A Greedy Evolutionary Hybridization Algorithm for the Optimal Network and Quadratic Assignment Problem

Mouhamadou A.M.T Baldé, Serigne Gueye, Babacar M. Ndiaye

Noname manuscript No.
(will be inserted by the editor)

A Greedy Evolutionary Hybridization Algorithm for the Optimal Network and Quadratic Assignment Problem

Mouhamadou A.M.T. Baldé · Serigne Gueye · Babacar M. Ndiaye

the date of receipt and acceptance should be inserted later

Abstract Our paper deals with a combinatorial optimization problem called the Optimal Network and Quadratic Assignment Problem. The problem has been introduced by Marc Los [18] as a model of an urban planning problem that consists in optimizing simultaneously the best location of the activities of an urban area (land-use), as well as the road network design (transportation network) in such a way to minimize as much as possible the routing and network costs. We propose a mixed-integer programming formulation of the problem, and a hybrid algorithm based on greedy and evolutionary heuristic methods. Some numerical experiments on randomly generated instances, and on real-life big data from Dakar city, show the efficiency of the method.

Keywords Transportation network · Land-use plan · Quadratic Assignment Problem · Heuristic · Metaheuristics · Big data

1 Introduction

The Optimal Network and Quadratic Assignment Problem (ON-QAP), introduced by Marc Los [18] in 1978, is a simplified planning problem dealing with the assignment of some amenities in an urban area, and the suitable design of the transportation network to route the transportation demands between these amenities as efficiently as possible. An "amenity" is defined

M.A.M.T Baldé, B.M. Ndiaye
Laboratoire de Mathématiques de la Décision et d'Analyse Numérique
LMDAN-FASEG, Université Cheikh Anta Diop
BP 4500 Dakar-Fann, 10700 Dakar, Sngal
E-mail: mouhamadouamt.balde,babacarm.ndiaye@ucad.edu.sn

S. Gueye
LIA, Université d'Avignon et des Pays de Vaucluse,
339 chemin des Meinajaries,
BP 1228, 84911 Avignon Cedex 9, France
E-mail: serigne.gueye@univ-avignon.fr

as any activity type which generates some flow (i.e moves, trips) : homes, schools, shops, leisure places, administrative centers, etc. Each amenity has a known location, and may be the origin or the destination (or both) of individual trips. The amount of trips between each pair of amenities is usually known as the Origin-Destination (O-D) matrix flows. These flows are routed on a given transportation network, through different paths, depending on individual choices, that themselves depend on different routing costs such as the path lengths and the travel times. Finding the routing paths is known in the literature as the traffic assignment problem. It has been intensively studied under various objectives and constraint assumptions (see [26] for a survey). A very basic version of the traffic assignment problem takes as an objective function the minimization of the overall sum of the path lengths (or travel times). Nevertheless, all standard versions of the problem assume that the amenity locations and the transportation network are known and fixed. As a consequence, in this basic version, given a network and an amenity assignment, the O-D flows are always routed through the shortest paths. Thus, it leads to a significant simplification of the problem not always observed in practice. In many other, more elaborated, transportation models flows are routed according using an equilibrium approach (see Y. Sheffi [31] for a survey).

The urban problem introduced by Los [18] is a more general problem in which, in addition to the traffic assignment problem, two other levels of decisions are introduced : the optimal amenity location and the optimal design of the network. The simultaneous optimization of the land-used (amenity locations) and the transportation network is then the problem of finding the best amenity locations and the best network minimizing the sum of routing costs, network costs, and amenity assignment costs (i.e cost of assigning an amenity in a given location). By assuming that the traffic flows are assigned as in the basic version described above, this very difficult problem has been formalized as a combinatorial optimization problem combining the well-known Quadratic Assignment Problem (QAP) [16], and the optimal network design problem [29]. Our paper deals with this latter problem. We propose a mathematical program and an evolutionary heuristic resolution scheme combining network design and QAP heuristic solutions.

In Section 2, the combinatorial optimization problem is formally defined. Then, we give a mixed-integer programming formulation in Section 3. In Section 4 and 5, the heuristic methods are presented. Numerical results with randomly generated instances, as well as real-life big data of the Dakar city, are provided in Section 6.

2 Problem definition and state-of-the-art

Let $G = (V, E)$ be an undirected connected weighted graph where $V = \{1, 2, ..., n\}$ is the set of nodes and E the set of edges. Each edge $e = (k, l) \in E$ between $k \in V$ and $l \in V$ can be used in both directions, by which we can introduce the set of arcs A making (V, A) a directed graph. For each edge

$(k, l) \in E$, w_{kl} represents the **direct** distance from k to l, which may differs from the direct distance from l to k w_{lk}. And b_{kl} is the building cost of the edge. Each node of G is a location in which one and only one amenity (or activity) of a set $F = \{1, 2, ..., n\}$ must be assigned. Assigning the activity $i \in F$ to the location $k \in V$ induced an assignment cost c_{ik}. Between any pair of activities i and j, some individuals, in an urban area, move. The amount of trips from i to j (resp. from j to i) is noticed f_{ij} (resp. f_{ji}). An assignment (of activities to locations) induces a routing cost, which is assumed proportional to the trip quantities and to the shortest distance that separates the amenities. If i and j are respectively located in k and l, and if the value of the shortest path between k and l is d_{kl}, then the routing cost is $f_{ij}d_{kl}$. The total location cost is defined as the sum of assignment and routing costs. Each edge may be chosen, or not, in the transportation network. The network cost is then given by the sum of the building costs of the chosen edges. The Optimal Network and Quadratic Assignment Problem (ONQAP) is the problem of finding the best assignment of activities on the node set, and the best edges to choose, in such a way to minimize the sum of the location, assignment and network costs.

ONQAP combines two well-known problems in the scientific litterature : an optimal network design problem and the quadratic assignment problem. But, at the opposed of these problems for which a rich litterature exists, few contributions dealing with problems closed to ONQAP exists in the litterature. Los [18] introduced the problem and is the first author to propose a discrete-convex programming approach in [19] to solve it. Before Los contribution, Lundqvist [23] published statement of a model. He proposed a combinatorial programming approach in which both the land-use decisions and the network decisions are treated as discrete variables : some variables representing the amount of building stock in a zone, and some binary ones denoting if a network link should be built or not. The model is solved using an heuristic scheme. Feng and Lin [24] proposed a nonlinear and multi-objective programming (MOP) model called the The Sketch Layout Model (SLM). In SLM, the land area is sub-divided in cells of equal surfaces and entities have to be assigned to them. Some interactions are considered between entities located in the cells, and one of the three objectives of the model corresponds to a quadratic objective similar to the quadratic assignment problem one. However, because of many others details not taken into account in ONQAP, SLM differs highly to our problem. For instance, the travel times on the transportation links is a function depending on the flows traversing the links. Moreover, the O-D flows are not known a priori but are computed by solving a trip distribution problem. To solve SLM, the cumulative genetic algorithm (CGA) developed by Xiong and Schneider [20] was modified to an heuristic algorithm to generate approximating non-dominated solutions. In addition to these references, we will give later, in the corresponding two sub-sections, contributions in the litterature dealing with the two sub-problems considered in ONQAP.

3 Formulation

The problem may be mathematically modeled as follows. Let x_{ik}, with $i \in F, k \in V$, be a binary variable equals to 1 if the activity i is assigned to the location k and 0 otherwise. Let also y_{kl}, with $k \in V, l \in V$, be another binary variable equal to 1 if the edge $(k,l) \in E$ is chosen and 0 otherwise. In the sequel $x = \{x_{ik}\}_{i,k=1,2,\ldots,n}$ will stand for the $n \times n$ square matrix composed of the variables x_{ik}. And $y = \{y_{kl}\}_{(k,l)\in E}$, the matrix corresponding to the variables y_{kl}.

In addition to y, we define the variables d_{kl} $(k \in V, l \in V)$ as the shortest path values in the graph defined by the variables y. d is the corresponding matrix. The location, assignment and network costs are then given by the following cubic function :

$$q(x,y,d) = \sum_{i\in F}\sum_{k\in V} c_{ik}x_{ik} + \sum_{i\in F}\sum_{k\in V}\sum_{j\in F}\sum_{l\in V} f_{ij}d_{kl}x_{ik}x_{jl} + \sum_{(k,l)\in E} b_{kl}y_{kl}.$$

To ensure a suitable computation of the variables d_{kl}. Let z_{ij}^{kl} $(k,l,i,j \in V)$ be equal to 1 if the shortest path between k and l used the arc $(i,j) \in A$ and 0 otherwise. $(ONQAP)$ is obtained by minimizing the objective function above under a set of constraints described below.

$$(ONQAP): \quad Min\ q(x,y,d) \tag{1}$$

$$s-t: \quad \sum_{i\in F} x_{ik} = 1 \qquad k \in V \tag{2}$$

$$\sum_{k\in V} x_{ik} = 1 \qquad i \in F \tag{3}$$

$$\sum_{\substack{j\in V\\(m,j)\in A}} z_{mj}^{kl} - \sum_{\substack{i\in V\\(i,m)\in A}} z_{im}^{kl} \tag{4}$$

$$= \begin{cases} 1\ if\ m = k, \\ -1\ if\ m = l, \\ 0\ otherwise. \end{cases} \qquad m \in V; k \neq l \in V \tag{5}$$

$$d_{kl} \geq \sum_{(i,j)\in A} w_{ij} z_{ij}^{kl} \qquad \forall k,l \in V; k \neq l \tag{6}$$

$$z_{ij}^{kl} + z_{ji}^{kl} \leq y_{ij} \qquad \forall (i,j) \in E; k,l \in V \tag{7}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in F, k \in V \tag{8}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{9}$$

$$z_{ij}^{kl} \geq \{0,1\} \qquad \forall (i,j) \in A, k \neq l \tag{10}$$

The constraints (2) and (3) are the standard assignment constraints ensuring that only one activity is localized in each area. The shortest paths value between k and l are catched using flow variables z_{ij}^{kl}, constrained by the standard flow conservation constraints (8). The constraints (7) ensure that only one arc $(i,j) \in A$ or $(j,i) \in A$ is used in the shortest path between k and l if

and only if the corresponding edge $(i,j) \in E$ belongs to the optimal network. The constraints (8) and (9) are the integrality constraints. Let

$$S = \{x \in \{0,1\}^{n\times n} \; satisfying \; (2) \; , \; (3) \; , \; (8)\}$$

and

$$S^{'} = \{(y,d) \in \{0,1\}^{n\times n} \times \mathbb{R}^{n\times n} \; satisfying \; (4) \; , \; (6) \; , \; (7) \; , \; (9) \; , \; (10)\}.$$

(ONQAP) may be rewritten as follows :

$$(ONQAP) : \min_{x\in S} \min_{(y,d)\in S'} q(x,y,d) \tag{11}$$

This shorter writing allows to derive a simple observation that leads to a metaheuristic scheme detailed in Section 4.

4 A Greedy Heuristic Method

One may observe using the formulation (11) that

$$\min_{x\in S} \min_{(y,d)\in S'} q(x,y,d) = \min_{(y,d)\in S'} \min_{x\in S} q(x,y,d). \tag{12}$$

As a consequence, we can derive the following result.

Theorem 1 *If x^*, y^* and d^* are optimal solutions then it is necessary that*

$$(y^*,d^*) \in \underset{(y,d)\in S'}{argmin}\, q(x^*,y,d)$$

and

$$x^* \in \underset{x\in S}{argmin}\, q(x,y^*,d^*)$$

where $argmin(q)$ denotes the set of optimal solutions of the function q.

The greedy heuristic method consists in finding a local optimal triplet $(\overline{x},\overline{y},\overline{d})$ verifying the necessary conditions described above. Such a point may be found using the algorithm 1.

At each iteration of the loop in algorithm 1, an assignment is fixed and the optimal network design problem is solved (line 14). The resulting network solution is then fixed and the best assignment corresponding to this solution is

1 x^0 : an initial assignment;
2 $(y^0, d^0) = \underset{(y,d)\in S'}{argmin}\, q(x^0, y, d)$;
3 $old = q(x^0, y^0, d^0)$;
4 $p = 0$;
5 stop = false ;
6 **while** *(! stop)* **do**
7 $\quad x^{p+1} = \underset{x\in S}{argmin}\, q(x, y^p, d^p)$;
8 $\quad new = q(x^{p+1}, y^p, d^p)$;
9 $\quad$ **if** $(old \leq new)$ **then**
10 $\quad\quad$ stop = true ;
11 $\quad$ **end**
12 $\quad$ **else**
13 $\quad\quad$ old = new ;
14 $\quad\quad$ $(y^{p+1}, d^{p+1}) = \underset{(y,d)\in S'}{argmin}\, q(x^{p+1}, y, d)$;
15 $\quad\quad$ p = p + 1;
16 $\quad$ **end**
17 **end**

Algorithm 1: Greedy Heuristic with initial assignment

computed (line 7), and so on. It can be observed that instead of starting by an initial assignment x^0 (line 1), one may start by an initial network y^0. Notice also that fixing y^0 gives automatically unique values of the shortest paths thus defining the pair (y^0, d^0). Hence, starting with this pair leads to another (symetrical) variant of the algorithm above where x and y are systematically switched. However, in any variant, finding the optimal solution of an optimal network design problem (line 14), as well as of a quadratic assignment one (line 7), is necessary. It is well known that both problems are NP-Hard [10] by which, in practice, we do not compute exact solutions but also some heuristic ones using two standard heuristic methods detailed below.

4.1 Deletion method for the network design problem

The optimal network problem is solved at each iteration using a deletion method associated with a shortest path algorithm (see [9] for more details on deletion method). Let $G = (V, E)$ be the graph in which the network design problem must be solved. Without loss of generality, we can assume (by renumbering if necessary) that :

$$E = \{e_1, e_2, ..., e_m\} \; with \; b_{e_1} \geq b_{e_2} \geq ... \geq b_{e_m}$$

Let x be a given (fixed) assignment, and let us notice $ShortestPathAlgorithm(G)$ an algorithm giving all pair shortest path values on G (for instance the Floyd-Warshall algorithm [8]). We assume that if no path exist between two nodes k and l then $d_{kl} = +\infty$. Using this notation, the algorithm 2 can be deduced.

```
1  y_e = 1, ∀ e ∈ E;
2  d = ShortestPathAlgorithm(G) ;
3  i = 1 ;
4  old = q(x, y, d) ;
5  while i ≤ m do
6  |  E = E\{e_i}, y_{e_i} = 0 ;
7  |  d = ShortestPathAlgorithm(G) ;
8  |  new = q(x, y, d) ;
9  |  if (new < old) then
10 |  |  old = new ;
11 |  end
12 |  else
13 |  |  E = E ∪ {e_i}, y_{e_i} = 1 ;
14 |  end
15 |  i = i + 1 ;
16 end
```

Algorithm 2: Deletion algorithm

The deletion algorithm consists in deleting the edges of G, one by one, in a given order. For each deletion, we check if the overall cost decreases or not. If it decreases, the deletion is accepted (and fixed in a greedy fashion), otherwise the edge is restored. In the algorithm above, the deletion is made in the decreasing order of the edge construction cost, and the Flyod-Warshall [8] algorithm is used for the shortest path computations.

4.2 2-opt neighbourhood search heuristic for QAP

QAP is a well-studied problem for which many meta-heuristic schemes have been developped and tested. Meta-heuristics algorithms for QAP include : local search methods, simulated annealing (Wilhelm and Ward [36]), tabu search (Battiti and Tecchiolli [37]), iterated local search ([38] [39]), and population-based approaches (as memetic and genetic algorithms). One may cite in the class of population-based approaches a recent (2018) contribution of Abdel-Baset el al. [21] dealing with a memetic algorithm using the Whale Optimization Algorithm (WOA) integrated with a Tabu Search. Tabu Search is used to improve the quality of solutions obtained by WA. Fourteen different case studies including 122 test problems are employed for analyzing the performance of the proposed algorithm. The results show that the proposed memetic algorithm finds good near optimal solutions with acceptable computational times. In [22], Abdel-Baset. et al. also presented an Elite Opposition-Flower Pollination Algorithm (EOFPA). The proposed algorithm is tested against a set of benchmarks of QAP from the public QAPLIB Library and compared against the best proposals from the related literatures. In the majority of instances, the results showed better performance than other algorithms of the litterature.

As QAP is just one of the two sub-problems of ONQAP, we choose to implement a much less elaborated method than the two cited above. We use a 2-opt neighbourhood search procedures based on Iterated Local Search (ILS)

hybridized with Tabu Search. ILS is essentially an improved version of Hill-Climbing with Random Restarts. That is, it tries to stochastically hill-climb in the space of the local optima (see [27]).

Let x be a initial assignment solution we wich to improve with our local searcj procedure. The local search procedure is explained starting by one solution x, but in practice n assignments are randomly generated, each one improved as described below. The best solution is retained at the end. According to a specific criterion based on the contribution in the affectation cost, we choose two assignments to permute as follows.

If i_0 is assigned to k_0, and j_0 to l_0 :

$$i.e\ x_{i_0k_0} = x_{j_0l_0} = 1,$$

the contribution of this assignment is evaluated with the values :

$$v(i_0, k_0, j_0, l_0) = c_{i_0k_0} + c_{j_0l_0} + \sum_{(i,k)\in F\times V\setminus(j_0,l_0)} f_{ij_0}w_{kl_0}x_{ik}x_{j_0l_0} + \sum_{(j,l)\in F\times V\setminus(i_0,k_0)} f_{i_0j}w_{k_0l}x_{i_0k_0}x_{jl}$$

With this criterion, the pair of assignments for which the location will be permuted, is the one that maximize the corresponding value. If the new assignment gives a reduction of the cost then the oldest solution is replaced by the permutation otherwise the oldest solution is kept. And so on, until a given number of iterations. As in tabu search heuristics, to avoid cycling the permutations performed are stored in a tabu list (L). By which, in fact, the pair of assignments chosen for permutation must maximize the criteria but also must be not"Tabu". When a solution is not improved a counter (k) is incremented. This counter gives the number of iterations since the last best solution. If it exceeds a given limit value then we consider that the algorithm is stuck on a local solution and we disrupt it by random assignments. The limit value is empirically and randomly chosen in the interval $[n^2/2,\ 2n^2]$. The algorithm implements all the points describe above. In this algorithm,

- $permute(x, i, k, j, l)$ denotes a procedure giving as result a new assignment matrix x where the locations k of i and l of j have been permuted,
- $evaluate(x)$ is the objective function value corresponding to x,
- $disrupt(x)$ is a procedure disrupting randomly x.

With all the greedy procedures described above, an evolutionary heuristic method have been implemented.

Data: n : number of locations and activity, m : number of iterations;
$x = a\ initial\ random\ assignments\ of\ entities\ to\ locations$;
$L = \varnothing$;
$lim = random\ value\ in\ [n^2/2, 2n^2]$;
$k = 0$;
$i = 0$;
while $i < m$ **do**
  $i = i + 1$;
  $BestSolution = x$;
  $(i_0, k_0, j_0, l_0) = \underset{\substack{(i,k,j,l)\ \in\ F\times V\times F\times V \\ (i,k,j,l)\notin L}}{argmax} v(i, k, j, l)$;
  $y = permute(x, i_0, k_0, j_0, l_0)$;
  $L = L \cup \{(i_0, k_0, j_0, l_0)\}$;
  if $evaluate(x) \leq evaluate(y)$ **then**
    $k = k + 1$;
  else
    $x = y$;
    $k = 0$;
  end
  if $lim < k$ **then**
    $x = disrupt(x)$;
  end
end

Algorithm 3: Heuristic for QAP

5 Evolutionary Algorithm and greedy hybridization

In this section, we propose a Genetic Algorithm (GA) in order to improve the solution of the greedy algorithms. Genetic Algorithms (GAs) are one of the most popular heuristic algorithms for solving optimization problems. The GAs are adaptive search techniques initially introduced by Holland [14]. Its name derives from the fact that their operations are similar to the mechanics of genetic models of natural systems. The GAs have been applied to a variety of combinatorial optimization problems (Goldberg [11] and Davis [5]). Many GAs for the QAP have been proposed. Bean [2] describes a genetic algorithm to solve problems whose solutions are specified by permutations. Tate and Smith [34] proposed another GA which uses the problem specific structure and tested it on 11 benchmark instances, due to Nugent et al. [25], of size $5-30$ facilities. This is a fairly direct implementation of the classical GA that do not use any greedy ideas. Fleurent and Ferland [7] describe another GA which uses local search methods to improve the fitness of individuals. They found that using local search methods substantially improves the correctness of the algorithm for the QAP. In our work, we investigate GAs incorporating our greedy heuristics. We present computational results of the algorithm on artificial instances as well as on a real-life big data of the Dakar (Senegal) region.

The GAs imitate the process of evolution on an optimization problem and can be roughly subdivided into the following steps. Each feasible solution of

a problem is composed of chromosomes and genes that must be first encoded. We will see in the subsection 5.1 the encoding choices made for our algorithm. At the beginning of the process, an initial population is created. Some individuals are then selected, among the population, and contribute to the population evolution using essentially two operators. The crossover operator allows two individuals to be crossed in such a way as to give rise to better individuals. And the mutation operator randomly changes some genes to the benefit of the diversity in the population. The successive evolution of the population using these operators provides new individuals in which the best one is chosen as the problem solution at the end of the algorithm.

We detail the choices made in our algorithm for all these steps. In particular, notice that two methods were implemented and compared for generating the initial population. A first one consisting in a pure random generation of the population, and a second one where the assignment solution found by our greedy algorithm is randomly perturbed. These two approaches lead to two algorithms called GA (Genetic Algorithm) and GGG (Greedy Genetic Greedy) for which the numerical experiments show clearly the added value of using the greedy algorithm.

5.1 Encoding scheme

Consider a population consisting of m individuals $x^j = (x^j_{ik}),\ i,k = 1,...,n$ where each x^j denotes the assignment matrix. We define as "chromosome" the (row) vector corresponding to each row of x and as "gene" each component x_{ik}. We thus have n chromosomes and n^2 genes.

$$\begin{array}{c} \qquad\qquad\qquad\qquad n \text{ chromosomes } \downarrow \\ \text{individual } \longrightarrow\ x^j = \overbrace{x^j_{11}\ x^j_{12}\cdots x^j_{ik}\cdots x^j_{nn}} \\ \qquad\qquad\qquad\qquad \text{gene } \uparrow \end{array}$$

$$\text{individual } \longrightarrow\ x^j = \left\{ \begin{array}{rl} x^j_{11}\ x^j_{12}\cdots x^j_{1n} & \leftarrow \text{ chromosome 1} \\ x^j_{21}\ x^j_{22}\cdots x^j_{2n} & \leftarrow \text{ chromosome 2} \\ \qquad\vdots & \\ x^j_{n1}\ x^j_{n2}\cdots x^j_{nn} & \leftarrow \text{ chromosome } n \\ \text{gene } \uparrow & \end{array} \right.$$

5.2 Initial population generation

In genetic algorithm approaches the individuals of the population can be randomly generated or based on solutions provided by other algorithms. What is the best approach between these two is not theoretically clear. One may

think that a huge random population gives an initial good diversity by better covering of the solution space. And that its evolution, with the operators, will lead to better solutions. But solutions based on the random perturbations of other (good) solutions may also help the algorithm to reach, more quickly, solution spaces where an optimal solution can be found.

Two types of initial population have been considered to analyze their impacts in the final solution. The first one is a pure random generation. The second one is derived by exploiting the greedy algorithm of the Section 4. More precisely in the second case, we first run the greedy algorithm and the best solution, here denoted by (x^0, y^0, d^0),will be the first individual of our population. This solution is then randomly perturbed by permutation of the assignment genes. These perturbations lead to $m-1$ other individuals (x^j, y^j, d^j) $(j = 1, \cdots, m-1)$. Recall that the algorithm using the first type of initial population will be called, in the numerical experiments, GA (Genetic Algorithm) and the other using the greedy solution is the GGG (Greedy Genetic Greedy) algorithm.

5.3 Selection

The selection process performs in this paper is called "selection by rank" in the literature [5]. The m individuals computed before are classified in the increasing order of their objectives function values q (also called fitness function). Then we attribute to individuals, a weight that depends on its rank in the sorting step. For an individual (x^j, y^j, d^j) this weight is equal to $W_j = (m - r_j)^a$, with $a = 1.5$, where r_j is the rank of the individual j in the classification. With these weights a probability of survival is determined by $P_j = W_j / \sum_{j=0}^{m-1} W_j$. Two parents are randomly selected for crossover if the sum of their probabilities is greater than a fixed crossover probability. The best individual or parent is always chosen to cross with a random choice of another parent.

5.4 Crossover

We propose a crossover based on permutation of chromosome. Two parents selected will form two children as follows. Let i be the index of a chromosome. We associate to i the following cost :

$$h_i = \sum_{k=1}^{n} c_{ik} x_{ik} + \sum_{k=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{n} f_{ij} d_{kl} x_{ik} x_{jl}.$$

We call "bad chromosome" of the Parent 1 a chromosome i with the highest cost h_i. If i in the parent 1 is bad then it will be replaced by the chromosome i of the parent 2. But before doing this change, for feasibility purpose, we first copy the chromosome i of the parent 1 at the chromosome l of the parent

1 identical to the chromosome i of the parent 2. And we get the first child. The second child is obtained by the same procedure but by exchanging parent roles. Hence, the child 1 is the parent 1 with the bad chromosome replaced. And the child 2 is the parent 2 with the copied bad chromosome of the parent 1. An example of this process, with $n = 4$, is given in figure 1. We can see that

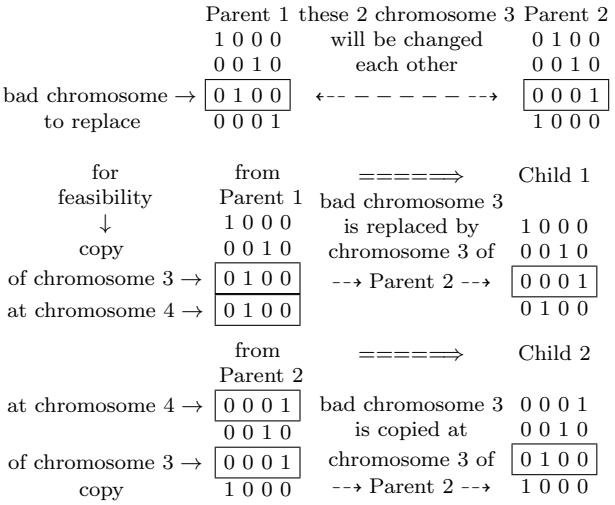

Fig. 1 Crossover

the crossover performed is like a permutation of a chromosome of a parent based on a chromosome of an other parent. And applied to two parents give two children.

5.5 New generation

The next generation will be obtained by replacing only individuals of older population, with survival probability lower than a fixed threshold, by the childrens obtained as above. Best individuals with probability greater than the threshold are always conserved in the next generation.

5.6 Mutation

A random individual mutes if a random probability chosen between 0.00001 and 0.1, is lower than a fixed probability of mutation. We fix the probability of mutation to 0.015. The mutation operator is a permutation of two random chromosomes.

5.7 Genetic Algorithm

Using all the operators described above leads to the algorithm 4.

Data: n number of locations and activity, m number of individuals, N number of generations.
Initial population : m individuals x^j, $j = 0, \cdots, m-1$ `population` are generated as in the Subsection5.2;
evaluate fitness of each individual $q(x^j, y^j, d^j)$ `selection`;
i ⟵0;
while $i < N$ **do**
    i ⟵i+1;
    Selection() `selection of parents to cross`;
    Crossover() `construction of childrens`;
    Population() `construction of the new generation`;
    Mutation();
end

Algorithm 4: Genetic Algorithm heuristic

5.8 Greedy Genetic hybridization

This algorithm (see figure 2) is an extension of the genetic algorithm. As explained in the beginning of Section 5, its initial population is obtained using the best solution of the greedy algorithm as initial assignment and network. Standard operators explained above are then applied in this population - leading to a new population where the best individual is chosen. Then, the resulting solution is used as initial assignments (or network) of the greedy algorithm. Despite the fact that iterating several times between Greedy, Genetic Algorithm and Greedy is possible, we limit ourselves in the numerical experiments to one iteration because of time limitations.

6 Numerical simulations

This section deals with the numerical results obtained using all the algorithms described above. The tests have been performed on instances deriving from academic sources, as well as on a real-life big data on Dakar city (Senegal). Two types of academic instances have been used. The first one comes from Los [18] (here denoted by "Los") and the second one from the QAPLIB[4] benchmark. The QAPLIB problems are the instances of Nugent, Vollman, and Ruml [25] (here denoted "Nug"), Elshafei [6]("Els"), Hadley, Rendl, Wolkowicz [13]("Had"), Krarup and Pruznan [15]("Kra"), Li and Pardalos [17]("Lipa"), Roucairol [28]("Rou"), Scriabin and Vergin [30]("Scr"), Skorin-Kapov [32]("Sko"), Taillard [33]("Tai"), Wilhelm and Ward [36]("Wil").

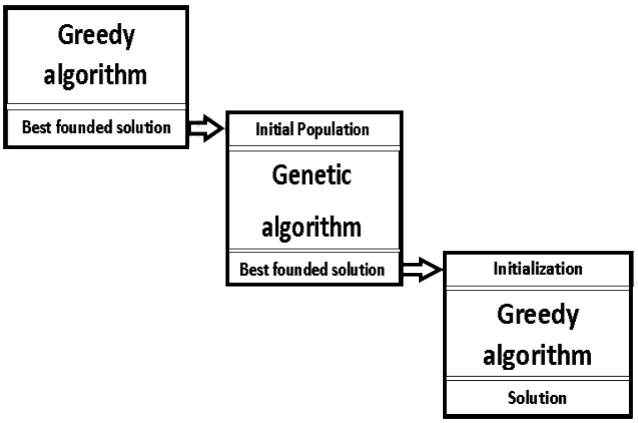

Fig. 2 Greedy Genetic Greedy hybridization (GGG)

Each instance gives, as input for our algorithms, the flows between the entities and the distances between the locations. The location costs are taken to be null in these experiments. The construction cost of any edge (k, l) of the network has been derived by multiplying the distance d_{kl} by 10 in the case of the instances "Nug", "Had", "Wil", "Tho", "Scr", by 100 for instances "Els", "Lipa", "Rou", "Tai", and by 1000 for "Kra". In the table where the results are reported, the acronym of the instance will be always followed by the size of the problem. For instance, Los10 will denote an instance coming from Los paper of size 10. The tests are performed on intel(R) core i3-2348M CPU @ 2.30GHZ 2.30GHZ with 4Gb of ram memory.

All algorithms 1, 3, 4, and GGG showed in figure 2, have stochastic nature because of the initial starting points or random choices within the algorithms. Thus, to increase the ability to explore a larger solution space, we ran these algorithms many times and present the best solution founded. All algorithms implies also some parameters fixed as follows. The maximal number of iterations (m) of the algorithm 3 has been fixed to 500 for instances of size up to 50, and to 50 for instances of size between 51 to 100 . The initial population size of the genetic algorithm 4 is 10 for instances of size 100 and 20 for others, for all the generations. And for each instance the number of generations N is 70 for the genetic algorithm 4, and 70 for the GGG algorithm 2.

6.1 Academic instances

To our knowledge, except the two instances of the Los paper, there are no other benchmarks in the literature with which we can evaluate our heuristic solutions by comparing different solution techniques for our problem. The academic instances, derived from QAPLIB ones, are used for the first time. Thus, to have a meaningful evaluation of our methods, we choose to compare our

results with one of the leading optimization software tools LocalSolver (version 7.5). LocalSolver combines several algorithmic techniques making it possible to tackle large-scale combinatorial and / or numerical optimization problems in very short times. If a maximal resolution time to solve a problem is specified, LocalSolver will not necessarily reach the optimal value but numerical experiments show that, in comparison to other standard codes (such as Gurobi or Cplex), LocalSolver can find (and often outperform) very good approximations. Thus, for each instance, the mathematical formulation of (ONQAP) has been implemented in LocalSolver, and solved with a maximal processing time of 1800 seconds for instances of size $n \leq 50$, 2000 seconds for instances of size $n = 90$, and 2500 seconds for instances of size $n = 100$. The best solution values are then compared them with the values provided by our heuristics. The results are reported in table 1.

UB-LS in this table corresponds to the value found by LocalSolver while the other columns concern our heuristics. Greedy(x) (resp. Greedy (y)) algorithm is the greedy algorithm where the assignments (resp. network) have been fixed at the beginning. The column GA reports the results of the standard Genetic Algorithm where the initial population is random and the different operators detailed above are applied. It has to be compared with the column GGG where the results of the hybridization with our greedy algorithm are reported. This comparison gives an evaluation of the benefit of the greedy algorithm. When our value is better than the LocalSolver one we indicate it by the sign "**", while "*" means that we have the same value. In table 2, we report permutations corresponding to the best solutions and in table 3, we report the processing times. In figure 3, we show graphically the processing times of all algorithms. Except two instances (Nug22 and Sko42), at least one of our upper bounds is always better than the LocalSolver solution. Notice that our results depend on the simulation parameters. One can obtain better result by choosing other values for the parameters, like the number of iterations of the heuristic of QAP, the number of population and generation of the genetic algorithm, etc. But the processing time increases with the values of these parameters. The algorithm GGG gives, on average, the best results, with computational times ranging from 2 to 1531 seconds. In comparison, LocalSolver needs between 2 seconds and 2500 seconds to find its best solutions. Nevertheless, the performance of our heuristic algorithms, in comparison to LocalSolver, should be balanced with the fact that Localsolver runs directly on the formulation (ONQAP). Another formulation on the same problem may lead to better results.

Data	UB-LS	Greedy(x)	Greedy(y)	GA	GGG
Els19**	1.67760e+07	1.67657e+07	1.74978e+07	1.72381e+07	1.67393e+07
Had12*	1932	1932	1932	1932	1932
Had16*	4120	4120	4120	4120	4120
Had20*	7422	7422	7422	7472	7422
Kra32**	2.66502e+06	2.24233e+06	2.24256e+06	2.27793e+06	2.23936e+06
Los14*	4.03755e+07	4.03875e+07	4.03875e+07	4.03875e+07	4.03755e+07
Los16*	5.85162e+07	5.85525e+07	5.86338e+07	5.92312e+07	5.85162e+07
Nug12*	892	892	892	892	892
Nug14*	1398	1398	1398	1398	1398
Nug15*	1570	1570	1570	1594	1570
Nug16a*	2070	2070	2070	2082	2070
Nug17**	2220	2228	2220	2228	2200
Nug18*	2468	2470	2470	2484	2468
Nug20**	3202	3190	3190	3262	3190
Nug22	4154	4200	4200	4200	4194
Nug25**	4554	4540	4540	4774	4538
Nug30**	7200	7132	7124	7474	7124
Tho30*	150914	156086	154942	171310	150914
Tho40**	252298	252302	252242	279696	252242
Sko42	17452	17780	17816	17900	17558
Sko100**	194962	171308	173272	174620	170230
Tai100b	1.31184e+09	1.48512e+09	1.36733e+09	1.7238e+09	1.36546e+09
Scr20**	114600	114250	114670	152698	114250
Rou20	243854	299416	301146	312626	298708
Lipa40b	584049	586510	585259	608121	584791
Lipa50b**	1.28929e+06	1.29008e+06	1.28816e+06	1.34308e+06	1.2863e+06
Lipa90a**	479110	467935	468671	468086	467699
Wil50	62936	65162	65092	66722	64628
Wil100**	333266	293882	293910	298188	292376

Table 1 Objective values : the instances with "*" mean we obtain same result with Local-Solver and with "**" mean we obtain better result than LocalSolver.

Data	Permutation of UB-LS	Permutation of our best result
Els19**	9, 10, 7, 19, 11, 14, 4, 5, 6, 12, 17, 13, 18, 8, 15, 16, 1, 2, 3	9, 10, 7, 19, 11, 14, 4, 17, 6, 12, 13, 5, 18, 8, 15, 16, 1, 2, 3
Had12*	3, 10, 11, 2, 12, 5, 6, 7, 8, 1, 4, 9	3, 10, 11, 2, 12, 5, 6, 7, 8, 1, 4, 9
Had16*	9, 4, 16, 1, 7, 8, 6, 14, 15, 11, 12, 10, 5, 3, 2, 13	9, 4, 16, 1, 7, 8, 6, 14, 15, 11, 12, 10, 5, 3, 2, 13
Had20*	8, 15, 16, 14, 19, 6, 7, 17, 1, 12, 10, 11, 5, 20, 2, 3, 4, 9, 18, 13	8, 15, 16, 14, 19, 6, 7, 17, 1, 12, 10, 11, 5, 20, 2, 3, 4, 9, 18, 13
Kra32**	16, 7, 9, 17, 21, 13, 20, 8, 2, 4, 15, 14, 6, 3, 19, 28, 31, 27, 18, 1, 23, 30, 32, 12, 25, 24, 11, 10, 29, 26, 5, 22	7, 8, 22, 20, 2, 3, 5, 9, 17, 11, 6, 12, 16, 1, 4, 13, 23, 21, 31, 10, 29, 25, 19, 32, 26, 30, 27, 15, 18, 14, 24, 28
Los14*	14, 6, 4, 10, 1, 9, 13, 12, 8, 5, 2, 11, 7, 3	14 , 6, 4, 10, 1, 9, 13, 12, 8, 5, 2, 11, 7, 3
Los16*	14, 1, 8, 7, 13, 16, 4, 9, 2, 11, 10, 12, 15, 3, 6, 5	14, 1, 8, 7, 13, 16, 4, 9, 2, 11, 10, 12, 15, 3, 6, 5
Nug12*	12, 11, 9, 3, 4, 7, 8, 1, 5, 6, 10, 2	12, 11, 9, 3, 4, 7, 8, 1, 5, 6, 10, 2
Nug14*	9, 8, 13, 2, 1, 11, 7, 14, 3, 4, 12, 5, 6, 10	9, 8, 13, 2, 1, 11, 7, 14, 3, 4, 12, 5, 6, 10
Nug15*	12, 5, 15, 6, 10, 11, 7, 14, 3, 4, 9, 8, 13, 2, 1	12, 5, 15, 6, 10, 11, 7, 14, 3, 4, 9, 8, 13, 2, 1
Nug16a*	9, 14, 15, 16, 3, 10, 12, 8, 11, 6, 5, 7, 1, 4, 13	9, 14, 15, 16, 3, 10, 12, 8, 11, 6, 5, 7, 1, 4, 13
Nug17**	9, 14, 12, 11, 16, 3, 2, 15, 8, 4, 6, 5, 7, 1, 17, 13	9, 14, 12, 11, 16, 10, 2, 15, 8, 4, 3, 5, 7, 1, 17, 6, 13
Nug18**	9, 14, 12, 11, 16, 10, 2, 15, 8, 4, 3, 18, 7, 1, 17, 6, 13, 5	9, 14, 12, 11, 16, 10, 2, 15, 8, 4, 3, 18, 7, 1, 17, 6, 13, 5
Nug20**	17, 4, 11, 19, 16, 20, 8, 15, 2, 18, 13, 7, 12, 14, 9, 6, 5, 1, 10, 3	9, 3, 10, 14, 18, 16, 11, 12, 2, 4, 13, 8, 20, 15, 19, 6, 1, 7, 5, 17
Nug22	5, 6, 12, 10, 7, 1, 22, 11, 8, 15, 17, 2, 13, 21, 9, 16, 3, 18, 19, 4, 14, 20	2, 21, 9, 7, 3, 1, 19, 8, 20, 17, 5, 13, 12, 16, 11, 22, 18, 4, 14, 15
Nug25**	5, 11, 20, 15, 22, 2, 25, 8, 9, 1, 18, 3, 16, 6, 19, 24, 4, 21, 7, 21, 10, 12, 17, 14, 13	12, 24, 18, 11, 5, 17, 21, 16, 25, 2, 4, 14, 3, 8, 20, 23, 7, 6, 9, 15, 13, 10, 19, 1, 22
Nug30**	5, 2, 21, 13, 6, 28, 29, 16, 9, 7, 12, 25, 4, 30, 19, 11, 10, 26, 20, 3, 8, 22, 1, 24, 14, 27, 18, 23, 15, 17	15, 23, 11, 30, 27, 4, 17, 18, 8, 16, 3, 14, 1, 22, 7, 19, 29, 20, 26, 6, 10, 9, 21, 28, 24, 12, 25, 13, 2, 5
Tho30*	11, 7, 14, 16, 23, 21, 22, 15, 18, 20, 3, 5, 24, 4, 17, 8, 26, 13, 6, 12, 29, 19, 1, 27, 2, 28, 30, 25, 10, 9	11, 7, 14, 16, 23, 21, 22, 15, 18, 20, 3, 5, 24, 4, 17, 8, 26, 13, 6, 12, 29, 19, 1, 27, 2, 28, 30, 25, 10, 9
Tho40**	25, 27, 8, 13, 21, 23, 24, 19, 36, 26, 35, 32, 37, 28, 33, 4, 34, 12, 29, 16, 30, 15, 6, 7, 17, 31, 11, 10, 14, 2, 1, 40, 3, 38, 5, 22, 9, 20, 39, 18	17, 31, 29, 10, 11, 12, 34, 3, 25, 35, 21, 8, 6, 14, 40, 36, 5, 27, 22, 20, 15, 1, 9, 7, 26, 32, 16, 24, 33, 30, 39, 18, 38, 13, 23, 37, 4, 28, 2, 19
Sko42	40, 10, 6, 16, 11, 30, 12, 28, 33, 8, 41, 34, 18, 9, 39, 32, 3, 14, 13, 19, 26, 24, 37, 23, 31, 17, 25, 27, 7, 20, 0, 36, 38, 1, 21, 5, 15, 2, 4, 29, 35, 22	27, 10, 42, 5, 17, 41, 25, 13, 31, 19, 32, 12, 7, 11, 2, 39, 30, 18, 34, 36, 6, 28, 14, 35, 26, 20, 3, 29, 22, 15, 40, 37, 38, 24, 23, 33, 4, 9, 21, 1, 16, 8

Sko100**	6, 87, 80, 42, 47, 78, 81, 10, 77, 96, 58, 54, 49, 48, 88, 57, 32, 4, 13, 33, 65, 26, 82, 50, 37, 53, 70, 94, 20, 74, 23, 36, 18, 46, 60, 72, 99, 11, 8, 44, 92, 93, 7, 71, 90, 55, 62, 40, 17, 85, 3, 15, 1, 66, 56, 21, 35, 89, 12, 75, 41, 79, 38, 84, 98, 63, 14, 34, 31, 83, 29, 2, 24, 16, 67, 91, 39, 69, 19, 76, 43, 68, 9, 86, 64, 25, 95, 97, 27, 61, 5, 0, 30, 28, 59, 73, 45, 22, 51, 52	23, 42, 7, 37, 17, 12, 70, 27, 31, 62, 19, 72, 96, 21, 89, 50, 60, 4, 63, 5, 90, 68, 93, 28, 11, 43, 46, 52, 56, 44, 32, 24, 97, 18, 47, 41, 26, 48, 25, 76, 53, 79, 91, 6, 3, 2, 57, 8, 82, 16, 65, 87, 34, 13, 83, 49, 22, 81, 74, 80, 54, 85, 1, 15, 94, 75, 98, 59, 69, 95, 45, 71, 67, 64, 58, 0, 61, 20, 30, 10, 39, 40, 36, 77, 92, 84, 66, 73, 38, 35, 33, 29, 9, 99, 14, 88, 51, 78, 55, 86
Tai100b	77, 47, 66, 82, 64, 24, 91, 83, 92, 58, 89, 40, 12, 63, 6, 70, 76, 52, 75, 25, 85, 55, 96, 78, 13, 15, 36, 49, 5, 69, 10, 72, 29, 39, 59, 57, 8, 68, 23, 14, 45, 74, 48, 22, 4, 31, 62, 42, 80, 17, 50, 46, 43, 21, 84, 18, 33, 90, 60, 61, 95, 97, 93, 65, 3, 56, 19, 2, 94, 51, 26, 88, 41, 54, 0, 79, 44, 30, 98, 7, 53, 28, 86, 73, 38, 11, 99, 9, 1, 71, 87, 35, 67, 37, 81, 32, 20, 34, 27, 16	33, 28, 92, 89, 24, 82, 86, 40, 13, 42, 97, 49, 8, 99, 74, 85, 66, 98, 88, 39, 60, 81, 75, 53, 12, 79, 83, 14, 15, 41, 17, 45, 25, 27, 78, 76, 62, 16, 50, 9, 20, 68, 32, 71, 11, 44, 18, 87, 95, 1, 19, 30, 47, 94, 80, 0, 36, 70, 58, 6, 73, 61, 23, 65, 64, 29, 2, 4, 90, 77, 26, 43, 37, 31, 5, 96, 59, 93, 22, 7, 54, 63, 52, 21, 69, 38, 57, 56, 48, 55, 34, 67, 3, 72, 35, 46, 10, 84, 51, 91
Scr20**	16, 17, 4, 13, 18, 12, 14, 15, 0, 5, 11, 10, 3, 1, 2, 7, 6, 8, 9, 19	19, 8, 9, 4, 13, 0, 12, 16, 11, 10, 2, 6, 15, 14, 18, 3, 7, 1, 5, 17
Rou20	15, 9, 0, 14, 10, 1, 19, 11, 6, 17, 13, 12, 7, 3, 18, 2, 8, 5, 4, 16	6, 2, 8, 14, 18, 19, 0, 5, 7, 9, 16, 1, 13, 10, 3, 15, 11, 17, 4, 12
Lipa40b	27, 14, 19, 2, 30, 8, 39, 6, 33, 0, 25, 22, 4, 5, 35, 38, 31, 9, 23, 29, 18, 15, 28, 13, 12, 17, 11, 20, 1, 7, 26, 24, 16, 3, 34, 10, 32, 36, 37, 21	11, 14, 16, 31, 24, 23, 19, 37, 9, 12, 26, 17, 13, 8, 6, 36, 35, 7, 1, 29, 20, 25, 21, 27, 32, 28, 39, 2, 18, 5, 0, 4, 15, 22, 3, 10, 34, 30, 38, 33
Lipa50b**	19, 9, 40, 5, 31, 16, 27, 34, 45, 25, 43, 44, 1, 4, 39, 46, 26, 37, 28, 14, 42, 8, 6, 11, 3, 15, 29, 36, 23, 30, 12, 7, 47, 49, 13, 22, 17, 35, 21, 32, 2, 48, 10, 41, 18, 20, 24, 38, 0, 33	10, 44, 47, 36, 28, 21, 12, 11, 41, 30, 43, 42, 26, 24, 33, 20, 45, 46, 25, 2, 32, 8, 13, 14, 15, 29, 19, 49, 40, 22, 31, 37, 18, 0, 39, 7, 23, 27, 3, 35, 4, 1, 16, 48, 9, 5, 17, 34, 38, 6
Lipa90a**	48, 57, 41, 62, 31, 3, 6, 80, 74, 54, 67, 66, 25, 82, 85, 16, 46, 33, 45, 47, 89, 23, 59, 29, 75, 5, 84, 8, 51, 9, 17, 37, 24, 64, 20, 1, 19, 70, 30, 32, 27, 0, 13, 18, 4, 10, 2, 36, 63, 40, 44, 77, 12, 53, 88, 78, 87, 58, 26, 15, 61, 22, 73, 34, 79, 11, 60, 7, 55, 52, 72, 71, 68, 50, 83, 76, 86, 14, 42, 35, 28, 49, 81, 38, 65, 69, 43, 21, 56, 39	84, 60, 42, 52, 73, 22, 15, 72, 21, 16, 78, 55, 30, 77, 28, 12, 23, 67, 14, 82, 46, 27, 10, 88, 41, 44, 13, 32, 4, 34, 58, 47, 48, 9, 79, 80, 76, 25, 43, 5, 65, 31, 3, 83, 62, 61, 59, 24, 35, 63, 17, 11, 2, 33, 56, 26, 70, 40, 36, 19, 89, 74, 0, 6, 68, 85, 1, 20, 18, 86, 53, 45, 75, 69, 57, 51, 39, 49, 81, 37, 7, 38, 66, 29, 71, 87, 8, 50, 54, 64
Wil50	0, 38, 5, 45, 1, 34, 19, 17, 9, 31, 14, 39, 18, 10, 15, 21, 44, 29, 6, 48, 11, 30, 7, 24, 28, 8, 49, 33, 42, 32, 12, 22, 4, 47, 41, 46, 35, 25, 36, 2, 23, 43, 37, 3, 13, 40, 16, 20, 26, 27	19, 42, 43, 17, 37, 47, 11, 33, 10, 27, 31, 4, 14, 28, 41, 25, 6, 2, 30, 0, 46, 9, 32, 45, 40, 20, 7, 8, 48, 38, 21, 34, 5, 1, 29, 35, 3, 23, 15, 36, 24, 18, 13, 16, 26, 49, 12, 39, 22, 44

Wil100**	80, 35, 58, 54, 45, 85, 18, 3, 91, 92, 86, 16, 75, 41, 90, 94, 95, 23, 39, 2, 98, 61, 56, 38, 17, 34, 55, 81, 47, 96, 78, 89, 99, 40, 87, 25, 22, 36, 57, 28, 44, 5, 30, 84, 53, 20, 26, 50, 79, 60, 73, 76, 48, 52, 29, 62, 0, 66, 46, 10, 64, 4, 88, 69, 49, 14, 43, 97, 37, 13, 15, 77, 83, 93, 68, 74, 59, 11, 72, 82, 65, 6, 67, 21, 12, 33, 71, 8, 9, 63, 24, 19, 32, 1, 31, 42, 7, 51, 27, 70	44, 71, 8, 59, 78, 88, 30, 80, 29, 50, 13, 43, 46, 9, 27, 14, 40, 54, 23, 84, 92, 73, 66, 35, 51, 55, 18, 37, 42, 10, 6, 34, 38, 17, 4, 94, 49, 2, 67, 93, 41, 65, 16, 3, 19, 99, 11, 61, 85, 58, 57, 83, 95, 76, 90, 98, 87, 5, 0, 75, 45, 60, 68, 47, 31, 64, 69, 91, 32, 28, 24, 22, 25, 48, 52, 97, 74, 62, 21, 96, 1, 86, 20, 79, 36, 33, 82, 77, 63, 39, 89, 53, 7, 81, 72, 56, 12, 15, 70, 26

Table 2 Activities location : permutation associated to the solution of LocalSolver and permutation corresponding to the best solution obtained with our algorithms.

Data	t_{UB-LS}	$t_{Greedy(x)}$	$t_{Greedy(y)}$	t_{GA}	t_{GGG}
Els19	1015	7	6	10	10
Had12	2	1	1	2	2
Had16	31	4	3	2	5
Had20	388	9	7	5	13
Kra32	829	35	28	28	56
Los14	146	2	2	3	4
Los16	1633	4	4	6	5
Nug12	129	1	1	2	2
Nug14	592	2	2	3	4
Nug15	243	3	3	4	5
Nug16a	546	4	3	3	5
Nug17	591	5	5	6	7
Nug18	79	6	5	5	8
Nug20	1097	10	10	6	8
Nug22	399	12	11	8	11
Nug25	753	26	26	13	19
Nug30	119	49	50	19	46
Tho30	504	25	22	25	36
Tho40	389	191	185	93	125
Sko42	128	214	210	120	150
Sko100	2373	1105	1065	2492	1341
Tai100b	2500	1417	1274	1861	1531
Scr20	454	4	3	10	9
Rou20	912	4	4	9	8
Lipa40b	968	72	65	155	55
Lipa50b	687	164	165	321	167
Lipa90a	1997	996	679	2363	1509
Wil50	1260	497	530	1895	958
Wil100	1692	813	770	1943	1359

Table 3 Processing times (in second)

6.2 Case study : Dakar

The heuristic algorithms have been also tested for a real-world case on Dakar (Senegal). Different data sources have been exploited for this case study. Origin-Destination flows were built from the mobile phone data used in the d4d challenge [12]. The d4d challenge (Data For Development) is an innovation challenge on ICT Big Data for the purposes of societal development. The second edition, in which we have participated, was organized in 2015 by the French telecommunication group Orange and its partner in Senegal, Sonatel. Sonatel and the Orange Group have provided anonymous data, extracted from the mobile phone network in Senegal. One family of these data contains trajectories of millions of agents (or customers), in Senegal, in 2014 which is compiled as follows. Each time an Orange/Sonatel customer makes or receives a call with its mobile phone, this call is in practice possible by connection through relay antennas (or mobile phone antennas) performing the message transmissions. If the customer does not move, its closest relay antennas will detect the call request. Whereas, if he moves, different (closest) antennas met

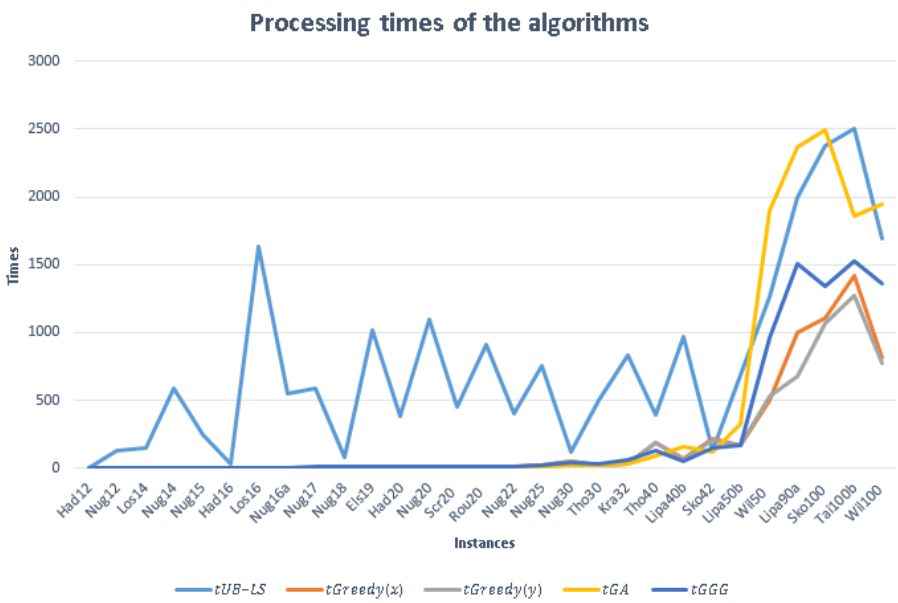

Fig. 3 Processing times of algorithms

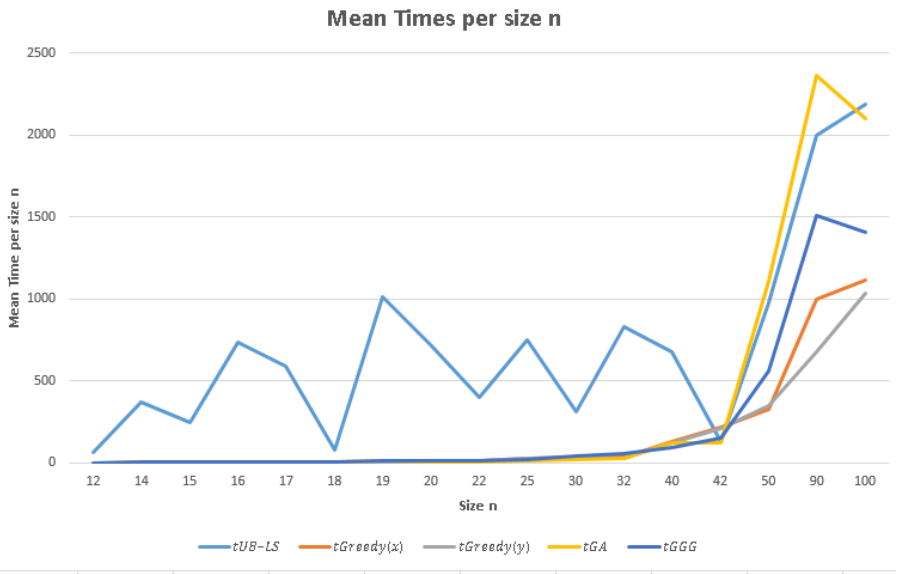

Fig. 4 Processing times per size of algorithms

during his trip may be used. Thus, for a customer, the registration (in files) of the relay antennas used in a given period can be useful to re-build customer trip trajectories. The d4d big data are organized in 25 files, each one containing the list of visited mobile phone antennas, over a period of 2 weeks, for 320,000 individuals, anonymized, and randomly selected. For each file, the sample of 320,000 individuals is renewed to ensure anonymity. For a given week in 2014, by analyzing the corresponding file, it is thus possible to deduce the total amount of trips between any pair of antennas. However, we are not interested in the antennas, where an agent has been detected, but on the activities performed around them because our goal is to find good locations for them. Our case study is limited to the file corresponding to the first 2 weeks of 2014, and for this file we limit ourself to agent trips in the Dakar region (not in the entirety of Senegal). With this limitation, we exploit the file as follows. The antennas have been first grouped in clusters to solve the problem of antenna co-localization. Co-localization means that several mobile phone antennas are located nearby. It follows that from time to time a call can be supported over a short period by several antennas even if the agent calling (receiving a call) does not move. In other words, detecting an agent with two antennas close to each other does not mean that the agent performed a trip which should be considered as a flow. Thus to have more precise information on real movements, we perform a hierarchical ascendant clustering of the antennas. This algorithm provides clusters of antennas where, in each cluster, the maximal distance between any pair of antennas does not exceed a given threshold. Thus an agent successively detected by two antennas in the same cluster will be then considered as motionless. Using the d4d file, we implement an algorithm giving for each pair of antennas cluster the total amount of moves. This corresponds precisely to the O-D matrix describing flows between the antenna clusters. Now, in addition to the D4D file, we have also retrieved and exploited Open Street Map (OSM) resources, in particular the OSM data for Senegal provided by the Humanitarian OpenStreetMap Team (HOT). Using the tool Osmosis and the open source Geographical System Quantum GIS, we separately construct a list of Dakar amenities (activities) with their geographical locations. As we also have mobile phone antenna locations, each amenity has been assigned to its closest antenna. As a consequence, the union of activities assigned to each antenna of a cluster also defines a cluster of activities. We assume that if an agent is detected moving from antenna cluster O to antenna cluster D, it means that he moves from an activity of the cluster of activities associated to O to another activity of the activities associated to D. Our problem is then to find where the cluster of activities should be located and what should be the transportation links between them to optimize the overall sum (for all agents) of the trip costs between the activity clusters.

For one flow going from O to D, another important data item is the evaluation of the trip cost. It is assumed in our model that this cost corresponds to the distance between O and D. But, there are at least two ways to compute this distance that may lead to different solutions. Geographical distance is the

simplest evaluation but it assumes that it is possible to build a direct road from O and D which may be not possible in practice because of land constraints. Another way is to consider the current transportation network and to compute the shortest road distance between the median antennas of O and D. Doing this removes the road feasibility problem since the distances correspond here to real-life roads. However, in turn, the optimal network computed by the algorithm will be in fact a sub-network of the current one. We choose in our experiments this second approach.The distances have been computed using Google Maps APIs on the real transportation network.

It is also important at this point to focus on the motivations of these experiments and their practical applicability. We obviously know that, in practice, permuting two clusters of activities, as well as building a new transportation line, may be difficult and in certain cases impossible. Of course, in the real-life, the city design cannot be modified as in the electronic game SimCity[1]. Activities are not puzzles or chess pieces that urban planners may easily move. Our work consists in giving models whose optimal solutions give an idea of the "best" activity locations and transportation network taking into account observed O-D flows ("best" in the sense of an optimization function). Knowing this "ideal" city, planners can then compare it with the current situation, and take any feasible decisions to reach it. Notice that in our experiments, all activity clusters may move but simple constraints may be added to allow the movements of a given subset of activities. An urban planner can then use the models to just analyze suitable locations of k (fixed) activity clusters. In our experiments, 43 antenna clusters have been built. Given that an activity cluster is associated with each antenna cluster, we also have 43 activity clusters for which we want to find the best location among them, and the best links connecting them.

Map 5 gives the geographical position of the "median" antenna of each cluster. The "median" antenna is defined as the antenna minimizing the sum of the distance to the others. That is in each cluster the geographical position of the optimal solution of a 1-median location problem. Each cluster (of antennas or activities) is identified by a number. Initially, the clusters and the areas (locations) where they are located have the same indices. New locations for the activity clusters correspond to a permutation of these clusters.

Using informations provided by Senegalese maps and geographical public offices (DGTC : Direction des Travaux Gographiques et Cartographiques, CETUD: Conseil Excutif des Transports Urbains de Dakar), we assume that between areas the possible road to be constructed is monolayer with width of $7m$. It is estimated (by these offices) that 1 km of this type of road costs $C_M = 33,246,500\ F\ CFA$. By which the cost to construct an edge (k, l) between the areas k and l is $b_{kl} = C_M * w_{kl}$.

For the location cost of activities, we consider two situations :

[1] https://www.ea.com/games/simcity

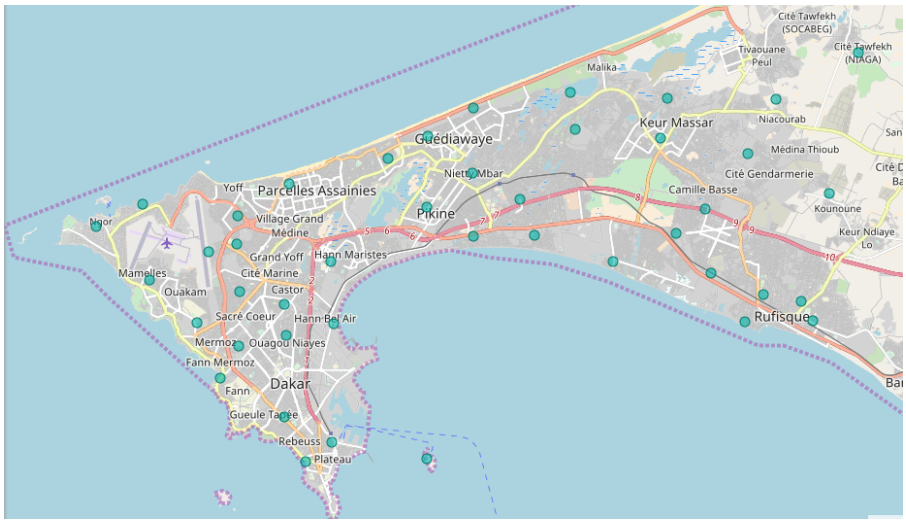

Fig. 5 Considered zones $n = 43$, green points : Average geographical position of subsets of activities

– The case without cost : the location cost is chosen equal to zero for all activities. We notice it data Dkr1_43.
– In the second case, the cost a_{ik} to locate the activities cluster i on the location k is equal to

$$a_{ik} = s_i p_k$$

where s_i is the total surface occupied by i and p_k the land price by (squarre meters) in k. s_i is obtained by adding the surface of each activity in the cluster and p_k from informations given by public organism. This data is noticed Dkr2_43.

The simulation results are given in the table 4 and the time processing times in the table 5.

Data	UB-LS	Greedy(x)	Greedy(y)	GA	GGG
Dkr1_43	281847	284250	286662	282085	281262
Dkr2_43	10986.5	7990.67	7991.65	8015.95	7990.1

Table 4 Values of real life tests : ($\times 10^6$).

The algorithms are particularly good for these real life instances. Our algorithm always finds better solutions than LocalSolver. We show in figure 6 the graphical processing times of all algorithms, and in figure 7 the new locations of the activities cluster found as well as the transportation links computed by our algorithm. With a significantly lower processing times in comparison to

Data	t_{UB-LS}	$t_{Greedy(x)}$	$t_{Greedy(y)}$	t_{GA}	t_{GGG}
Dkr1_43	1752	260	251	294	142
Dkr2_43	1246	289	296	186	183

Table 5 Processing times of tests on real life.

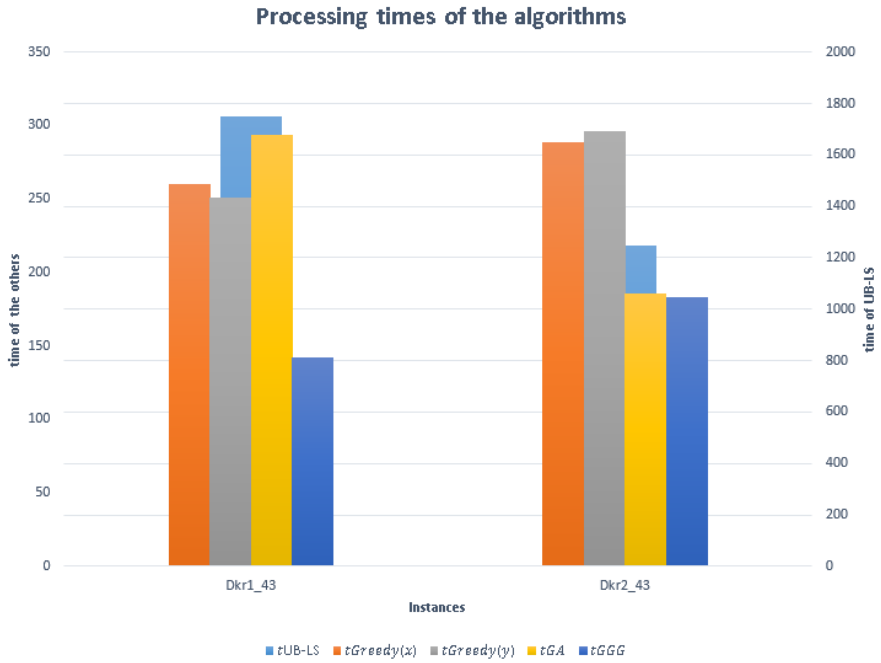

Fig. 6 Processing times of algorithms

LocalSolver, the best algorithm GGG always find better solution (in term of objective function value). The best solution is graphically presented in figure 7. We can observe in this figure that, for instance, the activity (in blue) 29 has been assigned to the location (in black) 11 and the activity 15 to the location 8. But, activities 29 and 15 was initially located in the areas 29 and 15. Actually 29 is a residential area, with few jobs activities, in comparison to the area 15 which is a dynamic activity center in Dakar. It is also known that, in Dakar, the traffic (specially in the work periods) goes from suburbs areas in the east of the picture towards the job areas in the west. Observing the map, we can see that the activities 29 and 15 are initially far away from each other. So, the solution found suggests to put them more closer to each other in the areas 8 and 11 in order to reduce the transportation costs of the individuals traveling from area 29 to 15. Even if in practice such permutation will be difficult, this is precise the kind of results we theoretically expect.

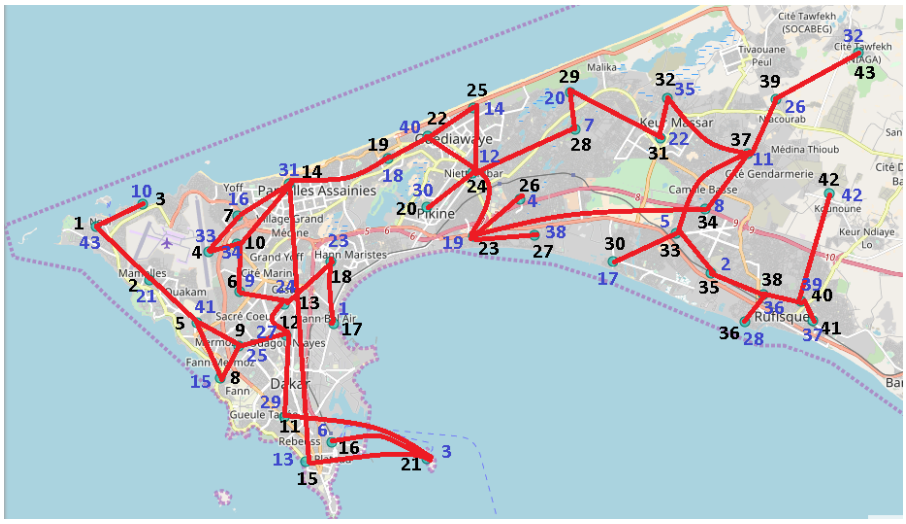

Fig. 7 Result coresponding to the best solution for Dkr1_43 instance : $n = 43$ activities(blue) and zones(black), with road(red) to build in logical link.

7 Conclusion and Perspectives

We propose, in this paper, methodologies to solve the Optimal Network and Quadratic Assignment Problem. We formulate it as a mixed integer programming problem, and propose a greedy algorithm, based on theorem 1, showing how to reach a local optima by solving alternatively a QAP problem and a network design one. To solve the sub-problems (all NP-Hard), standard schemes have been exploited : 2-opt neighbourhood search heuristic for QAP, deletion method for the network design problem. Genetic algorithm is then applied in an initial population composed of the greedy solution and some others randomly perturbed from it. The best solutions found by the genetic algorithm have been used as starting points for the greedy algorithm. We call this algorithm the Greedy-Genetic-Greedy algorithm. Despite the fact that several iterations between the greedy and the genetic method can be done, the algorithm GGG is limited to just 1 iteration for processing time reasons. With just 1 iteration, our method performed better (in terms of solution qualities and processing times) than the best local search software (LocalSolver) when using academic instances.

Hence, our main theoretical contribution resides in an original greedy method, based on theorem 1, to solve an NP-Hard problem combining QAP and a network design problem. This method provides solutions used as initial population in a genetic algorithm in which several implementation choices,, based on the literature, have been done. The best algorithm (GGG), in the light of the numerical results, alternates iteratively between the greedy and the genetic algorithms. The construction of a real-life data with which this

algorithm has been applied is also another important but practical contribution. Indeed, a significant work has been performed to derive from big data files, containing trajectories of millions of individuals in Senegal, suitable informations with which new activity locations and transportation network design have been computed. Notice that processing these big data to extract such relevant informations is not a trivial task. Several steps and choices, explained in the sub-section 5.2, are necessary : clustering of the mobile phone antenna, exploitation of the OSM resources to identify activities in Dakar, assignment of these activities to the closest clusters, processing the files to compute the amount of trips (flows) between clusters, etc. (see section 5.1). In comparison to LocalSolver, for this instance, our genetic algorithm is also able to compute meaningful solutions (as illustrated in the picture) in a reduced amount of times.

However, despite of the good numerical observations, attention must be paid to the results interpretation and the problem formulation. In this paper, two important assumptions have been made. In the first one, the well-known congestion phenomena have been omitted. Indeed, we always assume that the flows are routes on the shortest path according to the route lengths. This is not always true. A more sophisticated model should express the routing time in any arc as a function of the flows. We will fall in this case on more complex equilibrium models. The other hypothesis consists in considering that the activity locations are independent of the O-D flows. Wherever the activities are located, our model based on the Quadratic Assignment Problem assumes that the O-D flows are constant. But in practice, changing the locations of some activities may have a big impact on the flows arriving or leaving them. A model closer to the reality should express the O-D flows as a function of the activity locations leading to much more complex formulations.

Acknowledgements The authors are grateful to John Catherall and James Bleach (the õbex project) for their editorial support. As well as to the two reviewers for their valuable comments that help to improve the paper quality.

References

1. Balàc M., Ciari F., Genre-Grandpierre C., Voituret F., Gueye S., and Michelon P., 2014, Decoupling accessibility and automobile mobility in urban areas. *in Transport Research Arena*, Paris.
2. Bean J.C., 1994, Genetic algorithms and random keys for sequencing and optimization. *ORSA J. Comput.* , Vol. 6, pp. 154-60.
3. Billheimer J.W., 1970, Optimal route configurations with fixed link construction costs, *Stanford Research Institut. SRI-Project* 454531-309.
4. Burkard R. E., Karisch S. E., and Rendl F., 1997, Qaplib a quadratic assignment problem library. *J. Global Opt.*, Vol. 10, pp. 391403.
5. Davis L., 1991, Handbook of Genetic Algorithms, Van Nostrand. New York.
6. Elshafei A. N., 1977, Hospital Layout as a Quadratic Assignment Problem. *Operational Research Quarterly*, Vol. 28, No. 1, Part 2, pp. 167-179.

7. Fleurent C., Ferland J.A., 1994, Genetic hybrids for the quadratic assignment problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 16, Providence. RI: American Mathematical Society, pp. 173-87.
8. Floyd R. W., 1962, Algorithm 97: Shortest Path. Communications of the ACM 5 (6): 345.
9. Gamvros I., Golden B., Raghavan S., and Stanojevi D., Heuristic search for network design. *The Robert H. Smith School of Business*. University of Maryland. College Park, MD 20742-1815.
10. Garey M-R., and Johnson D-S.,1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. New York, NY, USA.
11. Goldberg D.E., Genetic Algorithms in Search, Optimization, and Machine Learning. *Addison-Wesley Publishing Company*, Reading, MA, 1989.
12. Gueye S., Ndiaye B.M., Josselin D., Poss M., Faye R.M., Michelon P., Genre-Grandpierre C., and Ciari F., 2015, Using mobile phone data for Spatial Planning simulation and Optimization Technologies (SPOT), Data for Development Challenge Senegal, *in Book of Abstracts: Scientific Papers. N T15*, p 516-534.
13. Hadley S. W., Rendl F. and Wolkowicz H., 1992, A New Lower Bound via Projection for the Quadratic Assignment Problem. *Mathematics of Operations Research*, Vol. 17, No. 3, pp. 727-739.
14. Holland, J. H., 1975, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI.
15. Krarup J., and Pruzan P. M., 1978, Computer-aided layout design. *Mathematical Programming Study*, Vol. 9, pp. 75-94.
16. Lawler E.L., 1963. The quadratic assignment problem. *Management Science*, Vol. 9, pp. 586-599.
17. Li Y., and Pardalos P.M, 1992, Generating quadratic assignment test problems with known optimal permutations. *Computational Optimization and Applications*, 1:163-184.
18. Los M., 1978, Simultaneous optimization of land use and transportation. A Synthesis of the Quadratic Assignment Problem and the Optimal Network Problem. *Regional Science and Urban Economics*, Vol. 8, pp. 21-42.
19. Los M., 1979, A discrete-convex programming approach to simultaneous optimization of land use and transportation. *Transp. Res-B*, Vol. 13B, pp. 33-48.
20. Xiong Y., and Schneider JB., 1995, Transportation network design using a cumulative genetic algorithm and neural network. *Transportation Research Record*, 1364, pp. 37-44
21. Abdel-Baset M. , Manogaran G. , El-Shahat D., and Mirjalili S., 2018, Integrating the whale algorithm with Tabu search for quadratic assignment problem: A new approach for locating hospital departments *Applied Soft Computing Journal*, 73, 530-546.
22. Abdel-Baset M. , Wu H. , Zhou Y., and Abdel-fatah L., 2017, Elite opposition-flower pollination algorithm for quadratic assignment problem. *Journal of Intelligent & Fuzzy Systems*, 33, 901-911.
23. Lundqvist L., 1973, Integrated location- Transportation Analysis; A decomposition approach. *Regional and Urban Economics*, Vol. 3, N3, pp. 233-262.
24. Lin J-L., and Feng C-M., 2003, A bi-level programming model for the land use- network design problem. *The Annals of Regional Science*, 37, pp. 93-105.
25. Nugent C.E., Vollman T.E., and Ruml, 1968, J. An experimental comparison of techniques for the assignment of facilities to locations. *Oper. Res.*, Vol. 16, pp.150-73.
26. Patriksson M., 1994, The traffic assignment problem : Models and methods. Linkping Institute of Technology, Linkping, Sweden, VSP.
27. Sean L., 2013, Essentials of Metaheuristics, Lulu, Second Edition, available at http://cs.gmu.edu/∼sean/book/metaheuristics/
28. Roucairol C.. Du sequentiel au parallèle: la recherche arborescente et son application à la programmation quadratique en variables 0 et 1, 1987. *Thèse d'Etat, Université Pierre et Marie Curie*, Paris, France.
29. Scott A.J., 1969, The optimal network problem: Some computational procedures. *Transportation Research* , Vol. 3, pp. 201-210.
30. Scriabin M., and Vergin R.C., 1975, Comparison of computer algorithms and visual based methods for plant layout. *Management Science*, 22:172-187.
31. Sheffi Y., 1984, Urban Transportation Networks: Equilibrium Analysis With Mathematical Programming Techniques. Prentice Hall.

32. J. Skorin-kapov. Tabu search applied to the quadratic assingnment problem. *ORSA Journal on Computing*, 2(1):33-45, 1990.
33. E.D. Taillard. Comparison of iterative searches for the quadratic assingnment problem. Location Science,3:87-105, 1995.
34. Tate D.E., and Smith A.E., 1985, A genetic approach to the quadratic assignment problem. *Comput. Oper. Res.*, Vol. 22, pp. 73-83.
35. Thonemann U.W., and Bölte A. An improved simulated annealing algorithm for the quadratic assignment problem. *Working paper, School of Business*, Department of Production and Operations Research, University of Paderborn, Germany, 1994.
36. Wilheilm M.R., and Ward T.L.. Solving quadratic assignment problems by simulated annealing. *IIE Transaction*, 19/1:107-119, 1987.
37. Battiti, R., and Tecchiolli, G., 1994, The reactive tabu search. *ORSA Journal of Computing*, 6(2), 126140.
38. Benlic, U., Hao, and J. K. (2013c). Breakout local search for the quadratic assignment problem. Applied Mathematics and Computation, 219(9), 48004815.
39. Stützle, T., 2006, Iterated local search for the quadratic assignment problem. European Journal of Operational Research, 174(3), 15191539.