Singleton Node Consistency for Quadratic Assignment Problems in Cost Function Networks

Guidio Sewa[1,2][0009−0003−7869−0793], David Allouche[1][0000−0003−3549−4592], Simon de Givry[1,2][0000−0002−2242−0458], George Katsirelos[2,3][0000−0002−3727−6698], and Thomas Schiex[1,2][0000−0001−6049−3415]

[1] Université de Toulouse, MIAT, INRAE, France
[2] Université de Toulouse, ANITI, France
{guidio.sewa,david.allouche,simon.de-givry,thomas.schiex}@inrae.fr
[3] MIA Paris, INRAE, France gkatsi@gmail.com

Abstract. The Quadratic Assignment Problem (QAP) consists of finding a permutation that minimizes a quadratic objective function. Exact methods generally rely on a branch-and-bound procedure, the efficiency of which depends heavily on the quality of its lower bound. In integer linear programming, several bounds have been investigated, exhibiting different trade-offs between speed and quality. The Gilmore-Lawler bound appears to be the most commonly used in practice. It requires solving a linear assignment problem (LAP) for each variable-value pair. We show how to obtain this bound using Singleton Node Consistency (SNC) and LAP. In Cost Function Networks (CFNs), we propose a reformulation that transforms the result of applying LAP to a given variable-value pair into cost functions of arity 1 and 2, which can be added to the original problem. Combined with existing lower bounds for CFNs, including EDAC and a recent CFN propagator for AllDifferent, this method (SNC-LAP-GLB), used as a preprocessing, significantly increases the initial lower bound and accelerates the search, resulting in competitive results on the QAPLIB benchmark. We then propose an extension of the AllDifferent propagator for the Global Cardinality Constraint. It allows us to exploit variable symmetries on some challenging QAPLIB instances, thus improving the results.

Keywords: Linear Assignment Problem · AllDifferent· Global Cardinality Constraint.

1 Introduction

The quadratic assignment problem (QAP) was introduced in [26] as a mathematical model for the location of indivisible economical activities. *"We want to assign n facilities to n locations with the cost being proportional to the flow between the facilities multiplied by the distances between the locations plus, eventually, costs for placing the facilities at their respective locations. The objective is to allocate each facility to a location such that the total cost is minimized"* [8]. It is a very challenging problem and even MIP solvers struggle to scale beyond

the smallest instances with less than thirty variables. The problem has many real-world applications, and has attracted the attention of the community [30].

The QAP is naturally modeled as a cost function network (CFN) [10, 11] with a single permutation constraint, and unary and binary cost functions to capture the costs of linear (on facility-location pairs) and quadratic terms (on pairs of facility-location pairs), respectively. Branch-and-bound methods for solving CFNs rely on soft local consistency (SLC) techniques [10]. The idea of SLC is to reformulate a problem into an equivalent one by rewriting the objective function. The rewritten objective function has a constant term which provides an explicit lower bound and may have infinite cost for some values, indicating they are pruned. The reformulation is computed by moving costs from the quadratic/binary level to the linear/unary level, and then to the constant term. This reformulation allows for better communication between the quadratic terms and the permutation (AllDifferent) constraint, for which a dedicated SLC based on solving the polynomial-time linear assignment problem (LAP) has recently been proposed [38].

This observation was already known in QAP and so-called reduction methods have been developed with exactly the same goal [8]. In particular, the Lagrangian relaxation proposed in [20] can be related to optimal soft arc consistency in CFNs [10]. It dominates the Gilmore-Lawler lower bound (GLB), which is a decomposition approach used in branch-and-bound methods for QAP because of its computation speed.

In crisp constraint networks, several extensions of arc consistency methods have been proposed to get better value pruning [36]. One technique is singleton arc consistency (SAC) [14, 3, 13, 29, 5]. In this paper, we propose an adaptation to CFNs of singleton node consistency (SNC), a weaker form of SAC. We show how it can be used to simulate GLB, by providing valid reformulations and even get improved lower bounds when used in conjunction with existing SLC techniques such as *existential directional arc consistency* (EDAC) [12, 10] during preprocessing.

In the following, we introduce necessary background on QAP and CFNs. Then, we present our singleton node consistency approach with SLC reformulations. Two iterative algorithms are proposed: SNC-LAP-GLB and a variant SNC-LAP-Greedy. Next, we exploit symmetries present in some specific QAP instances using an inverted formulation of the problem and a global cardinality constraint (GCC). Following our work on AllDifferent, and the relationship with the semi-assignment problem (SAP), we develop SLC techniques for the case of GCC in closed form. Replacing LAP by SAP in the previous algorithms, we obtain SNC-SAP-GLB and SNC-SAP-Greedy. Last, we give detailed experimental results for these four algorithms compared to CP and OR approaches on the QAPLIB benchmark and conclude.

The closest related work in CFNs is *virtual singleton arc consistency via super-reparametrizations* [16, 17]. However, a super-reparametrization does not reformulate the problem into an equivalent one. Also, SNC with the greedy strat-

egy applied on binary CFNs (without the AllDifferent constraint) implies the *existential arc consistent* (EAC) property [12].

2 Background

2.1 Linear Assignment Problem

The Linear Assignment Problem (LAP) of size n is defined by

$$l = \min \sum_{i=1}^{n} \sum_{u=1}^{n} c_{iu} x_{iu} \tag{1}$$

$$\sum_{u=1}^{n} x_{iu} = 1 \quad \forall i \in \{1, \ldots, n\} \tag{2}$$

$$\sum_{i=1}^{n} x_{iu} = 1 \quad \forall u \in \{1, \ldots, n\} \tag{3}$$

$$x_{iu} \in \{0, 1\} \quad \forall i, u \in \{1, \ldots, n\}^2 \tag{4}$$

The constraint matrix Eq.(2)–(3) being totally unimodular, it can be solved by linear programming algorithms in polynomial time, or *e.g.,* using the Hungarian [27] or LapJV [24] algorithms in $O(n^3)$ time.

2.2 Quadratic Assignment Problem

A Quadratic Assignment Problem (QAP) is defined by a set of n facilities and a set of n locations. For each pair (i, j) of facilities, a flow a_{ij} is specified, and for each pair (u, v) of locations, a distance b_{uv} is specified. Finally, there is a cost c_{iu} of locating facility i at location u. The problem is to assign all facilities to different locations while minimizing the total cost, assuming a unit cost per supply and unit of distance.

Formally, a QAP of size n is defined by

$$q = \min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{u=1}^{n} \sum_{v=1}^{n} a_{ij} b_{uv} x_{iu} x_{jv} + \sum_{i=1}^{n} \sum_{u=1}^{n} c_{iu} x_{iu} \tag{5}$$

$$\sum_{u=1}^{n} x_{iu} = 1 \quad \forall i \in \{1, \ldots, n\} \tag{6}$$

$$\sum_{i=1}^{n} x_{iu} = 1 \quad \forall u \in \{1, \ldots, n\} \tag{7}$$

$$x_{iu} \in \{0, 1\} \quad \forall i, u \in \{1, \ldots, n\}^2 \tag{8}$$

where coefficients a, b, and c are assumed to be non-negative integers. W.l.o.g., we assume $a_{ii} = b_{uu} = 0$ for all pairs (i, u). If not, c_{iu} can be updated by

$c_{iu} \leftarrow c_{iu} + a_{ii}b_{uu}$. We have $x_{iu} = 1$ if facility i is assigned to location u, otherwise, $x_{iu} = 0$. The problem is NP-hard, as it contains the Traveling Salesman Problem as a special case [8].

If we ignore the quadratic terms in the objective, we get a LAP. Solving this gives a basic lower bound for QAP.

2.3 Gilmore-Lawler Bound

The Gilmore-Lawler bound (GLB) [21, 41] is a strengthened variant of the simple LAP bound. It considers solving first the following n^2 specific LAPs LAP^{iu}_{GLB}, for each pair $(i, u) \in \{1, \ldots, n\}^2$:[4]

$$LAP^{iu}_{GLB} : l_{iu} = \min \sum_{j=1, j\neq i}^{n} \sum_{v=1, v\neq u}^{n} a_{ij}b_{uv}x_{jv} \tag{9}$$

$$\sum_{v=1, v\neq u}^{n} x_{jv} = 1 \quad \forall j \in \{1, \ldots, n\} \setminus \{i\} \tag{10}$$

$$\sum_{j=1, j\neq i}^{n} x_{jv} = 1 \quad \forall v \in \{1, \ldots, n\} \setminus \{u\} \tag{11}$$

$$x_{jv} \in \{0, 1\} \quad \forall j \in \{1, \ldots, n\} \setminus \{i\}, v \in \{1, \ldots, n\} \setminus \{u\} \tag{12}$$

Observe that each LAP^{iu}_{GLB} objective function Eq.(9) involves a different part of the a and b coefficients, which means that the individual bounds we compute can be added to the coefficient of the corresponding linear term x_{iu}.

The GLB is obtained by solving this final LAP:

$$glb = \min \sum_{i=1}^{n} \sum_{u=1}^{n} (l_{iu} + c_{iu})x_{iu} \tag{13}$$

$$\sum_{u=1}^{n} x_{iu} = 1 \quad \forall i \in \{1, \ldots, n\} \tag{14}$$

$$\sum_{i=1}^{n} x_{iu} = 1 \quad \forall u \in \{1, \ldots, n\} \tag{15}$$

$$x_{iu} \in \{0, 1\} \quad \forall i, u \in \{1, \ldots, n\}^2 \tag{16}$$

2.4 QAP as a Weighted Constraint Satisfaction Problem

A Cost Function Network is a triplet $P = (X, D, F)$ where $X = \{x_1, \ldots, x_n\}$ is a set of n decision variables with finite domain D_i for variable x_i. An assignment I to a set of variables S is a mapping from each variable $x_i \in S$ to a value

[4] These LAPs can all be solved together in $O(n^2 \log(n))$ time (see *e.g.,* [41, 8]). However, it assumes a decomposition of their objective function as a product of two vectors, which is not preserved by the problem reformulations done in Section 3.

in D_i. Let $\ell(S) = \Pi_{i \in S} D_i$ denote the set of all possible assignments for S. An assignment is complete if $S = X$, otherwise it is partial. F is a set of cost functions, each cost function $f_S \in F$ applies to a subset S of X and returns a non-negative integer cost depending on its variable assignment. Returning an infinite cost means a forbidden assignment. A cost function returning only zero or infinite costs defines a hard constraint.

The binary Weighted Constraint Satisfaction Problem (WCSP) is defined by:[5]

$$\min \sum_{i=1}^{n} \sum_{j>i}^{n} f_{ij}(x_i, x_j) + \sum_{i=1}^{n} f_i(x_i) + f_\varnothing \tag{17}$$

$$x_i \in D_i \quad \forall i \in \{1, \ldots, n\} \tag{18}$$

where $f_\varnothing$, f_i, f_{ij} are respectively called empty, unary, and binary cost functions. $f_\varnothing$ is a constant that serves as a trivial problem lower bound. The problem is NP-hard [11]. Complete solving approaches are usually based on branch-and-bound, whose efficiency greatly depends on the quality of its lower bound [10]. In the past, several so-called *Soft Local Consistency* (SLC) methods, including NC, EDAC [12, 10], and $F\emptyset IC$ [32, 38], have been proposed in order to reformulate P into an equivalent problem P' having the same cost for any complete assignment, but a better $f_\varnothing$.

The simplest SLC method is Node Consistency (NC): if all domain values of a variable x_i have a nonzero unary cost, then increase $f_\varnothing \leftarrow f_\varnothing + \min_{x_i \in D_i} f_i(x_i)$ and decrease the unary cost function by this minimum: $\forall x_i \in D_i, f_i(x_i) \leftarrow f_i(x_i) - \min_{x_i \in D_i} f_i(x_i)$. Moreover, if a problem upper bound ub is available, then NC will also prune all domain values $x_i \in D_i$ for which it holds that $f_i(x_i) + f_\varnothing \geq ub$. EDAC is a stronger SLC method which enforces NC and more, but is limited to binary (and ternary) cost functions. $F\emptyset IC$ has been introduced for the propagation of global constraints in CFNs (Knapsack [32], AllDifferent [38]). The aim of $F\emptyset IC$ is to enforce that for each nonunary cost function $f_S, |S| > 1$, it admits a partial assignment $I \in \ell(S)$ such that $f_S(I) + \sum_{x_i \in S} f_i(I[\{x_i\}]) = 0$. EDAC is stronger than $F\emptyset IC$ on binary cost functions.

A QAP of size n can be formulated as a WCSP $P = (X, D, F)$ with:

$$D_i = \{1, \ldots, n\} \qquad \forall i \in \{1, \ldots, n\} \tag{19}$$

$$f_{ij}(x_i = u, x_j = v) = a_{ij} b_{uv} + a_{ji} b_{vu} \qquad \forall i, j, u, v \in \{1, \ldots, n\}^4, i < j, u \neq v \tag{20}$$

$$f_{ij}(x_i = u, x_j = u) = \infty \qquad \forall i, j, u \in \{1, \ldots, n\}^3 \tag{21}$$

$$f_i(x_i = u) = c_{iu} \qquad \forall i, u \in \{1, \ldots, n\}^2 \tag{22}$$

$$f_\varnothing = 0 \tag{23}$$

[5] Later, we use the term WCSP indifferently to refer to the CFN or its associated minimization problem.

Here, decision variables are associated to facilities and values to locations. So $\{x_i = u\}$ in the WCSP represents $x_{iu} = 1$ in the QAP. We will see in Section 4 that associating variables to locations instead of facilities allows to exploit symmetries in some cases.

Using a AllDifferent(X) hard global constraint instead of Eq.(21) gives better propagation and stronger lower bounds as shown in [38]. We call this QAP model with the AllDifferent, P_{alldiff}.[6]

Reformulation with the AllDifferent Constraint The combination of the AllDifferent constraint and the unary cost functions $f_i \forall i \in \{1, \ldots, n\}$ is exactly the LAP of Eq.(1–4) where $c_{iu} \equiv f_i(x_i = u)$. We have shown in [38] a reformulation of P_{alldiff} by solving the corresponding LAP, then adding its optimum l to the problem lower bound, $f_\varnothing \leftarrow f_\varnothing + l$, and the reduced costs $rc(i, u)$ in the LAP become the new unary costs: $f_i(x_i = u) \leftarrow rc(i, u)$ for all $i, u \in \{1, \ldots, n\}^2$. This is a valid reformulation because the LAP contains only equality constraints [38, Theorem 1] [31]. Applying this reformulation enforces exactly $F\emptyset IC$ on the AllDifferent constraint. After the reformulation, P_{alldiff} has a better lower bound $f_\varnothing$, which can be used by the branch-and-bound method to prune the search when $f_\varnothing \geq ub$ or prune domains by NC when $f_i(x_i) + f_\varnothing \geq ub$. We have shown [38] that these pruning rules are stronger than the ones developed for the Weighted AllDifferent constraint [37, 9].

3 Singleton Node Consistency and Linear Assignment Problem

Singleton Arc Consitency (SAC) has been proposed in the context of Constraint Satisfaction Problems. The main idea is to test each partial assignment $\{x_i = u\} \forall x_i \in X, u \in D_i$ and propagate using arc consistency. If a domain wipe-out occurs, then u can be removed from D_i [14]. Refinements of this base algorithm improve practical or worst case performance [3, 13, 29, 5].

Here, our goal is to increase the unary costs by using search and $F\emptyset IC$, a weaker SLC method than soft arc consistency methods such as EDAC.

Based on the previously-defined QAP problem P_{alldiff}, we define the following subproblem P^{iu}_{GLB} where the value u is assigned to the variable x_i and only binary

[6] In practice, we add Eq.(21) as a redundant binary constraint to the AllDifferent constraint only if the function $f_{ij}(x_i = u, x_j = v)$ returns a nonzero cost for some $u \neq v$, otherwise the function f_{ij} can be discarded which makes the WCSP having less binary cost functions.

cost functions (or half quantities of them) related to $\{x_i = u\}$ are kept:[7]

$$P^{iu}_{GLB} : l'_{iu} = \min \sum_{j<i} f_{ji}(x_j, x_i = u) + \frac{1}{2} \sum_{j>i} f_{ij}(x_i = u, x_j) \tag{24}$$

$$\textsc{AllDifferent}(X \setminus \{x_i\}) \tag{25}$$

$$x_j \in D_j \setminus \{u\} \quad \forall j \in \{1, \ldots, n\} \setminus \{i\} \tag{26}$$

Our main observation is that we can reformulate the original problem P_{alldiff} using the solution of the modified LAP^{iu}_{GLB}. In P_{alldiff}, the optimum l'_{iu} is added to the unary cost $f_i(x_i = u)$ and the reduced costs $rc(j, v)$ update binary costs:

$$f_i(x_i = u) \leftarrow f_i(x_i = u) + l'_{iu} \tag{27}$$

$$f_{ji}(x_j = v, x_i = u) \leftarrow rc(j, v) \qquad \forall j < i, v \in D_j \tag{28}$$

$$f_{ij}(x_i = u, x_j = v) \leftarrow \frac{f_{ij}(x_i = u, x_j = v)}{2} + rc(j, v) \qquad \forall j > i, v \in D_j \tag{29}$$

Theorem 1. *The updates in Equations* (27)–(29) *preserve equivalence.*

Proof (Sketch). The proof is the same as for the AllDifferent constraint [38] but now the modified costs are conditioned on the partial assignment $\{x_i = u\}$. After conditioning on $\{x_i = u\}$, the lower bound, which is a constant term, becomes a linear term and the reduced costs, which are linear, become quadratic terms. □

Algorithm 1, SNC-LAP-GLB, iteratively reformulates the problem P_{alldiff} (lines 7-13) by solving the modified LAP^{iu}_{GLB} (line 5) for all (i, u) and applying the updates (27)–(29). After that, it enforces $F\emptyset IC$ on AllDifferent (line 16) to get a lower bound, which it improves with EDAC on P_{alldiff} (line 17). EDAC moves costs between unary and binary cost functions, which may invalidate previous optimal LAPs, thus it iterates until a stopping condition is met.

We show next that SNC-LAP-GLB is no weaker than GLB.

Theorem 2. *For any QAP with symmetric flow and distance matrices, the* SNC-LAP-GLB *bound is no smaller than the GLB bound.*

Proof. We prove first that for all variables i, $f_i(x_i = u) \geq l_{iu}$, where l_{iu} is defined in Equation (9). We show this by induction on the variable i.

For $i = 1$, due to our assumption of symmetric flow and distance matrices, enforcing $F\emptyset IC$ on P^{1u}_{GLB} is equivalent to solving LAP^{1u}_{GLB} for all u. We have $\frac{f_{1j}(x_1=u, x_j=v)}{2} = \frac{a_{1j}b_{uv} + a_{j1}b_{vu}}{2} = a_{1j}b_{uv}$, thus $l'_{1u} = l_{1u}$.

Assume the inductive hypothesis holds for all $j \leq i$. We show that it holds for $i + 1$. If the reduced costs of LAP^{iu}_{GLB} are all null, then in the ith reformulation of P_{alldiff}, we have $f_{ij}(x_i = u, x_j = v) \leftarrow \frac{f_{ij}(x_i=u, x_j=v)}{2} \forall j > i, v \in D_j$.

[7] This is why we call our approach singleton node consistency because we do not want interactions with other binary cost functions that would require introducing new higher-arity (ternary) cost functions to ensure a correct reformulation.

Otherwise, binary costs are higher due to positive reduced costs. Thus, the first term $f_{ji+1}(x_j, x_{i+1} = u) \forall j < i+1$ in Eq.24 of P_{GLB}^{i+1u} is greater than or equal to $a_{ji+1}b_{vu}$ after the previous i reformulations. The second term $f_{i+1j}(x_{i+1} = u, x_j) \forall j > i+1$ has not been modified by the previous reformulations, and it is also greater than or equal to $a_{i+1j}b_{uv}$.

Since the final step of SNC-LAP-GLB (line 16) solves a LAP with linear costs at least as great as those used in the final step of GLB, we get that the optimum of the former LAP is at least as great as the latter, as required. □

Algorithm 1 SNC-LAP-GLB(P_{alldiff}): Enforcing singleton node consistency with GLB-like LAP on a quadratic assignment problem P_{alldiff}.

1: **repeat**
2: **for all** $x_i \in X$ **do**
3: **for all** $u \in D_i$ **do**
4: */* Temporarily assign $x_i = u$ and enforce $F\emptyset IC$ on P_{GLB}^{iu} */*
5: $(l'_{iu}, rc) = \text{Solve-LAP}_{GLB}^{iu}(\{f_{ji} \mid j < i\} \cup \{\frac{f_{ij}}{2} \mid j > i\})$
6: */* Reformulate $P_{alldiff}$ */*
7: **for all** $j < i, v \in D_j$ **do**
8: $f_{ji}(x_j = v, x_i = u) \leftarrow rc(j, v)$
9: **end for**
10: **for all** $j > i, v \in D_j$ **do**
11: $f_{ij}(x_i = u, x_j = v) \leftarrow \frac{f_{ij}(x_i=u, x_j=v)}{2} + rc(j, v)$
12: **end for**
13: $f_i(x_i = u) \leftarrow f_i(x_i = u) + l'_{iu}$
14: **end for**
15: **end for**
16: Enforce $F\emptyset IC$ on AllDifferent(X)
17: Enforce EDAC and $F\emptyset IC$ on P_{alldiff}
18: **until** stopping condition

The time complexity of one pass over all the variables at line 2 is in $O(n^5)$, assuming $O(n^3)$ for solving LAP_{GLB}^{iu}. The space complexity is the same as for representing P_{alldiff}, in $O(n^4)$, using tables for binary cost functions. In our implementation, the stopping condition is that the lower bound $f_\varnothing$ has not increased sufficiently after one pass over all the variables, specifically if $\frac{f_\varnothing - f_\varnothing^{prev}}{f_\varnothing} \leq 1e-4$.

3.1 Improving Further the Iterated Gilmore-Lawler Bound

The intuitive idea of GLB is to distribute the binary costs evenly across the unary costs of all variables before solving the final LAP. We propose another strategy that will try to move all the unary and binary costs towards a single variable in order to increase its own unary cost function as much as possible. We call this strategy Greedy. We define the following subproblem P_{Greedy}^{iu} where we

assign $x_i = u$ and keep only unary and binary cost functions related to $\{x_i = u\}$:

$$P^{iu}_{Greedy} : s_{iu} = \min \sum_{j<i} f_{ji}(x_j, x_i = u) + \sum_{j>i} f_{ij}(x_i = u, x_j) + \sum_{j \neq i} f_j(x_j) \quad (30)$$

$$\textsc{AllDifferent}(X \setminus \{x_i\}) \quad (31)$$

$$x_j \in D_j \setminus \{u\} \qquad \forall j \in \{1, \ldots, n\} \setminus \{i\} \quad (32)$$

Enforcing $F\emptyset IC$ on this problem is equivalent to solving LAP^{iu}_{Greedy}, an LAP with a linear objective function given by Eq.30.

Algorithm 2 SNC-LAP-GREEDY(P_{alldiff}): Enforcing singleton node consistency with Greedy LAP on a quadratic assignment problem P_{alldiff}.

```
1: repeat
2:   for all x_i ∈ X do
3:     /* Move unary costs into binary cost functions related to x_i */
4:     for all x_j ∈ X \ {x_i}, v ∈ D_j do
5:       for all u ∈ D_i do
6:         f_ij(x_i = u, x_j = v) ← f_ij(x_i = u, x_j = v) + f_j(x_j = v)
7:       end for
8:       f_j(x_j = v) ← 0
9:     end for
10:    for all u ∈ D_i do
11:      /* Temporarily assign x_i = u and enforce FØIC on P^iu_Greedy */
12:      (s_iu, rc) = SOLVE-LAP^iu_Greedy({f_ij | j ≠ i})
13:      /* Reformulate P_alldiff */
14:      for all x_j ∈ X \ {x_i}, v ∈ D_j do
15:        f_ij(x_i = u, x_j = v) ← rc(j, v)
16:      end for
17:      f_i(x_i = u) ← f_i(x_i = u) + s_iu
18:    end for
19:    Enforce EDAC and FØIC on P_alldiff
20:  end for
21: until stopping condition
```

We define Algorithm 2, SNC-LAP-GREEDY, which reformulates P_{alldiff} by repeatedly solving LAP^{iu}_{Greedy} (line 12) for all (i, u) until a stopping condition is met (the same as for SNC-LAP-GLB). After singleton tests and reformulations for all domain values of the target variable (for-loop line 10), we enforce EDAC on the binary cost functions of P_{alldiff} and $F\emptyset IC$ on ALLDIFFERENT (line 19).

We give the following theorem without proof, as it is largely the same as the proof of Theorem 1. The main difference is that we cannot directly use the unary costs of P_{alldiff} inside LAP^{iu}_{Greedy}, it has to be done through the binary cost functions related to (i, u). Therefore, we move the unary costs to the binary cost

functions related to x_i before solving LAP^{iu}_{Greedy} (lines 4-9). Alg. 2 has the same worst-case time and space complexity as Alg. 1. However, the greedy algorithm does not guarantee a better bound than SNC-LAP-GLB or GLB.

Theorem 3. SNC-LAP-GREEDY *preserves equivalence.*

Pruned values $v \in D_j$ found by enforcing $F\emptyset IC$ on P^{iu}_{GLB} in Alg. 1 or P^{iu}_{Greedy} in Alg. 2 become forbidden tuples in P_{alldiff}. If a value v is pruned by all singleton tests ($\forall u \in D_i$), it can be removed from P_{alldiff}, as in SAC and constructive disjunction [40, 6]. Because EDAC is equivalent to classical AC on forbidden tuples [10], Alg. 1 (resp. Alg. 2) prunes such values at line 17 (resp. 19).

Example 1. Consider a CFN with 3 variables $\{x_1, x_2, x_3\}$ with 3 values each, an ALLDIFFERENT constraint over them, and 3 binary cost functions $f_{ij}(x_i = u, x_j = v) = 2\,\forall i, j, u, v \in \{1, 2, 3\}^4, i < j, u \neq v$ (we omit unary and binary tuples that have cost 0). Both SNC-LAP-GLB and SNC-LAP-GREEDY produce an optimal lower bound $f_\varnothing$ of 6 and all the unary and binary cost functions are set to zero after reformulation.

4 Exploiting Symmetries in the Flow Matrix of the QAP

For a subset of the QAPLIB instances, it has been shown [19] that is possible to group facilities into equivalence classes. Two facilities i and j are equivalent if the following conditions are met:

$$a_{ij} = a_{ji} \tag{33}$$

$$a_{ik} = a_{jk}, a_{ki} = a_{kj} \qquad \forall k \in \{1, \ldots, n\} \setminus \{i, j\} \tag{34}$$

$$c_{iu} = c_{ju} \qquad \forall u \in \{1, \ldots, n\} \tag{35}$$

$$b_{uv} = b_{vu} \qquad \forall u, v \in \{1, \ldots, n\}^2 \tag{36}$$

In CFNs, such variables are said interchangeable. For any solution of P_{alldiff}, we can permute the values of x_i and x_j to get a new solution with the same cost.

In order to exploit the symmetries, we need to invert the CFN model.[8] In the inverted model, variables are associated to locations and values to facilities. The resulting model is called $P^{\tau}_{\text{alldiff}}$.

Let us assume we have d equivalent classes $\{E_1, \ldots, E_d\}$ of size $m_k = |E_k|, k \in \{1, \ldots, d\}$. We define P^{τ}_{gcc}, the inverted QAP model of size n, with reduced domains of size d, grouping all facilities/values of the same equivalence class E_k into a single representative value $\varphi(k)$, and replacing ALLDIFFERENT by a Global Cardinality Constraint (GCC) [23]. In GCC, we set the upper bound on the

[8] It has been observed [19] that symmetries can only be exploited in the inverted model and not in the original model.

number of occurrences of $\varphi(k)$ to m_k. We define $P^{\tau}_{gcc} = (X, D, F)$:

$$D_i = \{1, \ldots, d\} \qquad \forall i \in \{1, \ldots, n\} \tag{37}$$

$$f_{ij}(x_i = k, x_j = k') = \quad a_{\varphi(k)\varphi(k')}b_{ij} + a_{\varphi(k')\varphi(k)}b_{ji} \qquad \forall i, j \in \{1, \ldots, n\}^2, i < j, k, k' \in \{1, \ldots, d\}^2 \tag{38}$$

$$f_i(x_i = k) = \quad c_{\varphi(k)i} \qquad \forall i \in \{1, \ldots, n\}, k \in \{1, \ldots, d\} \tag{39}$$

$$f_{\varnothing} = \quad 0 \tag{40}$$

$$\mathrm{GCC}(X, \{m_1, \ldots, m_d\}) \tag{41}$$

By definition, $d \leq n$ and $\sum_{k=1}^{d} m_k = n$. The GCC is in *closed form*, *i.e.,* all variables must take a value in $\{1, \ldots, d\}$ and each value k must be used exactly m_k times.

Reformulation with the Global Cardinality Constraint The combination of the GCC constraint in closed form and the unary cost functions $f_i \forall i \in \{1, \ldots, n\}$ corresponds exactly to the Semi-Assignment Problem (SAP) [25, 8]:

$$SAP : o = \min \sum_{i=1}^{n} \sum_{k=1}^{d} c_{\varphi(k)i} x_{ik} \tag{42}$$

$$\sum_{k=1}^{d} x_{ik} = 1 \quad \forall i \in \{1, \ldots, n\} \tag{43}$$

$$\sum_{i=1}^{n} x_{ik} = m_k \quad \forall k \in \{1, \ldots, d\} \tag{44}$$

$$x_{ik} \in \{0, 1\} \quad \forall i \in \{1, \ldots, n\}, k \in \{1, \ldots, d\} \tag{45}$$

where 0/1 variable $x_{ik} = 1$ corresponds to $x_i = k$ in P^{τ}_{gcc}. This problem can be solved efficiently by a modified version of LapJV algorithm in $O(dn^2)$ time [25]. As with the AllDifferent constraint, we can reformulate P^{τ}_{gcc}. The optimum o of the SAP is added to the problem lower bound $f_{\varnothing} \leftarrow f_{\varnothing} + o$ and the reduced costs $rc(i, k)$ of each variable x_{ik} in the SAP become the new unary costs: $f_i(x_i = k) \leftarrow rc(i, k), i \in \{1, \ldots, n\}, k \in \{1, \ldots, d\}$. This reformulation enforces $F\emptyset IC$ on the GCC. This offers the same advantages compared to GCC with costs [35] as $F\emptyset IC$ on AllDifferent does compared to Weighted AllDifferent.

We can apply SNC on P^{τ}_{gcc} by modifying Algorithm 1 (resp. Alg. 2) to use SAP on GCC instead of LAP on AllDifferent. The resulting algorithms SNC-SAP-GLB (resp. SNC-SAP-Greedy) have a time complexity in $O(d^2n^3)$.

5 Experimental Results

We implemented in TOULBAR2, an open-source C++ exact WCSP solver, a propagator for enforcing $F\emptyset IC$ on the GCC in closed form, based on the modified LAPJV algorithm [25].[9] TOULBAR2 already includes a propagator for enforcing $F\emptyset IC$ on ALLDIFFERENT [38]. We implemented four methods combining each of the two propagators with each of the two strategies GLB (Alg. 1) and Greedy (Alg. 2). Due to their relatively high cost, we apply them only in preprocessing.

We compared four versions of TOULBAR2 having different preprocessing:

- no preprocessing (TOULBAR2 with default options), called LAP in the results when applied to QAP models with the ALLDIFFERENT constraint (same results as tb2+LAPJV in [38]) and SAP when applied to a model with GCC,
- SNC-LAP-GLB using the GLB strategy (options *-S -glb=1*),
- SNC-LAP-GREEDY using the Greedy strategy (options *-S -glb=0*),
- SNC-LAP-GLB followed by SNC-LAP-GREEDY (options *-S -glb=2*), called SNC-LAP-GLB-GREEDY in the results.

For each version, after preprocessing, TOULBAR2 continues the search, enforcing EDAC [12] on unary and binary cost functions, and $F\emptyset IC$ on ALLDIFFERENT and GCC, at every node of a hybrid best/depth-first branch-and-bound search [2], using binary branching and Maximum Cardinality Search for the DAC ordering.[10] TOULBAR2 used the *dom/wdeg* variable ordering heuristic [7] with last conflict [28] and EAC *support value* ordering heuristic [10, 39], combined with solution phase saving [15]. Unless reported otherwise, all tests were run on a 2.5GHz Intel Xeon E5-2680 (2014), using one thread per CPU, and with a CPU-time limit of 1,200 seconds.

We took the 132 smallest of the 136 instances from the QAPLIB.[11] We compared the four versions of TOULBAR2 against Google OR-Tools CP-SAT, an open-source state-of-the-art constraint programming solver, and IBM CPLEX, a state-of-the-art integer programming solver.[12] Unary and binary cost functions are encoded using the support encoding for CPLEX [22], and to binary and ternary constraints in extension with extra objective variables for CP-SAT. The permutation constraint is encoded for CPLEX as binary constraints enforcing that any pair of two variables cannot take the same value. It is encoded as an ALLDIFFERENT constraint in CP-SAT and TOULBAR2. The GCC constraint is only tested using TOULBAR2 in the inverted model P^{τ}_{gcc}.[13]

[9] https://github.com/toulbar2/toulbar2, branch lapjv.

[10] TOULBAR2 options *-d: -o -O=-1*

[11] http://coral.ise.lehigh.edu/wp-content/uploads/2014/07/qapdata.tar.gz, sizes less than 100 variables.

[12] https://github.com/google/or-tools CP-SAT v9.14 in free-search mode, CPLEX version 22.1.1.0 with non-premature stop parameters *EPAGAP=EPGAP=EPINT=0*. All single-threaded.

[13] CP-SAT uses a decomposition of GCC into linear constraints. Our preliminary tests showed worse results on the inverted model with the decomposed GCC.

Preprocessing	AvgInitialLbGap	AvgSncIterations	AvgSncTimes
GLB	30.46	N/A	N/A
SNC-LAP-GLB-Greedy	**24.13**	352.53	60.76 sec.
SNC-LAP-Greedy	26.38	368.67	73.08 sec.
SNC-LAP-GLB	26.32	**160.38**	**25.46 sec.**
LAP	83.63	N/A	N/A

Table 1: Initial lower bound gap percent., nb. of iter., and av. time for QAPLIB.

Gap	mean	median	std	Score	Nb. instances
CPLEX	16.41	9.35	20.38	25.00	109
CP-SAT	5.12	2.29	11.00	33.00	122
LAP	4.01	1.88	5.27	41.50	**132**
SNC-LAP-Greedy	1.44	**0.00**	**2.29**	83.00	131
SNC-LAP-GLB	1.50	**0.00**	2.56	89.00	**132**
SNC-LAP-GLB-Greedy	**1.42**	**0.00**	2.33	**92.00**	131

Table 2: Optimality gap percentages on the original model for QAPLIB.

5.1 Comparison of Initial Lower Bounds

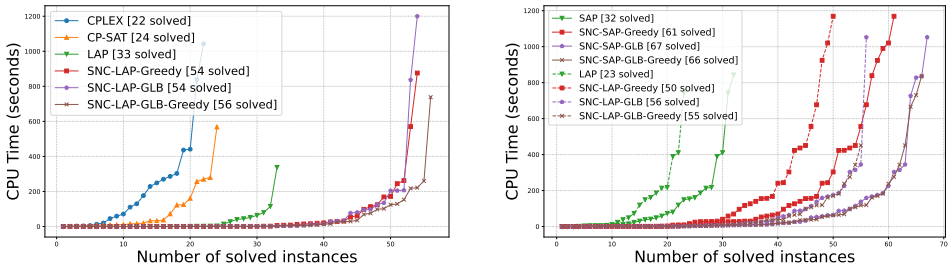
Tab. 1 gives average initial lower bound gap ($1-\frac{lb}{bestsolknown}$) found by the original GLB method and by the four preprocessing methods on P_{alldiff}. It also reports the average number of iterations on variables (for-loop line 2) and CPU-time of SNC methods. Without preprocessing, LAP had a poor lower bound. SNC methods were much better, all being superior on average to original GLB. SNC with the GLB or Greedy strategy got the same quality of lb, but SNC-LAP-Greedy was 2-3x slower to converge than SNC-LAP-GLB (see also Fig.2a). Hopefully, SNC-LAP-GLB-Greedy is a bit faster than SNC-LAP-Greedy thanks to the reformulations initially made by SNC-LAP-GLB, and it got the best lb gap. Similar results are observed on the other models $P^{\tau}_{alldiff}$ and P^{τ}_{gcc}.

For instance *tai100a*, SNC-LAP-GLB (resp. SNC-LAP-GLB-Greedy) found an initial lower bound of 15,864,722 in 196 seconds and 300 iterations (resp. 15,931,389 in 405s, 500 iter.), which are greater than the original GLB bound of 15,824,355 and the best bound of 15,844,731 reported on QAPLIB website using a semidefinite relaxation-matrix splitting bound [34].[14]

5.2 Comparison of Solving Times

On the original model P_{alldiff}, LAP solved 33 instances, whereas all SNC methods solved more than 54. SNC-LAP-GLB-Greedy got the best results (56 solved). On the inverted model $P^{\tau}_{\text{alldiff}}$, without exploiting symmetries, performances decreased for LAP (23 solved) and SNC-LAP-Greedy (50 solved), and remained stable for the two other SNCs. By exploiting symmetries in P^{τ}_{gcc}, we observed a very nice improvement for all the methods, SNC-SAP-GLB being the best (67 solved). The largest solved instance is *lipa90b* with size $n = 90$ in 87 seconds

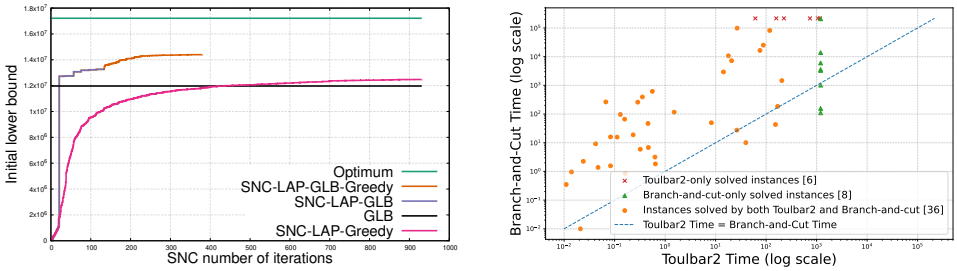
[14] https://coral.ise.lehigh.edu/data-sets/qaplib/qaplib-problem-instances-and-solutions/#Ta



(a) Original model ($P_{alldiff}$)

(b) Inverted models ($P^{\tau}_{alldiff}$ and P^{τ}_{gcc})

Fig. 1: Cactus plots on the original and inverted models for QAPLIB benchmark.



(a) Initial lower bounds obtained on QAPLIB instance *els19* using the original model. SNC-LAP-GLB-GREEDY took 0.69s and 380 iter. in preprocessing and then tb2 solved in 5.82s.

(b) TOULBAR2 vs Branch-and-Cut [18] runtime comparison. Note the drastically different hardware and timeouts (20 minutes for tb2 / 2.5 days for B&C) makes comparison difficult.

Fig. 2: Detailed results for the QAPLIB benchmark.

and 100 nodes by SNC-SAP-GLB (72s for its preprocessing). As previously reported [38], CFN methods performed much better than CP-SAT (24 solved) and default OR methods (CPLEX, 22 solved).

5.3 Comparison of Solution Quality

In Table2, we report optimality gap distribution ($\frac{ub-best}{ub}$), XCSP3 competition score [38], and number of instances with at least one solution found on P_{alldiff} (similar results are observed on the other models). The best average gap of 1.42% is obtained by SNC-LAP-GLB-GREEDY. However, it could not finish its preprocessing in less than 20 minutes for one instance (*tai100b*). SNC-LAP-GLB found a solution for all 132 instances with an average gap of 1.5%. It is clearly better than LAP (4%), CP-SAT (5%), and CPLEX (16%).

5.4 Comparison with a Dedicated Branch-and-Cut Approach

Fig.2b gives a log-log scatter plot between our best approach using TOULBAR2 (whatever the chosen SNC preprocessing or QAP model) and a branch-and-cut

Instance	n	d	OPT	CPLEX [19]	SNC-SAP-GLB-GREEDY
esc16a	16	9	68	**0.35**	0.65
esc16b	16	7	292	3.07	**2.06**
esc16c	16	12	160	130.98	**73.14**
esc16d	16	12	16	**0.51**	77.22
esc16e	16	8	28	**0.05**	1.11
esc16f	16	1	0	**0.00**	**0.00**
esc16g	16	9	26	**0.04**	0.18
esc16h	16	5	996	0.23	**0.11**
esc16i	16	10	14	**0.18**	7.88
esc16j	16	7	8	**0.03**	0.13
esc32c	32	10	642	**9,643.82**	-
esc32d	32	13	200	**2,973.26**	-
esc32e	32	6	2	**0.04**	106.33
esc32g	32	7	6	**0.06**	5.86
esc64a	64	15	116	**509.87**	-
tai64c	64	2	1,855,928	18,250.40	**995.51**

Table 3: Comparison on 16 QAPLIB instances with flow matrix symmetry and using the inverted model (P^{τ}_{gcc}). A "-" means CPU-time limit reached.

(B&C) method dedicated for QAP [18], using the results reported in that paper. Among the 71 instances TOULBAR2 could solve in less than 20 minutes, we kept 42 of size less than 30, removing *esc* and *lipa* families as done in B&C results. B&C could solve 44 instances with $n \leq 30$, but often taking much more time than TOULBAR2 (up to 3 orders of magnitude speedups on *bur26* and *chr* families).

5.5 Comparison with a Dedicated MILP Approach

In [19], a dedicated MILP formulation is proposed, exploiting symmetry in the flow matrix. In Table 3, we report their results obtained using CPLEX 12.2 with 8 threads on a quad core Intel Xeon CPU 3.2 GHz with 16 GB RAM. We give also the results obtained by our approach (SNC-SAP-GLB-GREEDY) using TOULBAR2 with the parallel version of its hybrid best/depth-first branch-and-bound search method [4] on a 8-core Intel Xeon E5-2687W v4 3.5 GHz with 256 GB. The CPU-time limit is fixed to 10 hours. The MILP approach produced the best results, solving 16 instances and TOULBAR2 only 13. However, on the large *tai64c* instance, our approach was 18 times faster than CPLEX, and developed less search nodes (162,790,320 compared to 1,216,074,081).

6 Conclusion

Taking ideas from CP (singleton consistency, global cardinality constraint) and OR (Gilmore-Lawler bound, linear and semi-assignment problems), we significantly improved a CFN solver on quadratic assignment problems, reaching state-of-the-art OR results in a few cases (mostly on asymmetric instances such as *bur26* and *lipa*, and also *tai64c*). In the future, we will explore other soft local consistency methods [10, 33] and problem formulation [1] to get stronger bounds.

Supplementary Materials. Detailed results (csv files) are available at:
https://web-genobioinfo.toulouse.inrae.fr/~degivry/cpaior2026/cpaior2026supp.zip.

Acknowledgments. This work has benefited from French grants managed by the National Research Agency under the "Investissements d'Avenir" program with the references ANR-18-EURE-0021 and ANR-19-PI3A-0004.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Adams, W.P., Guignard, M., Hahn, P.M., Hightower, W.L.: A level-2 reformulation–linearization technique bound for the quadratic assignment problem. European Journal of Operational Research **180**(3), 983–996 (2007)
2. Allouche, D., de Givry, S., Katsirelos, G., Schiex, T., Zytnicki, M.: Anytime Hybrid Best-First Search with Tree Decomposition for Weighted CSP. In: Proc. of CP-15. pp. 12–28. Cork, Ireland (2015)
3. Barták, R., Erben, R.: A new algorithm for singleton arc consistency. In: FLAIRS. pp. 257–262 (2004)
4. Beldjilali, A., Montalbano, P., Allouche, D., Katsirelos, G., de Givry, S.: Parallel Hybrid Best-First Search. In: Proc. of CP-22. vol. 235, pp. 7:1–7:10. Haifa, Israel (2022)
5. Bessiere, C., Cardon, S., Debruyne, R., Lecoutre, C.: Efficient algorithms for singleton arc consistency. Constraints **16**(1), 25–53 (2011)
6. Bessiere, C., Debruyne, R.: Theoretical analysis of singleton arc consistency and its extensions. Artificial Intelligence **172**(1), 29–41 (2008)
7. Boussemart, F., Hemery, F., Lecoutre, C., Sais, L.: Boosting systematic search by weighting constraints. In: ECAI. vol. 16, p. 146 (2004)
8. Burkard, R., Dell'Amico, M., Martello, S.: Assignment problems: revised reprint. SIAM (2012)
9. Claus, G., Cambazard, H., Jost, V.: Analysis of reduced costs filtering for alldifferent and minimum weight alldifferent global constraints. In: Proc. of 24th European Conference on Artificial Intelligence. vol. 325, pp. 323–330. Santiago de Compostela, Spain (2020)
10. Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence Journal **174**, 449–478 (2010)
11. Cooper, M.C., de Givry, S., Schiex, T.: Valued constraint satisfaction problems. In: A Guided Tour of Artificial Intelligence Research: Volume II: AI Algorithms, pp. 185–207. Springer (2020)
12. De Givry, S., Heras, F., Zytnicki, M., Larrosa, J.: Existential arc consistency: Getting closer to full arc consistency in weighted csps. In: IJCAI. vol. 5, pp. 84–89 (2005)
13. Debruyne, R.: Optimal and suboptimal singleton arc consistency algorithms. In: Ijcai (2005)
14. Debruyne, R., Bessière, C.: Some practicable filtering techniques for the constraint satisfaction problem. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29. pp. 412–417 (1997)
15. Demirović, E., Chu, G., Stuckey, P.J.: Solution-based phase saving for CP: A value-selection heuristic to simulate local search behavior in complete solvers. In: International Conference on Principles and Practice of Constraint Programming. pp. 99–108. Springer (2018)

16. Dlask, T., Werner, T., de Givry, S.: Bounds on weighted csps using constraint propagation and super-reparametrizations. In: Proc. of CP-21. Montpellier, France (2021)
17. Dlask, T., Werner, T., de Givry, S.: Super-Reparametrizations of Weighted CSPs: Properties and Optimization Perspective. Constraints **28**, 277–319 (2023)
18. Erdoğan, G., Tansel, B.: A branch-and-cut algorithm for quadratic assignment problems based on linearizations. Computers & Operations Research **34**(4), 1085–1106 (2007)
19. Fischetti, M., Monaci, M., Salvagnin, D.: Three ideas for the quadratic assignment problem. Operations research **60**(4), 954–964 (2012)
20. Frieze, A.M., Yadegar, J.: On the quadratic assignment problem. Discret. Appl. Math. **5**(1), 89–98 (1983)
21. Gilmore, P.C.: Optimal and suboptimal algorithms for the quadratic assignment problem. Journal of the society for industrial and applied mathematics **10**(2), 305–313 (1962)
22. Hurley, B., O'Sullivan, B., Allouche, D., Katsirelos, G., Schiex, T., Zytnicki, M., de Givry, S.: Multi-language evaluation of exact solvers in graphical model discrete optimization. Constraints An Int. J. **21**(3), 413–434 (2016)
23. Jean-Charles, R.: Generalized arc consistency for global cardinality constraint. American Association for Artificial Intelligence (AAAI 1996) pp. 209–215 (1996)
24. Jonker, R., Volgenant, A.: A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing **38**, 325–340 (1987)
25. Kennington, J., Wang, Z.: A shortest augmenting path algorithm for the semi-assignment problem. Operations Research **40**(1), 178–187 (1992)
26. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica: journal of the Econometric Society pp. 53–76 (1957)
27. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1-2), 83–97 (1955)
28. Lecoutre, C., Saïs, L., Tabary, S., Vidal, V.: Reasoning from last conflict(s) in constraint programming. Artificial Intelligence Journal **173**, 1592,1614 (2009)
29. Lecoutre, C., Cardon, S.: A greedy approach to establish singleton arc consistency. In: IJCAI. vol. 5, pp. 199–204 (2005)
30. Loiola, E.M., De Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. European journal of operational research **176**(2), 657–690 (2007)
31. Montalbano, P.: Linear Constraints and Conflict-free Learning for Graphical Models. Ph.D. thesis, Université de Toulouse (2023)
32. Montalbano, P., de Givry, S., Katsirelos, G.: Multiple-choice knapsack constraint in graphical models. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 282–299. Springer (2022)
33. Montalbano, P., de Givry, S., Katsirelos, G.: Virtual arc consistency for linear constraints in cost function networks. In: Proc. of ICTAI-25. Athens, Greece (2025)
34. Peng, J., Mittelmann, H., Li, X.: A new relaxation framework for quadratic assignment problems based on matrix splitting. Mathematical Programming Computation **2**(1), 59–77 (2010)
35. Régin, J.C.: Cost-based arc consistency for global cardinality constraints. Constraints **7**(3), 387–405 (2002)
36. Rossi, F., Van Beek, P., Walsh, T.: Handbook of constraint programming. Elsevier (2006)

37. Sellmann, M.: An arc-consistency algorithm for the minimum weight all different constraint. In: Hentenryck, P.V. (ed.) Proc. of 8th International Conference on Principles and Practice of Constraint Programming, Ithaca, NY, USA, September 9-13. Lecture Notes in Computer Science, vol. 2470, pp. 744–749. Springer (2002)
38. Sewa, G., Allouche, D., de Givry, S., Katsirelos, G., Montalbano, P., Schiex, T.: Assignment problems in cost function networks. In: Proc. of AAAI-26. Singapore (2026)
39. Trösser, F., De Givry, S., Katsirelos, G.: Relaxation-aware heuristics for exact optimization in graphical models. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 475–491. Springer (2020)
40. Van Hentenryck, P., Saraswat, V., Deville, Y.: Design, implementation, and evaluation of the constraint language cc (fd). The Journal of Logic Programming **37**(1-3), 139–164 (1998)
41. Xia, Y.: Gilmore-lawler bound of quadratic assignment problem. Frontiers of Mathematics in China **3**(1), 109–118 (2008)