y}_1 \in \mathbb{K}_1^*,\ \boldsymbol{Y}_2 \in \mathbb{K}_2^* \} \\
&= \min_{\boldsymbol{Y}_1} \{ \min_{\boldsymbol{Y}_2} \{ \|(\boldsymbol{G} - \boldsymbol{Y}_1) - \boldsymbol{Y}_2\|^2 \mid \boldsymbol{Y}_2 \in \mathbb{K}_2^* \} \mid \boldsymbol{Y}_1 \in \mathbb{K}_1^* \} \\
&= \min_{\boldsymbol{Y}_1} \{ \|(\boldsymbol{G} - \boldsymbol{Y}_1) - \Pi_{\mathbb{K}_2^*}(\boldsymbol{G} - \boldsymbol{Y}_1))\|^2 \mid \boldsymbol{Y}_1 \in \mathbb{K}_1^* \} \\
&= \min_{\boldsymbol{Y}_1} \{ \|\Pi_{\mathbb{K}_2}(\boldsymbol{Y}_1 - \boldsymbol{G})\|^2 \mid \boldsymbol{Y}_1 \in \mathbb{K}_1^* \} \quad (\text{where } \boldsymbol{Y}_2 = \Pi_{\mathbb{K}_2^*}(\boldsymbol{G} - \boldsymbol{Y}_1))
\end{aligned}
$$

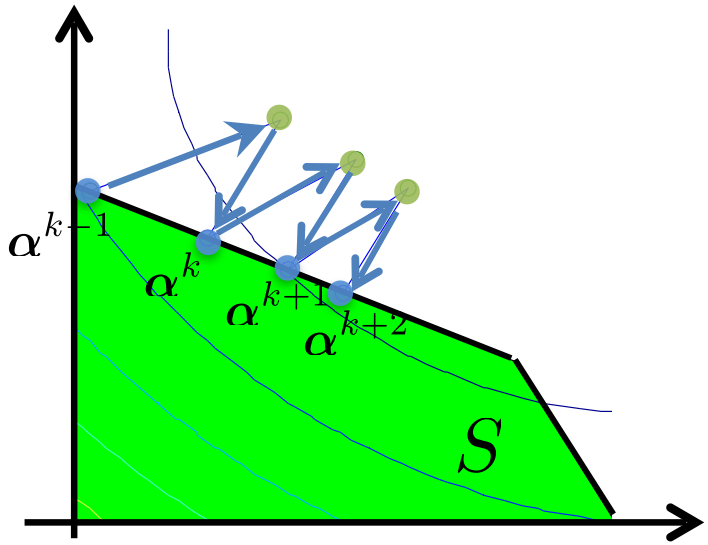
- Obviously, $f^* = 0 \Leftrightarrow \boldsymbol{G} \in \mathbb{K}_1^* + \mathbb{K}_2^*$.
- Apply accelerated proximal gradient (APG) to check if $f^* = 0$.
  $\rightarrow$ [Assumption 1] $\Pi_{\mathbb{K}_2}, \Pi_{\mathbb{K}_1}$ can be computed efficiently.

DNN Optimization : APG method

Constrained optimization: $\min_{\boldsymbol{\alpha} \in S} f(\boldsymbol{\alpha})$

Gradient projection method (e.g., [Goldstein, '64])

Step 1: $\boldsymbol{\alpha}^{k+1} = \Pi_S\left(\boldsymbol{\alpha}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\alpha}^k)\right)$

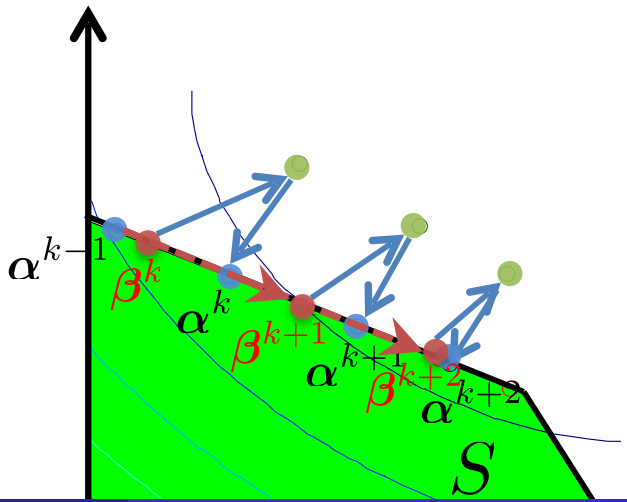


APG method [Beck and Teboulle, '09]

Step 1: $\boldsymbol{\alpha}^k = \Pi_S\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$

Step 2: $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

Step 3: $\boldsymbol{\beta}^{k+1} = \boldsymbol{\alpha}^k + \underbrace{\frac{t_k - 1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})}_{\text{momentum}}$



DNN Optimization: Newton-bracketing method

$$\begin{aligned}
&\text{P: } \eta^\lambda = \min\left\{\langle \boldsymbol{Q}^\lambda,\ \boldsymbol{X}\rangle : \boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2, \langle \boldsymbol{H}^0,\ \boldsymbol{X}\rangle = 1\ (X_{00}=1)\right\}\\
&\Updownarrow \text{ Strong duality}\\
&\text{D: } y^* = \max\{y \in \mathbb{R} : \underbrace{\boldsymbol{Q}^\lambda - \boldsymbol{H}^0 y}_{\boldsymbol{G}(y)} \in (\mathbb{K}_1 \cap \mathbb{K}_2)^* \equiv \mathbb{K}_1^* + \mathbb{K}_2{}^*\} = \eta^\lambda,
\end{aligned}$$

Define function $g : \mathbb{R} \to \mathbb{R}_+$ as:

$$g(y) \quad := \quad \|\boldsymbol{Q}^\lambda - \boldsymbol{H}^0 y - (\widehat{\boldsymbol{Y}}_1(y) + \widehat{\boldsymbol{Y}}_2(y))\| = \|\widehat{\boldsymbol{X}}(y)\|. \tag{3}$$

It holds:

Lemma (Arima, Kim, Kojima & Toh, '18)

- *$g : \mathbb{R} \to \mathbb{R}_+$ is a continuous and convex function.*
- *If $y > y^*$ holds, $g'(y) = dg(y)/dy = \langle \boldsymbol{H},\ \widehat{\boldsymbol{X}}(y)\rangle / g(y) > 0$. In this case, g is differentiable and monotonous increasing.*

DNN Optimization: Newton-bracketing method

Netwton method of Lagrangian DNN relaxation

Step 0: Start with upper bound $y^0 > y^*$.

Step 1: Obtain $(\boldsymbol{X}, \boldsymbol{Y}_1, \boldsymbol{Y}_2) \in (\mathbb{K}_1 \cap \mathbb{K}_2) \times \mathbb{K}_1^* \times \mathbb{K}_2^*$ satisfying KKT condition by APGR method.

Step 2: Update y^k by Newton iteration $y^{k+1} := y^k - \frac{g(y^k)}{g'(y^k)}$.

Step 3: $k \leftarrow k+1$, goto Step1.

Theorem (Kim, Kojima, & Toh, '20)

(i) y^k *converges to* y^* *as* $k \to \infty$.

(ii) Assume that $\boldsymbol{Q}^\lambda - \boldsymbol{H}^0 y$ *lies in the interior of* $\mathbb{K}_1^* + \mathbb{K}_2^*$ *for some* y. *Then the convergence of* y^k *to* y^* *are quadratic.*

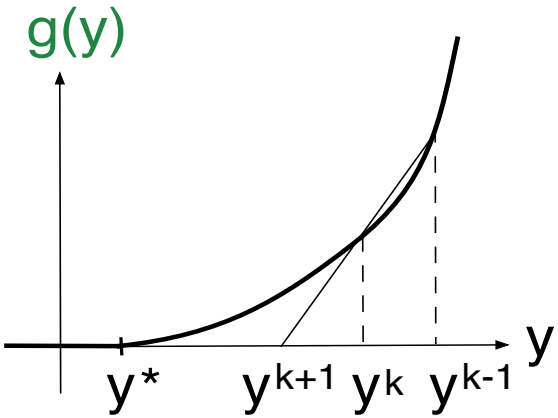
DNN Optimization : Newton-bracketing method

$$y^{k+1} = y^k - \frac{g(y^k)}{g'(y^k)} = \langle \boldsymbol{Q}^\lambda, \widetilde{\boldsymbol{X}}(y^k) \rangle \geq \eta^\lambda$$

$\widetilde{\boldsymbol{X}}(y^k)$: feasible solution of Lagrangian DNN relaxation.

$$y^* \leftarrow y^{k+1} \leq y^k \leq y^{k-1} \leq \cdots \leq y^0$$

y^0 : upper bound of y^*



$\Leftarrow$ For $k > 1$, apply **secant method** instead of Newton method.

DNN Optimization: Newton-bracketing method(lower bound)

$$\text{P: } \eta^\lambda = \min\left\{\langle \boldsymbol{Q}^\lambda,\ \boldsymbol{X}\rangle : \langle \boldsymbol{H}^0,\ \boldsymbol{X}\rangle = 1, \boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2)\right\}$$

$$\Downarrow \text{ Identity matrix } \boldsymbol{I} \in \mathbb{K}_1 \text{ and sufficiently large } \rho \geq 0$$

$$\min_{\boldsymbol{X}}\{\langle \boldsymbol{Q}^\lambda,\ \boldsymbol{X}\rangle \mid \langle \boldsymbol{H}^0,\ \boldsymbol{X}\rangle = 1,\ \langle \boldsymbol{I},\ \boldsymbol{X}\rangle \leq \rho,\ \boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2\}$$

$$\Updownarrow \text{ Strong duality}$$

$$\max_{y,\mu}\{y + \rho\mu \mid \underbrace{\boldsymbol{Q}^\lambda - y\boldsymbol{H}^0}_{\boldsymbol{G}(y)} - \mu\boldsymbol{I} \in \mathbb{K}_1^* + \mathbb{K}_2^*,\ \mu \leq 0\}$$

$$\Updownarrow$$

$$\max_{y,\mu,\boldsymbol{Y}_2}\{y + \rho\mu \mid \boldsymbol{G}(y) - \boldsymbol{Y}_2 - \mu\boldsymbol{I} \in \mathbb{K}_1^*(= \mathbb{S}_+^n),\ \boldsymbol{Y}_2 \in \mathbb{K}_2^*,\ \mu \leq 0\}$$

DNN Optimization: Summary

$$\text{P: } \eta^\lambda = \min\left\{\langle \boldsymbol{Q}^\lambda,\ \boldsymbol{X}\rangle : \langle \boldsymbol{H}^0,\ \boldsymbol{X}\rangle = 1, \boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2)\right\}$$

Dual of Lagrangian relaxation with parameter $\rho \geq 0$

$$\max_{y,\mu,\boldsymbol{Y}_2} \{y + \rho\mu \mid \boldsymbol{G}(y) - \boldsymbol{Y}_2 - \mu\boldsymbol{I} \in \mathbb{K}_1^*(= \mathbb{S}_+^n),\ \boldsymbol{Y}_2 \in \mathbb{K}_2^*,\ \mu \leq 0\}$$

Newton-bracketing method

- Step 0: Start with upper bound $y^0 > y^*$.
- Step 1: Obtain $(\boldsymbol{X}, \boldsymbol{Y}_1, \boldsymbol{Y}_2) \in (\mathbb{K}_1 \cap \mathbb{K}_2) \times \mathbb{K}_1^* \times \mathbb{K}_2^*$ satisfying KKT condition by APGR method.
- Step 2: Update y^k by Newton iteration $y^{k+1} := y^k - \frac{g(y^k)}{g'(y^k)}$ or secant iteration.
- Step 3: $\mu \leftarrow$ min eig. of $\boldsymbol{G}(y^k) - \boldsymbol{Y}_2$, $\ell^k \leftarrow \max\{y^k + \rho\mu, \ell^k\}$
- Step 4: Obtain $[\ell^k, y^k](\ni \eta^\lambda)$, $k \leftarrow k+1$, goto Step1.

QapNB: DNN-based Branch-and-bound solver for QAP

What is different from MIP solver?

- Node relaxation is much harder.
- No warm start.
- No presolving.

solver	#nodes	time/node
NuOpt (LP-based B & B)	1.74e+8	4.66-4s
QapNB	195	1.97s

Table: comparison between LP-based and DNN-based branch-and-bound with chr20a

QapNB: DNN-based Branch-and-bound solver for QAP

QAP

$$\zeta^* := \min \left\{ \langle \boldsymbol{B} \otimes \boldsymbol{A},\ \boldsymbol{x}\boldsymbol{x}^T \rangle \;\middle|\; \begin{array}{l} \boldsymbol{x} \in \{0,1\}^m \\ (\boldsymbol{I} \otimes \boldsymbol{e}^T)\boldsymbol{x} = (\boldsymbol{e}^T \otimes \boldsymbol{I})\boldsymbol{x} = e \end{array} \right\}.$$

(1) Incumbent solution

- Apply robust tabu search method （ [Tailard 1991] ） to obtain global upper bound $\hat{\zeta}$.

(2) Stop condition of Newton-bracketing method

[Condition 1] Stop if lower bound of DNN ℓ^k exceeds $\hat{\zeta}$
$\rightarrow$ prune the node

[Condition 2] Stop if upper bound of DNN y^k gets below $\hat{\zeta}$
$\rightarrow$ branch on the node

(3) Branching strategy (Polytomic branching)

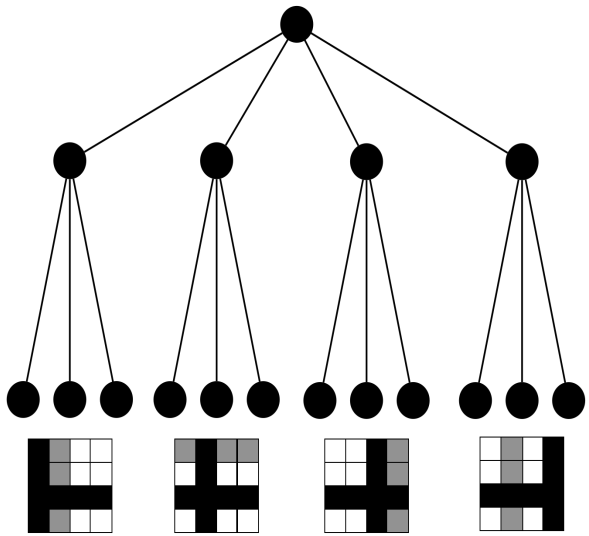
QapNB: DNN-based Branch-and-bound solver for QAP

QAP

$$\zeta^* := \min \left\{ \langle \boldsymbol{B} \otimes \boldsymbol{A},\ \boldsymbol{x}\boldsymbol{x}^T \rangle \;\middle|\; \begin{array}{l} \boldsymbol{x} \in \{0,1\}^m \\ (\boldsymbol{I} \otimes \boldsymbol{e}^T)\boldsymbol{x} = (\boldsymbol{e}^T \otimes \boldsymbol{I})\boldsymbol{x} = e \end{array} \right\}.$$

(3) Branching strategy (Polytomic branching)

1. Select facility $f \rightarrow$ fix $x_{f,l} = 1$ for $\forall \ell$.
2. Select location $\ell \rightarrow$ fix $x_{f,l} = 1$ for $\forall f$.



QapNB: DNN-based Branch-and-bound solver for QAP

Branching strategy

Estimate the score $\varphi(f, \ell)$ for unfixed facility f and location ℓ.

average branching

$\varphi(f, \ell) :=$ avg. obj. value of feasible solutions in $QAP(F \cup f, L \cup \ell)$

primal DNN branching

Step1. Obtain primal feasible solution of DNN Lagrangian relaxation $\widehat{\boldsymbol{X}} := \boldsymbol{X}/\langle \boldsymbol{H}^0, \boldsymbol{X} \rangle$.

Step2. Project primal solution $\widehat{\boldsymbol{X}}$ onto subproblem space : $\widehat{\boldsymbol{X}} \mapsto \widehat{\boldsymbol{X}}(f, \ell)$

Step3. $\varphi(f, \ell) := \langle \boldsymbol{Q}^0((F \cup f), (L \cup \ell)), \widehat{\boldsymbol{X}}(f, \ell) \rangle$.

Computational Results of QapNB

problem	ADMM([1])		BBCPOP([2])		QapNB	
	#node	time(s)	#node	time(s)	#node	time(s)
nug12	23	43.74	35	31.39	35	20.77
nug14	14	49.56	28	73.64	15	28.67
nug15	15	147.85	72	138.59	16	34.86
nug16a	16	144.84	46	194.35	17	42.05
nug16b	31	419.06	197	384.59	17	57.13
nug17	188	1151.46	191	541.90	50	97.23
nug18	805	5071.32	355	1532.24	104	143.47

Table: ADMM vs BBCPOP vs NewtonBracket

Mac OSX 10.14.2, Intel(R) Core(TM) i5-8500B CPU @ 3.00GHz

[1] Zhenyu Liao. (2016). Branch and bound via the alternating direction method of multipliers for the quadratic assignment problem.

[2] Koichi Fujii, Naoki Ito, Yuji Shinano (2019). DNN-based Branch-and-bound for the Quadratic Assignment Problem.

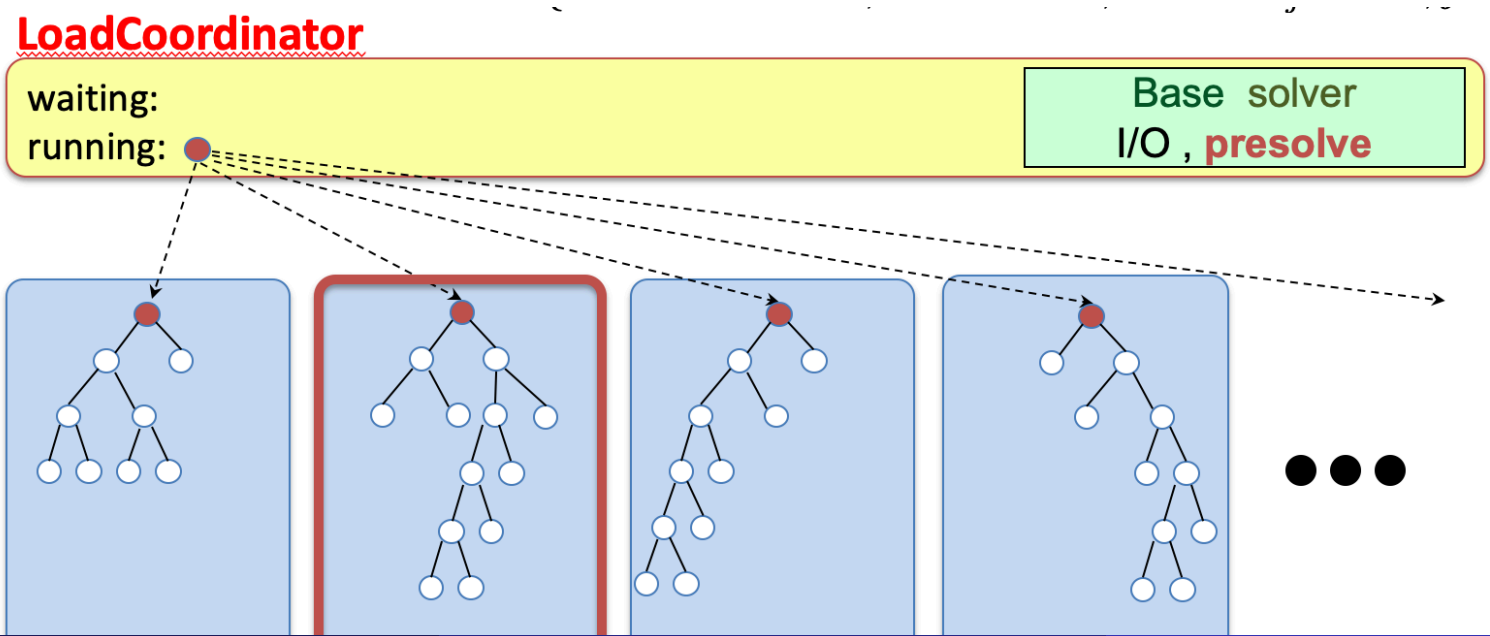
Computational Results of QapNB

problem	average branching		primal DNN branching	
	#node	time(s)	#node	time(s)
nug12	35	20.77	13	19.66
nug14	15	28.67	15	28.97
nug15	16	34.86	16	35.03
nug16a	17	42.05	17	42.35
nug16b	17	57.13	17	59.36
nug17	50	97.23	50	98.58
nug18	104	143.47	87	128.86
nug20	685	1210.33	247	509.72
nug21	312	745.53	218	472.78
nug22	429	1083.88	290	596.18
nug24	1245	4978.72	586	2807.63
tai10a	11	13.42	11	13.79
tai10b	11	12.72	11	12.72
tai12a	34	18.50	34	18.66
tai12b	34	24.18	34	25.16
tai15a	188	100.39	201	108.42
tai15b	76	53.19	76	54.05
tai17a	364	226.79	269	169.67
tai20a	2464	3245.62	2525	3169.68
tai20b	166	172.22	166	185.95
tai25b	372	1248.68	372	1376.28

ParaQapNB: parallelized DNN-based branch-and-bound solver for QAP

UG : Ubiquity Generator Framework

- C++ parallel branch-and-bound framework
- NuOpt, SCIP and Xpress has been parallelized with UG
- New features (Self-splitting ramp-up, etc.) were added for ParaQapNB



ParaQapNB: parallelized DNN-based branch-and-bound solver for QAP

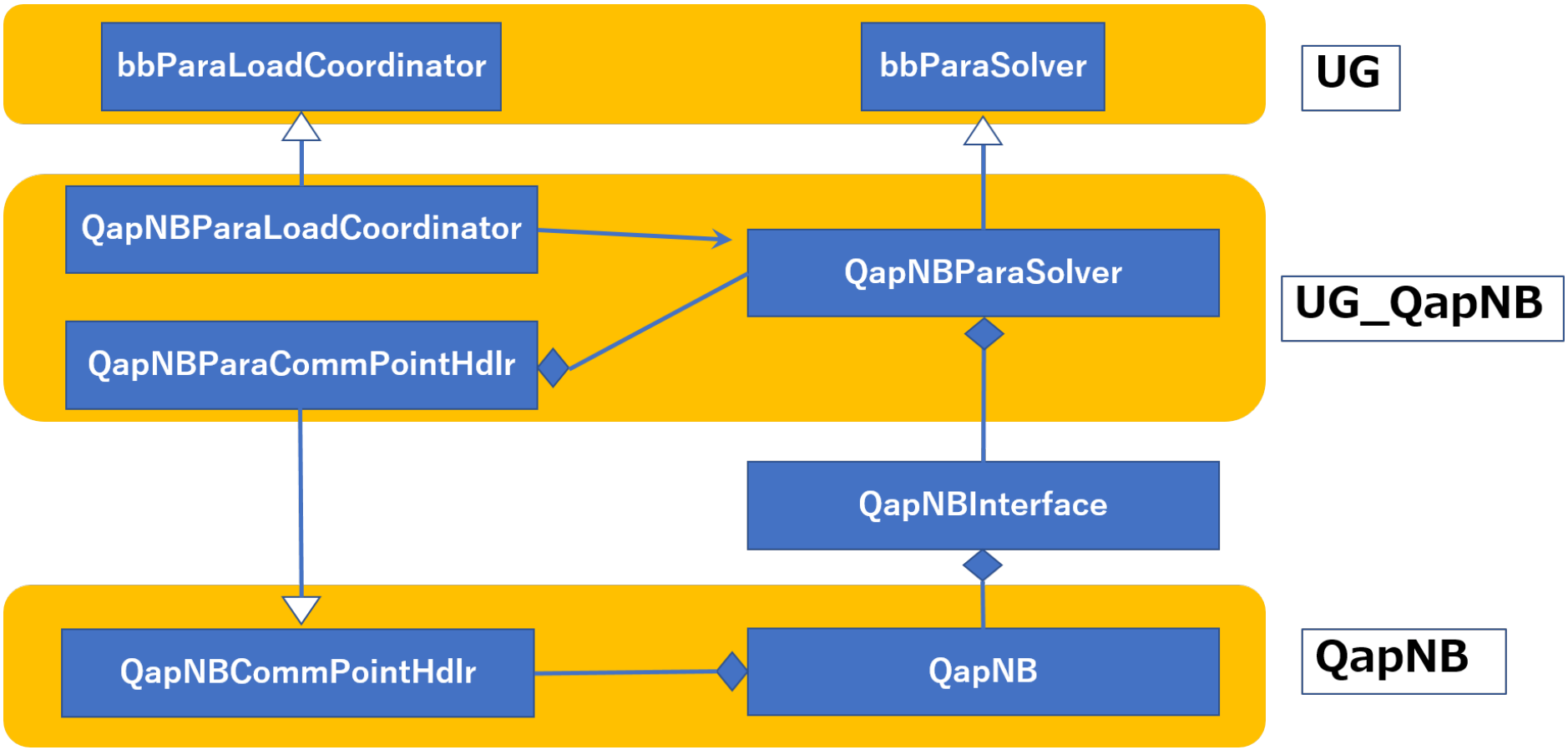
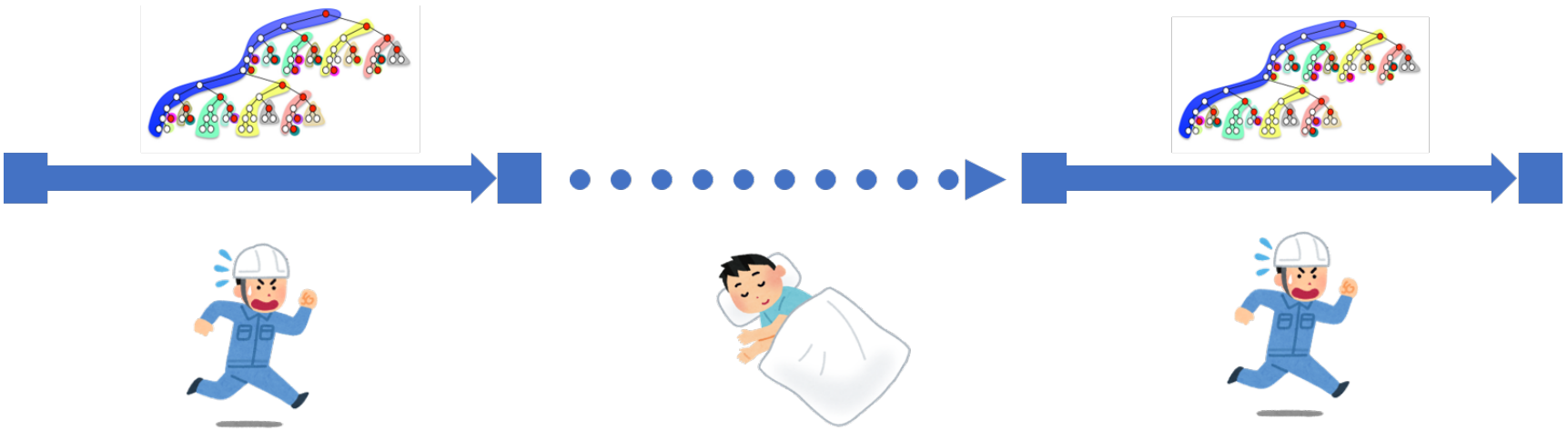


Figure: Design of ParaQapNB

ParaQapNB: parallelized DNN-based branch-and-bound solver for QAP

TIPS for UG (prospective) users:

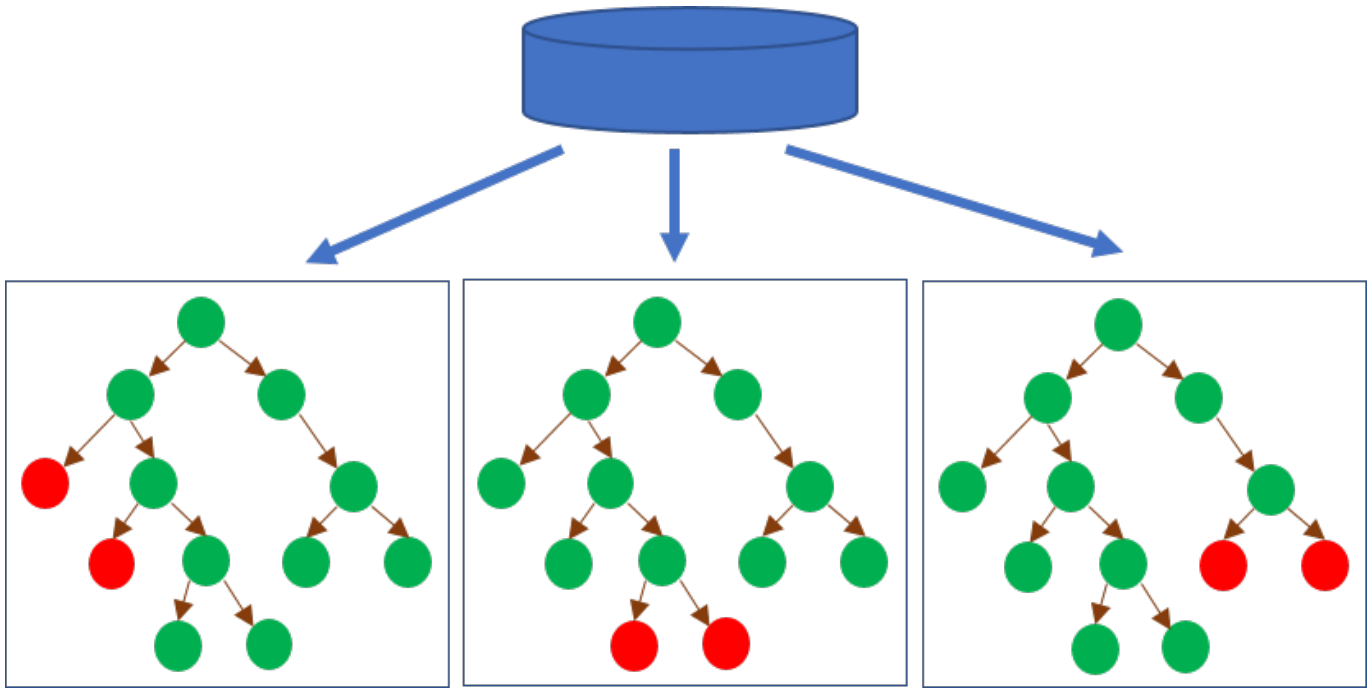
- Insert callback code into your solver
- The callbacks should be called as much frequently as possible
- Do not worry about load balancing, UG will take care
- Supercomputer is not always available
- Test carefully, especially multiple checkpoint loading (how to check validity)
- Enjoy working with Yuji (UG)!



ParaQapNB: parallelized DNN-based branch-and-bound solver for QAP

Self-splitting [M. Fischetti et al. 2014]

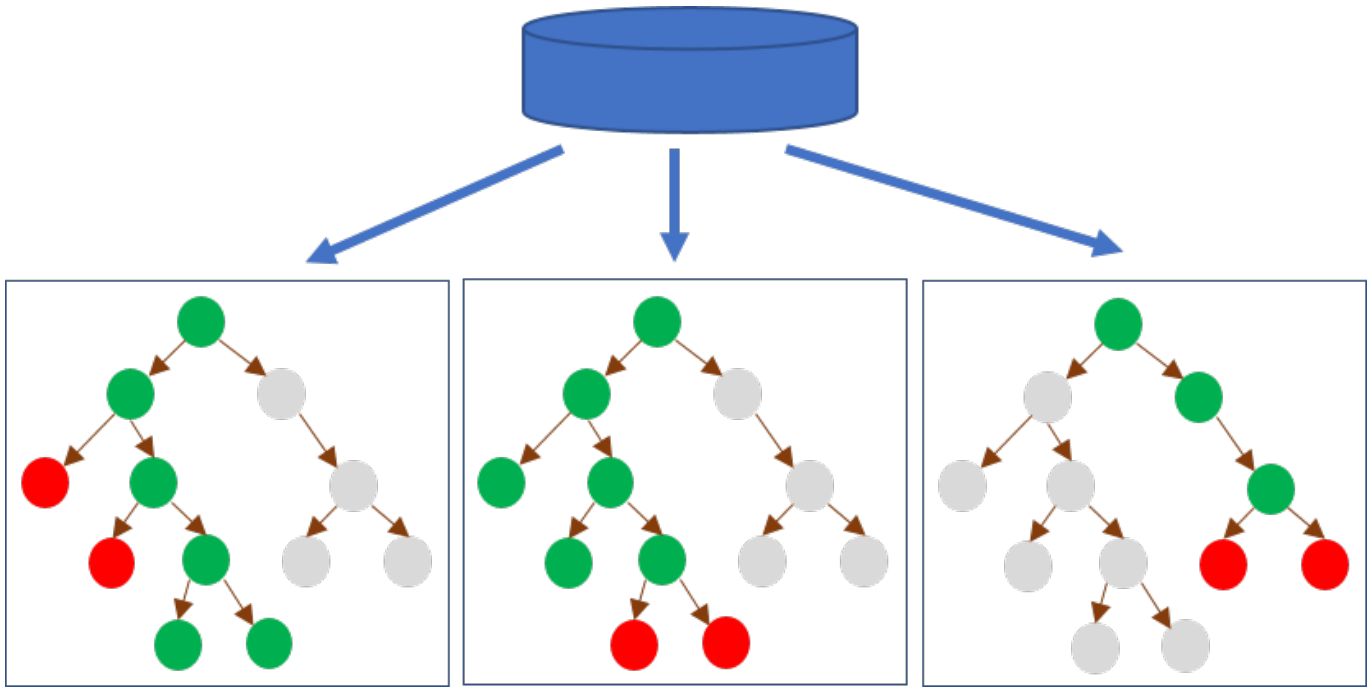
- The same enumeration tree is initially build by all workers.
- After the sampling phase, each workers solve the nodes which belong to it.
- Assumuption: sampling phase is not a bottleneck.



ParaQapNB: parallelized DNN-based branch-and-bound solver for QAP

Self-splitting for ParaQapNB

- Avoid redundant sampling
- Only with light sampling (branching rule average only)



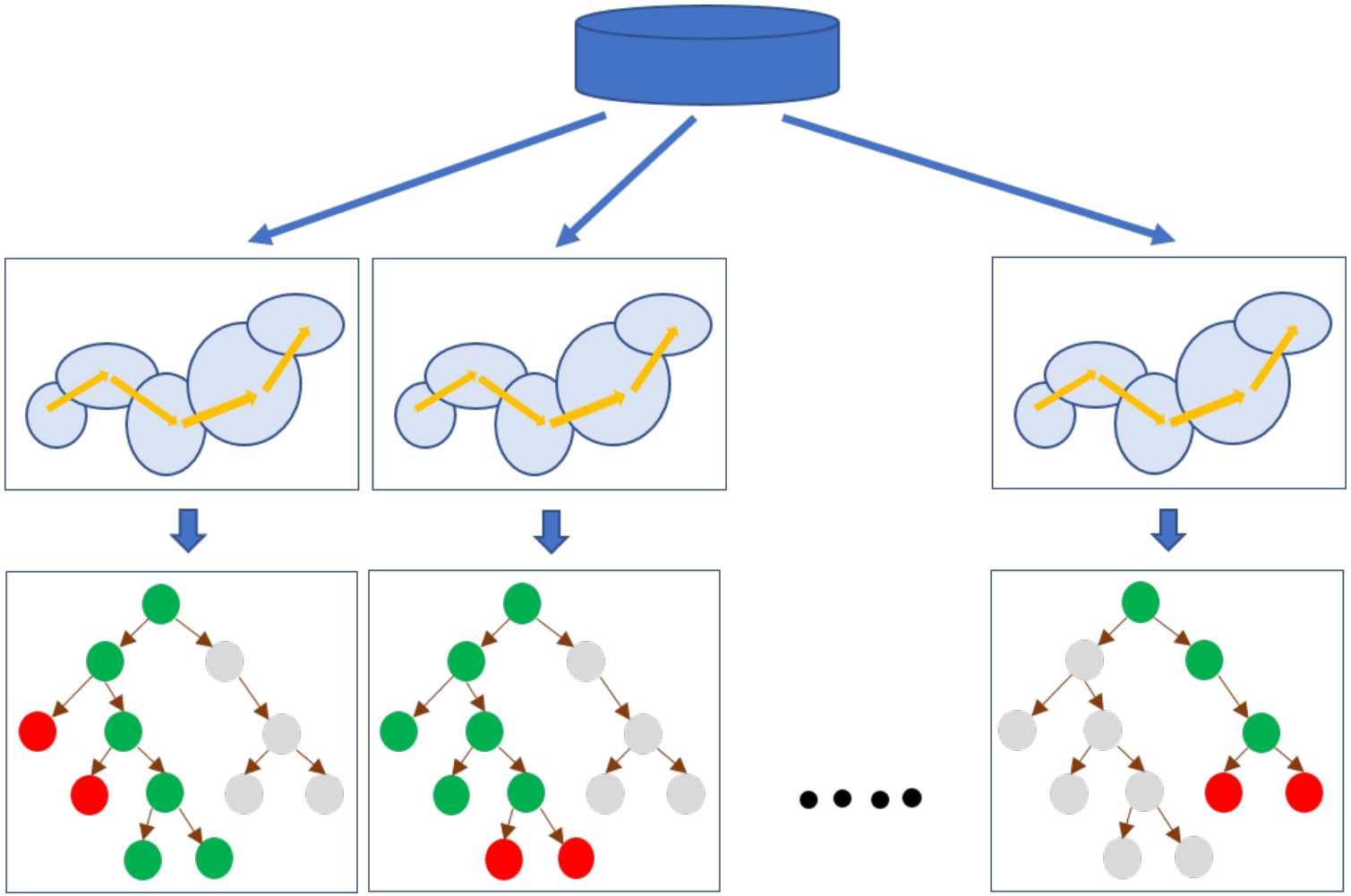
ParaQapNB: parallelized DNN-based branch-and-bound solver for QAP

Self-splitting + heuristics for ParaQapNB

- Robust Tabu Search [Tailard 1991] + randomization
- Extremal Optimization [Boettcher et al 2000]
- Memetic Iterated Tabu Search [Silva et al 2020]
- Genetic Algorithm [Tate et al 1995]

ParaQapNB: parallelized DNN-based branch-and-bound solver for QAP

Self-splitting + heuristics for ParaQapNB



Computational Results of ParaQapNB

problem	solver=1		solver=3		solver=5	
	#node	time(s)	#node	time(s)	#node	time(s)
nug18	104	165.31	104	74.86	104	72.69
nug20	685	1882.15	685	786.94	685	662.80
nug21	312	1002.70	312	428.34	312	366.66
nug22	689	2939.45	429	776.80	429	672.71
nug24	1245	9076.03	1245	3817.10	1245	3428.92
nug25	8270	59752.86	8270	25827.81	8270	22958.26
nug27	2610	39234.07	2610	16506.58	2610	14303.89
tai17a	379	364.66	379	143.17	394	121.29
tai20a	2554	5193.36	2949	2258.41	3069	1988.36
tai20b	166	207.62	166	183.67	166	196.56
tai25b	919	4019.42	1020	2417.45	659	1689.15

Table: computational results of ParaQapNB on medium instances

Computational Results of ParaQapNB

problem	depth=1		depth=2		depth=3		depth=4	
	#node	time(s)	#node	time(s)	#node	time(s)	#node	time(s)
nug18	104	72.69	104	50.55	325	31.52	5221	110.95
nug20	685	662.80	685	688.19	761	588.25	7241	630.67
nug21	312	366.66	312	364.48	632	373.88	8422	684.37
nug22	429	672.71	429	682.61	765	686.17	9725	1363.23
nug24	1245	3428.92	1245	3382.96	1521	3219.32	13225	4805.34
nug25	8270	22958.26	8270	22221.25	8558	21889.72	20242	23278.26
nug27	2610	14303.89	2610	13885.30	3026	13715.36	19376	19939.51
nug28	10291	60081.25	10921	57520.11	11245	62761.53	28145	63198.93
tai17a	394	121.29	379	95.17	394	92.54	4489	161.87
tai20a	3069	1988.36	3304	1973.81	3088	1878.41	7535	1658.28
tai20b	166	196.56	220	124.72	557	109.61	7457	315.86
tai25b	659	1689.15	815	1794.82	1215	1535.79	15367	4029.07

Table: computational results of Self-splitting ParaQapNB on medium instances with solver=5

*depth: sampling tree depth at Self-splitting

Computational Results on Large Instances

problem	Opt.val	#node	time(sec)	No. of CPU cores used
nug30	6,124	26,181	3.14e3	1,728
tai30a	1,818,146	34,000,579	5.81e5 $\approx$ 6.8 days	1,728
tai35b	283,315,445	2,620,547	2.49e5	1,728
tai40b	637,250,948	278,465	1.05e5	1,728
sko42	15,812	6,019,419	5.12e5 $\approx$ 5.9 days	5,184

Table: Computational results on large scale QAPs

- HPE SGI 8600 (384 nodes, 13,824 cores, 144TB) @ ISM
- Adopt **average branching** as branching strategy
- Normal ramp-up

Computational Results on large instances

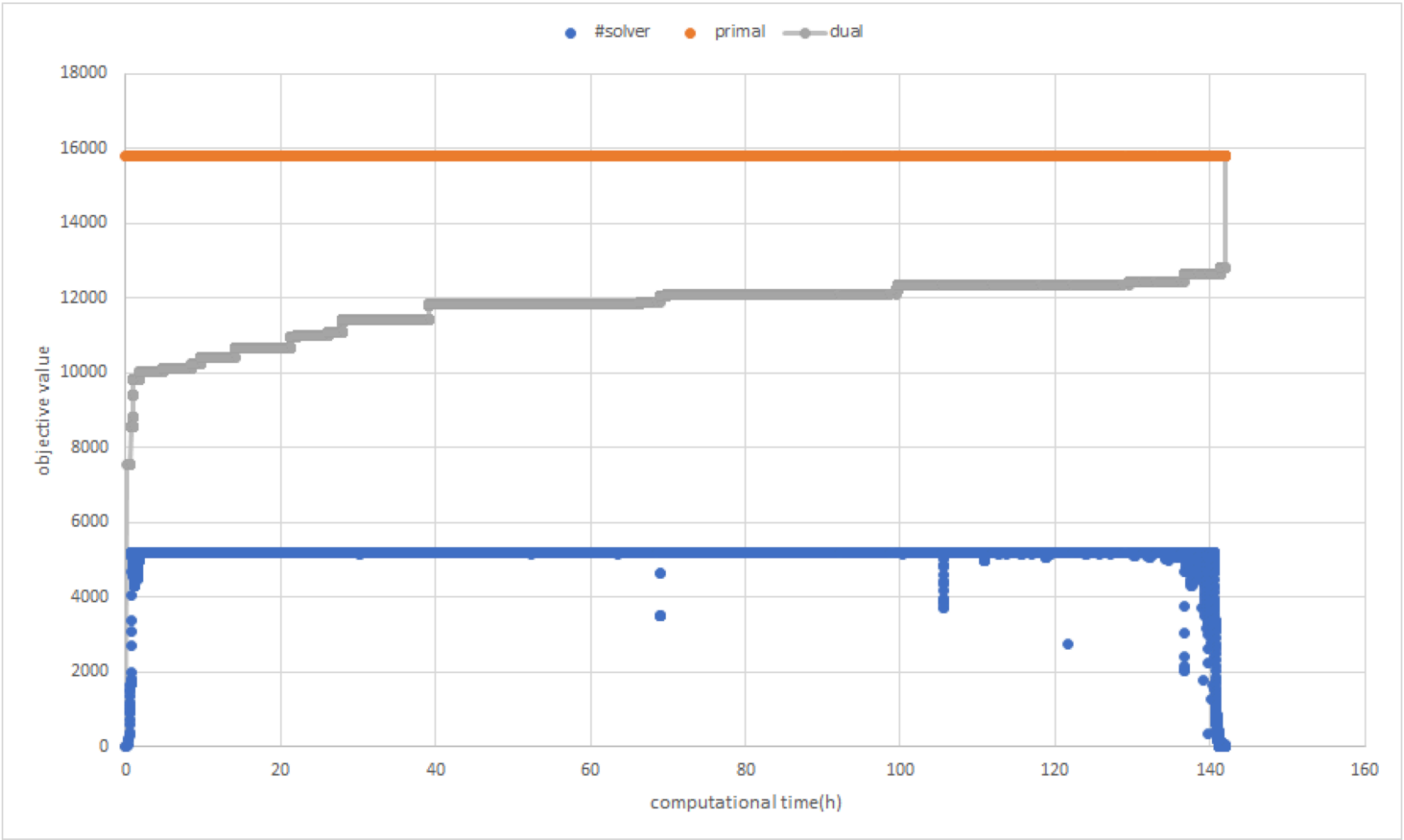


Figure: ParaQapNB computation log of sko42

Summary

- Newton-bracketing method for Lagrangian DNN relaxation of QAP
- QapNB: DNN-based Branch-and-bound solver for QAP
  (1) Incument solution: robust tabu search
  (2) Stop condition of Newton-bracketing method
  (3) Branching strategy: average branching and primal DNN branching
- ParaQapNB: massively parallel DNN-based branch-and-bound solver for QAP
  - Solved open instances tai30a and sko42.

Fujii, K., Ito, N., Kim, S., Kojima, M., Shinano, Y., & Toh, K. C. (2021). Solving Challenging Large Scale QAPs. arXiv preprint arXiv:2101.09629.

Conguratulations on UG ver1.0!

- ParaQapNB solved nug24 in 3219.32(s) with 6 processes

Table 4. Computational results for QAPLIB instances

Prob.	Time(sec)	# of subproblems	Improvements
nug21	13287	593656913	none
nug22	147378	6712276783	none
nug24	1269218	44317904109	none

Yuji Shinano, Tetsuya Fujie (1999). Parallel Branch-and-Bound Algorithms on a PC Cluster using PUBB.