An Introduction to Fair Combinatorial Optimization

Viet Hung Nguyen[1]

[1] Clermont-Auvergne University, LIMOS lab and LIP6 lab, Clermont-Ferrand and Paris, France

Ho Chi Minh City, Dec. 5, 2023

Overview

1. Background and Motivation of Generalized Gini Index
2. OWA Combinatorial Optimization
3. Nash Proportional Fairness
4. Experimental Results

Classic Combinatorial Optimization Problems

- Let $[n] = 1, \ldots, n$ be a set of n entities and let $[m] = 1, \ldots, m$ be another set of m entities.
- We focus on a class of combinatorial optimization problems:

$$\begin{aligned} \max/\min. \quad & \sum_{i \in [n]} \sum_{j \in [m]} u_{ij} z_{ij} \\ \text{s.t.} \quad & \mathbf{Az} \leq \mathbf{b} \\ & \mathbf{z} \in \{0,1\}^{n \times m} \end{aligned}$$

- The binary variable $z_{ij} = 1$ could signify the "relation" ij is chosen in the solution. $z_{ij} = 0$ otherwise.
- "Relation" ij could be a network link between two node i and j or the assignment of j to i, etc.
- Here, the data u_{ij} could be interpreted as the utility/cost of the relation between i and j. The objective is to maximize/minimize the total utility/cost.
- $0/1$ vectors $z\{0,1\}^{n \times m}$ is a solution if z satisfy $\mathbf{Az} \leq \mathbf{b}$ and vice-versa.

Classical examples of combinatorial optimization problems

- Graph and Network Optimization Problems such as Minimum Spanning Tree, Shortest Path, Maximum Flow, Minimum Cut, Min-Cost Flow, Traveling Salesman Problem, Maximum Cut, Vehicle Routing, ...
- Packing and Covering: Matching, Dominating set, Set Cover, Vertex Cover, Bin Packing, Graph Partitioning et Clustering ...,
- Coloring problems: Clique, Stable, ...

Approaches for solving combinatorial optimization problems

Many "classical" approaches allowing to solve exactly or approximately combinatorial optimization problems such as

- combinatorial algorithms: greedy algorithms, divide-and-conquer, dynamic programming, primal-dual algorithms, branch-and-bound algorithms...
- mathematical programming based algorithms: cutting-plane algorithms, branch and cut algorithms.
- heuristic and meta-heuristic: neighbourhood search algorithms, genetic algorithms, ...

Fair Combinatorial Optimization Problems

- Let us call *agents* the elements in $[n]$ and we want to find fair solutions for every agent in term of utility or cost.
- $v_i = \sum_{j \in [m]} u_{ij} z_{ij}$ represents i-th agent's utility/cost.
- **Fairness:** Let us call $v = (v_1, \ldots, v_n)^t$ the utility/cost vector then in fair COP, we want solutions with balanced vector v with respect to its components.
- **Efficiency:** The solutions should be also efficient in terme of total utility/cost.

A fair sharing problem

- Let A, B and C be three persons who have to share six objects of value 325, 225, 210, 115, 75 and 50 euros.
- The total value of the object is 1000 euros.
- The objects are indivisible $\Rightarrow$ an equal share is not possible.
- We need to a provide a solution as fair as possible and we should be able to justify the fairness of the solution.
- This example is inspired from the article about Fair Optimization written by P. Perny in OR special issue of the magazine Tangente.

A greedy algorithm

Recall : the persons A, B and C and six objects 1, 2, . . . , 6 of value 325, 225, 210, 115, 75 and 50 euros respectively.

A greedy algorithm : Start from giving 0 object for all the persons. Take the objects one by one by the decreasing order of value and give the next object to the person having the least.

Execution

- At some stage of the algorithm, we have $A = \{325\}$, $B = \{225\}$ and $C = \{210\}$.
- Then $A = \{325\}$, $B = \{225\}$ and $C = \{210, 115\}$ and,
- $A = \{325\}$, $B = \{225, 75\}$ and $C = \{210, 115\}$ and
- finally $A = \{325\}$ of value 325, $B = \{225, 75, 50\}$ of value 350 and $C = \{210, 115\}$ of value 325.

Is the solution found by this greedy algorithm is a "fair" solution?

Balance optimization and Mixed Integer Program (MIP)

Variables:

- $$x_{ij} = \begin{cases} 1 & \text{if object } j \text{ is assigned to person } i, \\ 0 & \text{otherwise} \end{cases}$$

  for all $i = 1, \ldots, 3$ and $j = 1, \ldots, 6$.
- P a variable denoting the biggest value assigned to a person.
- Q a variable denoting the smallest value assigned to a person.

Objective (balance optimization): Minimizing $P - Q$, the gap between the richest person and the poorest one.

Balance optimization and Mixed Integer Program (MIP)

Constraints:

- Each object should be assigned to exactly one person:

$$\sum_{i=1}^{3} x_{ij} = 1 \text{ for all } j = 1, \ldots, 6.$$

- Bound constraint for P:

$$P \geq \sum_{j=1}^{6} x_{ij} \text{ for all } i = 1, \ldots, 3.$$

- Bound constraint for Q:

$$Q \leq \sum_{j=1}^{6} x_{ij} \text{ for all } i = 1, \ldots, 3.$$

- Bound (and binary) constraints for x_{ij}:

$$x_{ij} \in \{0, 1\} \text{ for } i = 1, \ldots, 3 \text{ and } j = 1, \ldots, 6.$$

The whole MIP for balance optimization

$$
\begin{array}{lll}
\text{minimize} & P - Q & \\
\text{subject to} & \sum_{i=1}^{3} x_{ij} = 1 & \text{for all } j = 1, \ldots, 6 \\
& P \geq \sum_{j=1}^{6} x_{ij} & \text{for all } i = 1, \ldots, 3 \\
& Q \leq \sum_{j=1}^{6} x_{ij} & \text{for all } i = 1, \ldots, 3 \\
& x_{ij} \in \{0, 1\} & \text{for } i = 1, \ldots, 3 \text{ and } j = 1, \ldots, 6.
\end{array}
$$

- One of the bound constraints for P (resp. for Q) should be saturated as we minimize (resp. maximize) P (resp. Q).
- Solving the MIP by a MIP solver (Cplex, Gurobi, Scip, Clp, ...), we obtain an optimal solution : $A = \{325\}$ of value 325, $B = \{225, 115\}$ of value 340 and $C = \{210, 75, 50\}$ of value 335.

Pigou-Dalton transfer : a test for fairness

- **The "greedy" solution:** $A = \{325\}$ of value 325, $B = \{225, 75, 50\}$ of value 350 and $C = \{210, 115\}$ of value 325.
- **The balance solution:** $A = \{325\}$ of value 325, $B = \{225, 115\}$ of value 340 and $C = \{210, 75, 50\}$ of value 335.
- The balance solution is apparently better in term of fairness but if we have in our possession only the "greedy" solution, how can we know that the latter is not optimal?
- **Pigou-Dalton transfer:** a transfer of value $\epsilon > 0$ from a richer agent to a poorer agent without changing their rank.
- We can realize a Pigou-Dalton transfer on the "greedy" solution by transferring the objects 5 and 6 of values 75 and 50 from B to C and transferring back from C to B the object 4 of value 115 to B.
- The new solution is the balance solution.

When efficiency is also necessary

- In the previous example, we neglected efficiency because it is not necessary: every object is assigned to one person and the total utility is always 1000 in every solution.
- Let us consider an additional constraint to the problem: Because of taxes, the total value of assigned objects should not exceed 900 euros.
- Hence, every object is not necessary assigned to one person.
- The fair solution should be also efficient, i.e. having a maximal total utility.

A modified greedy algorithm

Recall : the persons A, B and C and six objects 1, 2, . . . , 6 of value 325, 225, 210, 115, 75 and 50 euros respectively.

A modified greedy algorithm : Remove the objects one by one by the increasing order of value until the total value goes under 900.

Start from giving 0 object for all the persons. Take the objects one by one by the decreasing order of value and give the next object to the person having the least.

Execution

- Removing the objects 50 then 75.
- Then the final solution is $A = \{325\}$ of value 325, $B = \{225\}$ of value 225 and $C = \{210, 115\}$ of value 325.
- The total value of the greedy solution is 875.

Is the solution found by this modified greedy algorithm is optimal?

Combination of Pareto dominance and Pigou-Dalton transfer

- For two utility vectors $v = (v_1, \ldots, v_n)^t$ and $v' = (v'_1, \ldots, v'_n)^t$, v Pareto dominates v' if $v_i \geq v'_i$ for all $i \in [n]$.
- v is Pareto optimal if v is not Pareto dominated by any other utility vector v'.
- Let us consider the modified greedy solution $A = \{325\}$ of value 325, $B = \{225\}$ of value 225 and $C = \{210, 115\}$ of value 325. The utility vector $v = (325, 225, 325)^t$.
- By a Pigou-Dalton transfer of 40 from C to B, we obtain a pseudo-solution $v_1 = (325, 265, 285)^t$.
- v_1 is Pareto dominated by $v_2 = (325, 275, 285)$ which correspond to a real solution $A = \{325\}$, $B = \{225, 50\}$ and $C = \{210, 75\}$.
- By transitivity, solution v_2 is better than the solution proposed by the modified greedy algorithm.

Fair Combinatorial Optimization Problems

- We focus on a class of combinatorial optimization problems:

$$\begin{aligned} \text{max. } & f\big((\sum_{j\in[m]} u_{ij}z_{ij})_{i\in[n]}\big) \\ \text{s.t. } & \mathbf{Az} \leq \mathbf{b} \\ & \mathbf{z} \in \{0,1\}^{n\times m} \end{aligned}$$

- For maximum weight problems, $f(\boldsymbol{v}) = \sum_{i\in[n]} v_i$
- Here, $\sum_{j\in[m]} u_{ij}z_{ij}$ represents i-th criterion or i-th agent's utility
- **Example:**

Fair Combinatorial Optimization Problems

- We focus on a class of combinatorial optimization problems:

$$\begin{aligned} \text{max. } & f\big((\sum_{j\in[m]} u_{ij}z_{ij})_{i\in[n]}\big) \\ \text{s.t. } & \mathbf{Az} \leq \mathbf{b} \\ & \mathbf{z} \in \{0,1\}^{n\times m} \end{aligned}$$

- For maximum weight problems, $f(\boldsymbol{v}) = \sum_{i\in[n]} v_i$
- Here, $\sum_{j\in[m]} u_{ij}z_{ij}$ represents i-th criterion or i-th agent's utility
- **Example:** Assignment (m tasks to be assigned to n agents)
  $\sum_{j\in[m]} u_{ij}z_{ij}$ = utility of i-th agent after receiving some tasks

Fair Combinatorial Optimization Problems

- We focus on a class of combinatorial optimization problems:

$$\begin{aligned} \text{max. } & f\big((\sum_{j\in[m]} u_{ij}z_{ij})_{i\in[n]}\big) \\ \text{s.t. } & \mathbf{Az} \leq \mathbf{b} \\ & \mathbf{z} \in \{0,1\}^{n\times m} \end{aligned}$$

- For maximum weight problems, $f(\boldsymbol{v}) = \sum_{i\in[n]} v_i$
- Here, $\sum_{j\in[m]} u_{ij}z_{ij}$ represents i-th criterion or i-th agent's utility
- **Example:** Perfect matching (on complete graph with $n = m$ nodes) $\sum_{j\in[m]} u_{ij}z_{ij}$ = utility of the i-th agent's pair

Fair Combinatorial Optimization Problems

- We focus on a class of combinatorial optimization problems:

$$\begin{aligned} \text{max. } & f\big((\sum_{j\in[m]} u_{ij}z_{ij})_{i\in[n]}\big) \\ \text{s.t. } & \mathbf{Az} \leq \mathbf{b} \\ & \mathbf{z} \in \{0,1\}^{n\times m} \end{aligned}$$

- For maximum weight problems, $f(\boldsymbol{v}) = \sum_{i\in[n]} v_i$
- Here, $\sum_{j\in[m]} u_{ij}z_{ij}$ represents i-th criterion or i-th agent's utility
- **Example:** TSP (with $n = m$ cities)
  $\sum_{j\in[m]} u_{ij}z_{ij} \sim$ time to arrive and leave the i-th city

Fair Combinatorial Optimization Problems

- We focus on a class of combinatorial optimization problems:

$$\begin{aligned} \text{max. } & f\big((\sum_{j\in[m]} u_{ij}z_{ij})_{i\in[n]}\big) \\ \text{s.t. } & \mathbf{Az} \leq \mathbf{b} \\ & \mathbf{z} \in \{0,1\}^{n\times m} \end{aligned}$$

- For maximum weight problems, $f(\boldsymbol{v}) = \sum_{i\in[n]} v_i$
- Here, $\sum_{j\in[m]} u_{ij}z_{ij}$ represents i-th criterion or i-th agent's utility
- **Example:** TSP (with $n = m$ cities)
  $\sum_{j\in[m]} u_{ij}z_{ij} \sim$ time to arrive and leave the i-th city
- **Issue:** maximum weight formulation does not offer any control on the distribution of the utilities over components i

Different Formulations to Encode Fairness

Some classic aggregating functions f to represent fairness:

- maxmin: $f(\boldsymbol{v}) = \min_{i \in [n]} v_i$
  Issue: $(0, \ldots, 0)$ preferred to $(0, 100, \ldots, 100)$
- leximin: lexicographic min
  Issue: $(1, \ldots, 1)$ preferred to $(0, 100, \ldots, 100)$

But, what properties should f satisfy?

- Symmetric: $f(\boldsymbol{v}) = f(\boldsymbol{v}_\sigma)$ for any permutation σ
  No agent/criterion is favored
- Strictly increasing: $\boldsymbol{v} \succ_P \boldsymbol{w} \Rightarrow f(\boldsymbol{v}) > f(\boldsymbol{w})$
  Compatible with Pareto dominance $\succ_P$
- Strictly Schur-concave: strictly monotonic with Pigou-Dalton transfers
  Compatible with Lorenz dominance: balanced distribution is preferred

Pigou-Dalton transfers and Lorenz Dominance

Fairness and Pigou-Dalton transfers

Let $\boldsymbol{v} \in \mathbb{R}^n_+$ such that $v_i < v_k$. Then for all $\varepsilon \in (0, v_k - v_i)$,
$(v_1, \ldots, v_i, \ldots, v_k, \ldots, v_n) \prec (v_1, \ldots, v_i + \varepsilon, \ldots, v_k - \varepsilon, \ldots, v_n)$

Example: $(18, 11, 12) \prec_{PD} (\mathbf{14}, \mathbf{15}, 12) \prec_{PD} (14, \mathbf{13}, \mathbf{14}) \prec_{P} (14, \mathbf{14}, 14)$

Pigou-Dalton transfers and Lorenz Dominance

Fairness and Pigou-Dalton transfers

Let $\boldsymbol{v} \in \mathbb{R}^n_+$ such that $v_i < v_k$. Then for all $\varepsilon \in (0, v_k - v_i)$,
$(v_1, \ldots, v_i, \ldots, v_k, \ldots, v_n) \prec (v_1, \ldots, v_i + \varepsilon, \ldots, v_k - \varepsilon, \ldots, v_n)$

Example: $(18, 11, 12) \prec_{PD} (\mathbf{14}, \mathbf{15}, 12) \prec_{PD} (14, \mathbf{13}, \mathbf{14}) \prec_{P} (14, \mathbf{14}, 14)$

Theorem [Hardy, Littlewood and Polya 34, Chong 76]

There exists an improving sequence from $\boldsymbol{v}$ to $\boldsymbol{u}$ made of Pigou-Dalton transfers and Pareto-improvements iff $\boldsymbol{u} \succ_L \boldsymbol{v}$

Pigou-Dalton transfers and Lorenz Dominance

Fairness and Pigou-Dalton transfers

Let $\boldsymbol{v} \in \mathbb{R}^n_+$ such that $v_i < v_k$. Then for all $\varepsilon \in (0, v_k - v_i)$,
$(v_1, \ldots, v_i, \ldots, v_k, \ldots, v_n) \prec (v_1, \ldots, v_i + \varepsilon, \ldots, v_k - \varepsilon, \ldots, v_n)$

Example: $(18, 11, 12) \prec_{PD} (\mathbf{14}, \mathbf{15}, 12) \prec_{PD} (14, \mathbf{13}, \mathbf{14}) \prec_P (14, \mathbf{14}, 14)$

Theorem [Hardy, Littlewood and Polya 34, Chong 76]

There exists an improving sequence from $\boldsymbol{v}$ to $\boldsymbol{u}$ made of Pigou-Dalton transfers and Pareto-improvements iff $\boldsymbol{u} \succ_L \boldsymbol{v}$

(Generalized) Lorenz Dominance [Shorrocks, 83]

$\boldsymbol{u} \succ_L \boldsymbol{v}$ iff $L(\boldsymbol{u}) \succ_P L(\boldsymbol{v})$ where $L(\boldsymbol{v}) = (L_1(\boldsymbol{v}), \ldots, L_n(\boldsymbol{v}))$, $L_k(\boldsymbol{v}) = \sum_{i=1}^{k} v_i^{\uparrow}$

Example:
$\boldsymbol{v} = (18, 11, 12) \quad L(\boldsymbol{v}) = (11, 23, 41)$
$\boldsymbol{u} = (14, 14, 14) \quad L(\boldsymbol{u}) = (14, 28, 42)$

Generalized Gini Index

Generalized Gini Index (GGI) [Weymark, 81]

$$G_{\boldsymbol{w}}(\boldsymbol{v}) = \sum_i w_i v_i^{\uparrow}$$
$$= \sum_i (w_i - w_{i+1}) L_i(\boldsymbol{v})$$

where $v_1^{\uparrow} \leq \ldots \leq v_n^{\uparrow}$ and $w_1 > w_2 > \ldots > w_n > w_{n+1} = 0$.

- GGI is a well-known inequality measure in economics
- GGI is symmetric, strictly increasing and strictly Schur-concave
- GGI is a weighted sum on Lorenz components (with positive weights)
- Classic Gini index is a special case of GGI

OWA Combinatorial Optimization

Our problem is formulated as follows:

$$\max. \; G_{\mathbf{w}}\Big(\big(\sum_{j\in[m]} u_{ij}z_{ij}\big)_{i\in[n]}\Big) \quad \text{s.t.} \; \mathbf{Az} \leq \mathbf{b}, \; \mathbf{z} \in \{0,1\}^{n\times m}$$

$$\Leftrightarrow$$

$$\max. \; \sum_i w'_k L_k(v) \quad \text{s.t.} \; \mathbf{Az} \leq \mathbf{b}, \; \mathbf{z} \in \{0,1\}^{n\times m}$$

where $w'_k = w_k - w_{k+1}$ and $v = (v_1, \ldots, v_n)$ with $v_i = \sum_{j\in[m]} u_{ij}z_{ij}$.

- non-linear combinatorial optimization problem
- but GGI can be linearized by introducing new variables

Linearization of Ogryczak and Sliwinski (2003)

Primal for Lorenz component L_k

$$\begin{aligned} \text{min.} \quad & \sum_{i\in[n]} a_{ik} v_i & \\ \text{s.t.} \quad & \sum_{i\in[n]} a_{ik} = k & \\ & 0 \leq a_{ik} \leq 1 & \forall i \in [n] \end{aligned}$$

Dual for Lorenz component L_k

$$\begin{aligned} \text{max.} \quad & kr_k - \sum_{i\in[n]} d_{ik} & \\ \text{s.t.} \quad & r_k - d_{ik} \leq v_i & \forall i \in [n] \\ & d_{ik} \geq 0 & \forall i \in [n] \end{aligned}$$

Linearization Ogryczak and Sliwinski (2003)

Dual for Lorenz component L_k

$$\begin{aligned} \text{max.} \quad & kr_k - \sum_{i\in[n]} d_{ik} & \\ \text{s.t.} \quad & r_k - d_{ik} \leq v_i & \forall i \in [n] \\ & d_{ik} \geq 0 & \forall i \in [n] \end{aligned}$$

Our problem can then be reformulated as a MIP:

(O-MIP)

$$\begin{aligned} \text{max.} \quad & \sum_{k\in[n]} w'_k (kr_k - \sum_{i\in[n]} d_{ik}) & \\ \text{s.t.} \quad & \boldsymbol{Az} \leq \boldsymbol{b} & \\ & \boldsymbol{z} \in \{0,1\}^{n\times m} & \\ & r_k - d_{ik} \leq \sum_{j\in[m]} u_{ij} z_{ij} & \forall i \in [n], \forall k \in [n] \\ & d_{ik} \geq 0 & \forall i \in [n], \forall k \in [n] \end{aligned}$$

Linearizations of Chassein and Goerigk (2015)

OWA can be represent as

$$\mathrm{OWA}_{\boldsymbol{w}}(\boldsymbol{v}) = \min_{\tau \in \Pi} \sum_{i \in [n]} w_{\tau(i)} v_i$$

where Π denotes the set of all permutations τ of $[n]$.

LPs for OWA

$$\begin{aligned}
\min \quad & \sum_{i \in [n]} \sum_{j \in [n]} w_j v_i p_{ij} \\
\text{s.t} \quad & \sum_{i \in [n]} p_{ij} = 1 \; \forall j \in [n] \\
& \sum_{j \in [n]} p_{ij} = 1 \; \forall i \in [n] \\
& p \in \mathbb{R}_+^{n \times n}.
\end{aligned}
\quad \Leftrightarrow \quad
\begin{aligned}
\max \quad & \sum_{i \in [n]} (\alpha_i + \beta_i) \\
\text{s.t} \quad & \alpha_i + \beta_j \leq w_j v_i \qquad \forall i, j \in [n] \\
& \alpha_i, \beta_i \in \mathbb{R} \qquad \forall i \in [n].
\end{aligned}$$

Linearizations of Chassein and Goerigk (2015)

$$
(C-MIP)\quad
\begin{aligned}
\max\ & \sum_{i\in[n]}(\alpha_i+\beta_i) \\
\text{s.t } & \alpha_i+\beta_k \le w_i v_k \quad \forall i,k\in[n] \\
& \boldsymbol{v}=\boldsymbol{Cx} \\
& \boldsymbol{Ax}\le \boldsymbol{b} \\
& \boldsymbol{x}\in\{0,1\}^m
\end{aligned}
$$

Fair Optimization Problem as a Bi-objective Optimization Problem

- An alternative to OWA is to use Bi-objective Optimization where the first objective is maximizing the total utility $M = \sum_{i \in [n]} \sum_{j \in [m]} u_{ij} z_{ij}$ to ensure the efficiency of the solution and the second objective is minimizing the balance $M = P - Q$ where P is greatest agent's utility and Q is the smallest agent's utility.
- We aim at finding Pareto optimal solution achieving a Nash Proportional equilibrium between the two objectives M and N. Litteraly, a solution (M^*, N^*) is Nash Proportional equilibrium if an improvement of p percent of the objective M implies necessarily a degradation of at least p percent of the objective N and vice-versa.
- Mathematically,

Comparisons between C-MIP and O-MIP

O-MIP Ogryczak and Sliwinski (2003)	C-MIP Chassein and Goerigk (2015)
$\max \sum_{k\in[n]} w'_k(kr_k - \sum_{i\in[n]} d_{ki})$	$\max \sum_{i\in[n]} (\alpha_i + \beta_i)$
s.t $r_k + d_{ki} \geq v_i \quad \forall k, i \in [n]$	s.t $\alpha_i + \beta_k \leq w_i v_k \quad \forall i, k \in [n]$
$d_{ki} \geq 0 \quad \forall k, i \in [n]$	$\boldsymbol{v} = \boldsymbol{Cx}$
$\boldsymbol{v} = \boldsymbol{Cx}$	$\boldsymbol{Ax} \leq \boldsymbol{b}$
$\boldsymbol{Ax} \leq \boldsymbol{b}$	$\boldsymbol{x} \in \{0,1\}^m$
$\boldsymbol{x} \in \{0,1\}^m$	
• $n^2 + n$ additional variables	• $2n$ additional variables
• n^2 new constraints	• n^2 new constraints

▶ C-MIP is reported to be more efficient than that of O-MIP for some problems like OWA simplified portfolio optimization problem.

Comparisons of C-MIP with O-MIP

Theorem

C-MIP and O-MIP are equivalent in terms of linear relaxations.

Comparisons of C-MIP with O-MIP

Theorem

C-MIP and O-MIP are equivalent in terms of linear relaxations.

Instance	O-MIP	C-MIP
gr17	1.68	1.28
gr21	1.46	1.08
fri26	21.01	49.97
bayg29	31.06	35.86
bays29	37.89	57.85
swiss42	354.47	915.22
gr48	863.02	2419.39
hk48	1943.39	719.89
berlin52	> 10800	> 10800
brazil58	> 10800	> 10800

Table: Numerical results of O-MIP and C-MIP in small-size fair TSP instances.

Linear Programming relaxation of (O-MIP) and its dual

$$\max. \sum_{k\in[n]} w'_k(kr_k - \sum_{i\in[n]} d_{ik})$$

$$\text{s.t. } \boldsymbol{Az} \leq \boldsymbol{b}$$

$$\boldsymbol{z} \in [0,1]^{n\times m}$$

$$r_k - d_{ik} \leq \sum_{j\in[m]} u_{ij}z_{ij}$$

$$d_{ik} \geq 0$$

$$\min. \mathbf{b}^\intercal\mathbf{v} + \sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij}$$

$$\text{s.t. } (\mathbf{v}^\intercal\mathbf{A})_{ij} + t_{ij} - \sum_{k=1}^{n} u_{ij}y_{ik} \geq 0$$

$$\sum_{i=1}^{n} y_{ik} = kw'_k$$

$$y_{ik} \leq w'_k$$

$$v_j \geq 0$$

$$y_{ik} \geq 0$$

$$t_{ij} \geq 0$$

Dual's interpretation

Variables y_{ik}'s are likely to approximate the Lorenz vector for solutions z.

Partial dual with fixed y

$$\text{min. } \mathbf{b}^\intercal \mathbf{v} + \sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij}$$

$$\text{s.t. } (\mathbf{v}^\intercal \mathbf{A})_{ij} + t_{ij} \geq \sum_{k=1}^{n} u_{ij} y_{ik}$$

$$v_j \geq 0, t_{ij} \geq 0$$

Dual of the partial dual (DPD)

$$\text{max. } \sum_{i=1}^{n} (\sum_{k=1}^{n} y_{ik}) \sum_{j=1}^{m} u_{ij} z_{ij}$$

$$\text{s.t. } \mathbf{A}\mathbf{z} \leq \mathbf{b}$$

$$\mathbf{z} \in [0,1]^{n \times m}$$

Theorem

Given a dual feasible solution $y*$ and z^* is an optimal solution of 0/1-(DPD) with y^*, if y^* induces the Lorenz vector corresponding to z^* then z^* is optimal for (Fair-MIP).

Performance guarantee for particular solutions

Theorem (N., Weng 2017)

Let $\bar{\boldsymbol{z}}$ be an optimal solution of the maximum weight version. Let $\bar{T}_i = \sum_{j\in[m]} u_{ij}\bar{z}_{ij}$ for all $i \in [n]$ and assume without loss of generality that $\bar{T}_1 \geq \bar{T}_2 \geq \ldots \geq \bar{T}_n$. Let $w'_{\max} = \max_{k\in[n]} w'_k$. Then the GGI value of $\bar{\boldsymbol{z}}$ is at worst $\max(\frac{2w'_n}{(n+1)w'_{\max}}, \frac{n\bar{T}_n}{(\sum_{i\in[n]}\bar{T}_i)})$ of the optimal objective value of (Fair-MIP).

Proof.

The proof is done by observing that $\bar{z}$ is optimal for 0/1-(DPD) with fixed $\bar{\boldsymbol{y}}$ defined as $\bar{y}_{ik} = \frac{k}{n}w'_k$ for $i, k \in [n]$. □

0/1-(DPD) as Lagrangian relaxation

(Fair-MIP)

$$\begin{aligned}
\text{max.} \quad & \sum_{k\in[n]} w'_k\Big(kr_k - \sum_{i\in[n]} d_{ik}\Big) \\
\text{s.t.} \quad & \boldsymbol{Az} \le \boldsymbol{b} \\
& \boldsymbol{z} \in \{0,1\}^{n\times m} \\
& r_k - d_{ik} \le \sum_{j\in[m]} u_{ij} z_{ij} \\
& d_{ik} \ge 0
\end{aligned}$$

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\lambda}) = \quad \text{max.} \quad & \sum_{k\in[n]} \Big(w'_k k - \sum_{i\in[n]} \lambda_{ik}\Big) r_k \\
& - \sum_{k\in[n]} \sum_{i\in[n]} (w'_k - \lambda_{ik}) d_{ik} \\
& + \sum_{i\in[n]} \Big(\sum_{k\in[n]} \lambda_{ik}\Big) \sum_{j\in[m]} u_{ij} z_{ij} \\
\text{s.t.} \quad & \boldsymbol{Az} \le \boldsymbol{b} \\
& \boldsymbol{z} \in \{0,1\}^{n\times m} \\
& d_{ik} \ge 0
\end{aligned}$$

$\mathcal{L}(\boldsymbol{\lambda})$ gives a finite upper bound for (MIP-LP) if $\sum_{i=1}^{n} \lambda_{ik} = kw'_k$ and $0 \le \lambda_{ik} \le w'_k$. These are exactly the constraints for y_{ik}'s.

We found a way to iteratively update y_{ik} and solve 0/1-(DPD)!

Primal-Dual Algorithm (N. and Weng 2017)

1. $t \leftarrow 0$
2. compute $\boldsymbol{y}^{(0)}$ corresponding to the maximum weight solution z^0.
3. Repeat
   - $t \leftarrow t + 1$
   - solve the (0/1-DPD) with y^{t-1} to obtain feasible solution $\boldsymbol{z}^{(t)}$
   - update $\boldsymbol{y}^{(t)}$ by subgradient algorithm for solving the Lagrangian dual.
4. Until max iteration has been reached or change on $y_{ik}^{(t)}$ is small
5. return $\boldsymbol{z}^{(t)}$ with highest GGI

Experimental Results for Fair Assignment Problem

Instance	CPLEX		HO	
	CPU1	CPU2	CPU	Gap
v50-20	1.02	1.02	0.23	0%
v50-30	3.14	3.14	0.26	0%
v50-40	64.95	14.26	0.45	0.28%
v50-50	1054.14	100.23	0.65	0.26%
v30-20	0.89	0.89	0.2	0%
v30-30	8.83	8.83	0.3	0.015%
v30-40	590.66	45.93	0.48	0.13%
v10-20	1.55	1.55	0.18	0%
v10-30	342.78	342.78	0.94	0%

Table: Numerical results for fair assignment problems.

Experimental Results for General Fair Matching Problems

Instance	CPLEX		HO	
	CPU1	CPU2	CPU	Gap
v50-30	0.86	0.86	0.79	0%
v50-40	2.43	2.43	1.42	0%
v50-50	5.14	5.14	2.67	0%
v50-60	148.5	25.45	13.43	0.01%
v50-70	2406.02	1282.8	17.71	0.005%
v30-30	1.15	1.15	0.78	0%
v30-40	7.13	7.13	1.44	0%
v30-50	81.75	75.5	2.45	0.01%
v30-60	1003.69	615.16	12.8	0.036%
v10-30	5.33	5.33	0.76	0%
v10-40	1325.7	806.8	1.4	0.06%
v10-50	29617.78	3370.7	2.48	0.053%

Table: Numerical results for general matching problems

Comparisons of C-MIP with O-MIP

Instance	O-MIP	C-MIP	$\mathcal{H}_O$		$\mathcal{H}_C$	
			CPU	**Gap**	**CPU**	**Gap**
burma14	0.83	**0.45**	1.50	0.23%	1.62	0.06%
gr17	1.68	**1.28**	1.97	1.15%	2.23	0.23%
gr21	1.46	**1.08**	1.82	0.02%	2.25	0.00%
gr24	8.47	8.46	**3.59**	0.00%	4.09	0.00%
fri26	21.01	49.97	5.74	0.00%	**4.12**	0.02%
bayg29	31.06	35.86	9.31	2.51%	**8.38**	1.11%
bays29	37.89	57.85	**9.92**	0.86%	11.10	1.19%
swiss42	354.47	915.22	**16.86**	0.51%	20.77	0.50%
att48	731.25	655.06	**45.16**	1.77%	97.09	1.66%
gr48	863.02	2419.39	57.17	2.86%	**56.19**	1.75%
hk48	1943.39	719.89	32.21	0.03%	**31.14**	0.48%
Average	363.14	442.23	**16.84**	0.90%	21.73	0.64%

Table: Numerical results of all methods in small-size instances. [1]

[1] $\mathcal{H}_O, \mathcal{H}_C$ are respectively Primal-Dual algorithms utilizing O-MIP, C-MIP

Comparisons of C-MIP with O-MIP

Instance	$\mathcal{H}_O$		$\mathcal{H}_C$	
	CPU	**ObjVal**	**CPU**	**ObjVal**
berlin52	18.01	691.23	**17.13**	689.13
brazil58	**21.85**	4770.24	21.88	4772.03
st70	207.48	35.89	**169.534**	35.97
eil76	217.55	24.46	**210.86**	24.55
pr76	3268.48	6808.09	**2687.25**	6808.09
rat99	**833.34**	34.02	1285.8	31.99
kroA100	**1202.54**	811.42	1657.44	790.76
kroB100	**1807.06**	821.23	5464.95	930.65
Average	**947.04**	1749.57	1439.36	1760.40

Table: Numerical results of heuristic in larger instances which MIPs can not be solved exactly within **three hours (**$>$ 10800**s)**.