Applied Mathematical Sciences, Vol. 1, 2007, no. 46, 2293 - 2312

A Survey of Meta-Heuristic Solution Methods for the Quadratic Assignment Problem

Hassan Mishmast Nehi [1] **and Shahin Gelareh**

Department of Mathematics
University of Sistan and Baluchestan
Zahedan, Iran

Abstract

The quadratic assignment problem (QAP) belongs to the class of NP-Hard problems and also is one of the hardest problems in this class. Today, regarding current hardware, solving the large size instances of this problem, using exact methods, is not possible in reasonable amount of time. In this way many heuristic (Meta-heuristic) and approximation methods and soft-computing approaches have been applied to this problems that we will review some of them in this paper. The aim of this paper is to compare some of efficient heuristic (Meta-heuristic) and soft-computing methods known up to now. Some of them are imitated from the nature's behavior while some other are most analytical. These methods are known as Ant Colony Optimization (ACO), Artificial Neural Networks (NN), Genetic Algorithms (GA), Scatter Search (SS), Simulated Annealing (SA), Tabu Search (TS) and Greedy Randomized Adaptive Search Procedure (GRASP).

Mathematics Subject Classification: 49N10

Keywords: Quadratic Assignment Problem; Meta-Heuristic; Ant Colony Optimization; Genetic Algorithms; Neural Networks; NP-Hard problem

1 Introduction

The Quadratic Assignment Problem (QAP) is one of the classical optimization problems and is widely regarded as one of the most difficult problems in this class. Given a set of $N = \{1, 2, ..., n\}$, and $n \times n$ matrices $F = \{f_{ij}\}$, called flow matrix, $D = \{d_{ij}\}$, as distance matrix, and $C = \{c_{ij}\}$, as setup cost. The QAP is to find a permutation ϕ of the set N which minimizes:

[1]Corresponding author, hmnehi@hamoon.usb.ac.ir (H. M. Nehi), gelareh@mail.usb.ac.ir (S. Gelareh)

$$z = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}d_{\phi(i)\phi(j)} + \sum_{i=1}^{n} c_{i\phi(i)}. \quad (1)$$

In 1957, Koopman and Beckman introduced this problem as a mathematical model for the location of a set of indivisible economical activities.

Consider the problem of allocating a set of facilities to a set of locations with the cost(that corresponds to objective function)being a function of distance and flow between the facilities, plus costs associated with a facility being placed at a certain location. the objective is to assign each facility to a location such that the total cost is minimized.

2 Formulations

Many formulations are proposed for this problem. These includes formulations based on boolean program followed by integer linear programming, mixed integer linear programming, relaxation, permutation based formulations, trace formulation, semidefinite programming and graph based formulations.
Integer linear programming (IP) formulation initially is proposed by Koopman and Beckman in 1957. Working on this formulation is continued until recent works of Yu and Sarker[35] and finally Fedjki and Duffaa [16].
This formulation of a QAP with size n is as follow:

$$\begin{array}{ll} min & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{p=1}^{n} f_{ij}d_{kp}x_{ik}x_{jp} \\ s.t. & \sum_{i=1}^{n} x_{ij} = 1, 1 \le j \le n \\ & \sum_{j=1}^{n} x_{ij} = 1, 1 \le i \le n \\ & x_{ij} \in \{0,1\}, 1 \le i,j \le n \end{array}$$

where $F = [f_{ij}]$, $B = [b_{ik}]$ are the flow and distance matrix, respectively. Considering the initialization of cost of assignment of facility i to location j a matrix B is introduced as the cost of initial assignment and formulation has its following general form:

$$\begin{array}{ll} min & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{p=1}^{n} f_{ij}d_{kp}x_{ik}x_{jp} + \sum_{i=1}^{n}\sum_{k=1}^{n} b_{ik}x_{ik} \\ s.t. & \sum_{i=1}^{n} x_{ij} = 1, \quad 1 \le j \le n \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad 1 \le i \le n \\ & x_{ij} \in \{0,1\}, \quad 1 \le i,j \le n \end{array}$$

The linear summation term of this formulation can easily be solved. So in most papers this term is ignored.

In 1963 Lawler [25] proposed a more general form of QAP as follow:

$$\begin{array}{ll} min & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{p=1}^{n} f_{ij}d_{kp}x_{ik}x_{jp} + \sum_{i=1}^{n}\sum_{k=1}^{n} b_{ik}x_{ik} \\ s.t. & \sum_{i=1}^{n} x_{ij} = 1, \quad 1 \leq j \leq n \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad 1 \leq i \leq n \\ & x_{ij} \in \{0,1\}, \quad 1 \leq i,j \leq n \end{array}$$

where c_{ijkp} do not necessarily corresponds to product of flows by distances. The other formulation is due to the work of Lawler which replaced the quadratic term with a linear term using n^4 variables follow:

$$c_{ijkp} = f_{ij}d_{kp}$$

and

$$y_{ijkp} = x_{ik}x_{jp}$$
$$1 \leq i,j,k,p \leq n$$

Formulation based on relaxation of the original problem is also proposed in work of Love and Wong [26] and many other works was done until the work of Ramakrishnan *et al.* [33].
Between 1976 up to 2004, most of techniques were focused on linearization methods.
Permutation based formulation is based on pair-wise allocation of object cost to adjacent positions. This approach is initially introduced by Hillier and Michaeel [22] and recent works on this formulation is refered to Angel and Zissimopoulos [2, 3, 4], Gutin and Yeo [20], Boaventura-Netto [8]. Formulation is as follow :

$$min_{\pi \in S_n} \sum_{j,j=1}^{n} f_{ij}d_{\pi(i)\pi(j)}$$

where $x_{ij} = \begin{cases} 1 & if\ \pi(i) = j \\ 0 & if\ \pi(i) \neq j \end{cases}$ and S_n is a permutation of the set N.

2.1 Solution methods

2.1.1 Exact algorithms

This algorithms for QAP include approaches based on :
(i) Branch and Bound, with the first works of Gilmore [19] Land in 1963 [24] and Lawler [25]
(ii) Dynamic programming, used for special case of QAPs, with the first work of Christofides [10]
(iii) Cutting plane, with the first work of Bazara and Sherali in 1980 [7].

2.1.2 Heuristic algorithm

Construction methods: Introduced by Gilmore [19] that completes an empty permutation with each iteration of the algorithm. In this method at each iteration of method an object is selected from a list of available object and also a location is selected from the list of available locations and these are removed from their lists and are assigned to each other.

Limited enumeration methods: The convergence to optimality is guaranteed if they can go to end of the enumeration process. However it is possible that a good solution or even an optimal solution is found by the beginning of the process. But it takes a lot of time to converge and may be get stuck in a loop and can not escape. The first works in this area are belong to Burkard and Bonniger [9], Nissen and Paul [32] and West [38].

Local improvement method: These are local search algorithms and include most of heuristics applied to QAP, These methods begins with a feasible solution and tries to improve it by searching neighborhood.
Work in this field is started by the work of Heider [21] and recent works belong to Misevicius in 2000 [30],Mills *et al.* [29].

3 Review of Meta-Heuristic methods

In this section we briefly introduce seven cases of this algorithms as follow:

3.1 *Simulated Annealing*

In 80's, a simulated annealing method proposed for it. Connoly [11], also,proposed an improved scheme for it.
Simulated annealing (SA) originated in statistical mechanics. It is based on a Monte Carlo model that was used by Metropolis *et al.* in 1953 to simulate energy levels in cooling solids. Boltzmann's law was used to determine the probability of accepting a perturbation resulting in a change ΔE in the energy at the current temperature t, i.e.

$$\begin{cases} 1 & \delta E < 0 \\ e^{\frac{-\delta E}{C_B^t}} & \delta E \geq 0. \end{cases}$$

, where C_B is a Boltzmann's constant. Cerny in 1982 and Kirkpatrick *et al.* [**?**] were the first who applied simulated annealing to solve combinatorial optimization problems (COPs). Starting from 1984, several authors applied simulated annealing to the QAP.

3.1.1 Principles

Let S be a set of solutions of combinatorial optimization problems with objective (cost) function $f : S \rightarrow R^1$. Furthermore, let $N : S \rightarrow 2^s$ be a neighborhood function which defines for each $s \in S$ a set $N(s) \subseteq S - a$ a set of neighboring solutions of s. Each solution $s' \in N(s)$ can be reached directly from s by an operation called a *move*. Generally, the move follows objective function evaluation which is called a *trial*.
Table 1 summarized analogy between physical system and optimization problem factors in SA presented:

Table 1: Analogy between physical system and optimization problem

Physical System	Optimization problem
System state	Feasible solution
Energy	Evaluation function
Ground state	Optimal solution
Rapid quenching	Local search
Temperature	Control parameter

3.1.2 Algorithm

Algorithm *SimAn*
1. T← *findStartTemp()*
2. s← *randomStartSolution()*
3. while **not** *frozen()*
4. do while **not** *equilibrium()*
5. do $s_n \leftarrow$ *getNeighbourhoodSolution(s)*
6. δf *eval*(s_n)-*eval*(s)
7. if $\delta f \leq 0$ then $s \leftarrow s_n$
8. else if $random(0,1) \leq \exp(-\delta f/T)$ then $s \leftarrow s_n$
9. $T \leftarrow cool(T)$
10. *report*(s)

Simulated annealing algorithms differ each from other with respect to the following factors: *neighborhood search*, *cooling (annealing) schedule* and *termination criterion*.

Let $S = \{s|s = (s(1), s(2), ..., s(n))\}$, where n is the cardinality of the set. Given a solution s from S, a $k - exchange$ neighbourhood function $k(s)$ is defined as follows:

$$N_k(s) = \{s'|s' \in S, d(s, s') \leq k\}$$

where $d(s, s')$ is the distance between solutions s and s' :

$$d(s, s') = \sum_{i=1}^{n} sgn|s(i) - s'(i)|.$$

If k=2, one obtains 2-exchange neighbourhood function which is widely used in combinatorial problems. In this case, any neighboring solution s' can be reached from the solution s by interchanging (displacement) exactly two elements in s.

3.1.3 Cooling Scheme

The cooling schedule, in turn, is specified by :
- an initial (and final) value of the temperature
- an updating function for changing the temperature
- an equilibrium test.

The behaviour of the simulated annealing algorithm depends on the temperature t. Perhaps the most important thing is how the initial temperature t_0 is determinate.

3.1.4 Termination criterion

In theory the simulated annealing procedure should be continued until the final temperature t_f is zero, but in practice other stopping criteria are applied:
- the value of the objective function has not decreased for a large number of consecutive trials
- the number of accepted moves has become less than a certain small threshold for a large number of consecutive trials
- a fixed a priori number of trials have been executed.

3.2 *Tabu Search*

Skorin-Kapov, in 1990, presented a kind of Tabu search [36]. In 1991, Taillard proposed a robust tabu method [37]. The Reactive tabu method has been proposed in 1994 [6]. Recently, Drezner [14] introduced a new tabu search.
The main ideas of Tabu search (TS) may briefly sketched as follows. The first ingredient that is common to most heuristic and algorithmic procedures is to define a neighborhood or a set of moves that may be applied to a given solution to produce a new one.
Among all the neighboring solutions, TS seeks one with a best heuristic evaluation. In the simplest case such an evaluation dictates the choice of a move that improves most the objective function.If there are no improving moves (indicating a kind of local optimum), TS choose one that least degrades the objective function.

In order to avoid returning to the local optimum just visited, the reverse move must be forbidden. This is done by storing this move (or more precisely a characterization of this move) in a data structure that is called *tabu-list*. This list contains a number s of elements defining forbidden (tabu) moves. The parameter s is called the tabu list size. Choice of this tabu list size is so critical. Many researches are done to determining the optimal or good size of this tabu list and various approaches including the fixed size and dynamic size are proposed by authors. Consequently, an aspiration criterion is introduced to allow tabu moves to be chosen if these are judged to be interesting.

3.2.1 Memory Structure

The memory used in tabu search is both explicit and attributive. Explicit memory records complete solutions, typically consisting of elite solutions visited during the search. An extension of this memory records highly attractive but unexplored neighbors of elite solutions. The memorized elite solutions (or their attractive neighbors) are used to expand the local search.
Alternatively, TS uses attributive memory for guiding purposes. This type of memory records information about solution attributes that change in moving from one solution to another. For example, in a graph or network setting, attributes can consist of nodes or arcs that are added, dropped or repositioned by the moving mechanism. In production scheduling, the index of jobs may be used as attributes to inhibit or encourage the method to follow certain search directions.

3.2.2 Intensification and Diversification

Two highly important components of tabu search are intensification and diversification strategies. Intensification strategies are based on modifying choice rules to encourage move combinations and solution features historically found good. They may also initiate a return to attractive regions to search them more thoroughly. Since elite solutions must be recorded in order to examine their immediate neighborhoods, explicit memory is closely related to the implementation of intensification strategies.

Here the term 'neighbors' has a broader meaning than in the usual context of 'neighborhood search'. That is, in addition to considering solutions that are adjacent or close to elite solutions by means of standard move mechanisms, intensification strategies generate neighbors by either grafting together components of good solution or by using modified evaluation strategies that favor the introduction of such components into a current (evolving) solution. The diversification stage on the other hand encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from those seen before. Again, such an approach can be based

on generating subassemblies of solution components that are producing full solutions, or can rely on modified evaluations as embodied, for example, in the use of penalty/incentive functions. Intensification strategies require a means for identifying a set of elite solutions as basis for incorporating good attributes into newly created solutions. Membership in the elite set is often determined by setting a threshold which is connected to the objective function value of the best solution found during the search. However, considerations of clustering and anti-clustering are also relevant for generating such a set, and more particularly for generating subsets of solutions that may be used for specific phases of intensification and diversification.

3.2.3 Algorithm

Algorithm *TS*
1. *choose the initial solution S randomly, set its tabu list* $T = \phi$.
3. *set BEST_SOLUTION:=S*
4. *set i:=0* {i is the iteration counter}
5. **repeat**
6. *set i:=i+1*
7. *identify S', the best neighbor of* $\mathcal{S}$
8. *set SWAP:= move(S,S')* {*SWAP holds the move transforming S into S'*}
9. *if SWAP* $\notin$ *T*
10. *then update(SWAP,T,LENGTH_TABU_LIST)*
11. *set S:=S'*
12. *if z(S)<z(BEST_SOLUTION)*
13. *then set BEST_SOLUTION:=S*
14. *elseif z(S')<z(BEST_SOLUTION)* {*use of aspiration level*}
15. *then update(SWAP,T,LENGTH_TABU_LIST)*
16. *set BEST_SOLUTION:=S'*
17. *set S:=S'*
18. *if (i mod 2×MAX_TABU_LIST) = 0*
19. *then set LENGTH_TABU_LIST:=random(MIN_TABU_LIST, MAX_TABU_LIST)*
20. **until** {END_TEST}

The algorithms uses the following functions. *update*: inserts SWAP as the first element of T and removes the last element of T if the tabu list was full. *move*: returns the swap that transforms a solution into a second one. The parameters of the algorithm are *MIN_TABU_LIST*, *MAX_TABU_LIST*: specify the minimal and maximal allowed length of the tabu list, respectively.

3.3 *Ant Colony Optimization*

Gambardella *et al.*, 1999 [18] and Hoos *et al.* in 1999 proposed Ant algorithm approaches. Recent work in this are due to the works of Middendorf *et al.* [28] and Ying and Liao in 2004 [40].
Ant Colony Optimization (ACO) is a population-based approach which has been successfully applied to several NP-hard COPs. As the name suggests, ACO has been inspired by the behavior of real ant colonies, in particular, by their foraging behavior. One of its main ideas is the indirect communication among the individuals of a colony of agents, called (artificial) ants, based on an analogy with trails of a chemical substance, called *pheromone*, which real ants use for communication. The (artificial) pheromone trails are a kind of distributed numeric information (called stigmergic information in some papers) which is modified by the ants to reflect their experience accumulated while solving a particular problem.
The ants construct iteratively their candidate solution to a COP.The ants' solution construction is guided by (artificial) pheromone trails and problem-dependent heuristic information. By defining solution components ,ACO can be applied to any binary COP. An individual ant constructs candidate solutions by starting with an empty solution and then iteratively adding solution components until a complete candidate solution is generated.Each point at which an ant has to decide which solution component to add to its current partial solution a *choice point*.After the solution construction is completed, the ants give feedback on the solutions they have constructed by depositing pheromone on solution components which they have used in their solution. Typically, solution components which are part of better solutions or are used by many ants will receive a higher amount of pheromone, and hence, will more likely be used by the ants in future iterations of the algorithm. To avoid getting stuck, in an local optima before the pheromone trails get reinforced, all pheromone trails are decreased by a factor ρ which is called evaporation coefficient .

3.3.1 Available ACO algorithms for the QAP

Ant System (AS) is the first among ACO algorithms which has been applied to the QAP. After it, several improved ACO applied to this problem have been proposed by various authors. Despite the differences among these algorithms, they share at least two important common aspects. **solution construction**: All the proposed ant algorithms for the QAP associate pheromone trails τ_{ij} only to couplings of the form $\psi_i = j$, hence, τ_{ij} can be interpreted as the desirability of assigning facility i to location j. **Local Search Mechanism:** all the proposed algorithms improve the ants' solutions using a local search algorithm.
A general Scheme for ACO algorithm can be sketched as follow:

procedure *ACO algorithm for static combinatorial problems*
1. Set parameters, initialize pheromone trails
2. **while** (termination criterion not met) **do**
3. ConstructSolutions
4. ApplyLocalSearch %optimal
5. UpdateTrails
6. **end**
7.**end**

Here we present a brief description of the first algorithm which named Ant System and we refer it AS-QAP. In AS-QAP, the assignment order is determined by a pre-ordering of the facilities, as explained below. Then, at each step a facility i is assigned probabilistically to some location j preferring locations with a high pheromone trail τ_{ij} and promising heuristic information η_{ij}.

Heuristic information: The heuristic information on the potential goodness of an assignment is determined in AS-QAP as follows. Two vectors d and f are calculated in which the ith component represents respectively the sum of the distances from location i to all other locations and the sum of the flows from facility i to all other facilities. The lower d_i, the distance potential of location i, the more central is considered the location, the higher f_i, the flow potential of facility i, the more important is the facility. Next a coupling matrix $E = f.d^T$ is calculated, where $e_ij = f_i.d_j$. Then, the heuristic desirability of assigning facility i to location j is given by $\eta_{ij} = 1/eij$. The motivation for using this type of heuristic information is that, intuitively, good solutions will place facilities with high flow potential on locations with low distance potential.
Solution construction A solution is constructed as follows. In AS-QAP facilities are sorted in non increasing order of the flow potentials. At each construction step an ant k assigns the next still unassigned facility i to a free location j with a probability given by:

$$p_{ij}^k = \frac{[\tau_{ij}(t)]^\alpha.[\eta_{ij}]^\beta}{\sum_{l\in N_i^k}[\tau_{il}(t)]^\alpha.[\eta_{ij}]^\beta}$$

where $\tau_{ij}(t)$ is the pheromone trail at iteration t, α and β are parameters which determine the relative influence of the pheromone strength and the heuristic information, and N_i^k is the feasible neighborhood of node i, that is, only those locations that are still free (note that $k\ \sum_{j\in N_i^k} p_{ij}(t) = 1$). The single steps are repeated until a complete assignment is constructed.

Pheromone update: The pheromone trail update applied to all couplings is done according to the following equation:

$$\tau_{ij}(t+1) = \rho.\tau_{ij}(t) + \sum_{k=1}^{m} \triangle\tau_{ij}^{k}$$

where ρ, with $0 < \rho < 1$, is the persistence of the pheromone trail, so that $(1-\rho)$ represents the evaporation. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and allows the algorithm to forget previously done bad choices. $\triangle\tau_{ij}^{k}$ is the amount of pheromone ant k puts on the coupling (i,j); it is given by

$$\begin{cases} Q/J_{\psi}^{k} & if\ facility\ i\ is\ assignemd\ to\ location\ j\ in\ the\ solution\ of\ ant\ k \\ 0 & otherwise \end{cases}$$

where ψ^k is the kth ant solution, J_{ψ}^{k} its objective function value, and Q is the amount of pheromone deposited by an ant.
A first improvement of AS-QAP is presented in [27] and we refer to it as AS2-QAP. AS2-QAP differs mainly in the way the heuristic information is calculated and in a different way of calculating the probability of assigning facilities to locations. A further improvement over AS2-QAP, called ANTS-QAP and after it many other improvements are done to ant algorithms.

3.4 *Genetic Algorithms*

Genetic algorithms approach also introduced by Tate *et al.* in 1995 [39]. In 1994, Fleurent *et al.* [17], Ahujain in 2000 [1]. Drezner [13] proposed a more hybrid efficient versions of it. Recently Misevicius [31] introduced a hybrid genetic algorithm, Balakrishnan *et al.* [5], Rodriguez *et al.* in 2004 [34] used a combined GA-TS .
Genetic algorithms(GA) represent a powerful and robust approach for developing heuristics for large-scale combinatorial optimization problems. The motivation underlying genetic algorithms can be expressed as follows: Evolution has been remarkably successful in developing complex and well-adapted species through relatively simple evolutionary mechanisms.
GAs imitate the process of evolution on an optimization problem. Each feasible solution of a problem is treated as an individual whose fitness is governed by the corresponding objective function value. A GA maintains a population of feasible solutions (also known as *chromosomes*) on which the concept of the survival of the fittest (among string structures) is applied. There is a structured yet randomized information exchange between two individuals (crossover

operator) to give rise to better individuals. Diversity is added to the population by randomly changing some genes (mutation operator) or bringing in new individuals (immigration operator). A GA repeatedly applies these processes until the population converges.
An implementation of a genetic algorithm begins with a population of (typically random) chromosomes. One then evaluates these structures and allocates reproductive opportunities in such a way that those chromosomes which represent a better solution to the target problem are given more chances to 'reproduce' than those chromosomes which are poorer solutions. The 'goodness' of a solution is typically defined with respect to the current population.
This particular description of a genetic algorithm is intentionally abstract because in some sense, the term genetic algorithm has two meanings. In a strict interpretation the genetic algorithm refers to a model introduced and investigated by John Holland and by students of Holland.
In a broader usage of the term, a genetic algorithm is any population based model that uses selection and recombination operators to generate new sample points in a search space. Many genetic algorithm models have been introduced by researchers largely working from an experimental perspective. Many of these researchers are application oriented and are typically interested in genetic algorithms as optimization tools.

3.4.1 Algorithm

```
   algorithm genetic;
1  begin
2    obtain initial population;
3      repeat select
4        TwoIndividualsI1AndI2InThePopulation;
5        ApplyTheCrossoverOperatorOnI1AndI2ToProduceAChildI3;
6        ReplaceOneOfTheTwoIndividualsI1OrI2ByI3;
7        OccasionallyPerformImmigration;
8       until ThePopulationConverges;
9  end;
```

This is a simple high-level description of our GA. Each execution of the repeat loop is called a *trial*.

3.5 *Neural Networks*

In conventional neural approaches, however an Ising (Binary) Spin system introduced by Hopfield and Tank[23], Bilbro, Man, Miller, Snyder, Van Den Bout and White in 1989, Ishii and Sato in 1997, has been used to present a solution. The constraint for the permutation are then implemented as 'soft'

constrains. This implementation often produce infeasible solutions. This is one of the reason why these approaches are not very good when the problem size is expanded.
Artificial neural networks (ANN) methods and particularly, the recurrent neural networks method based on deterministic annealing are most interesting to COPs. In contrast to most other methods, these are not based on a direct exploration of the given discrete state-space. Instead, they utilize an interpolating continuous (analog) space, allowing for shortcuts to good solutions. Key concepts here are the mean-field (MF) approximation (Hopfield and Tank, 1985 [23]; Peterson and Söderberg, 1989) and annealing.

3.5.1 *Recurrent neural networks*

They appear in the context of associative memories (Hopfield, 1982) as well as difficult optimization problems (Hopfield and Tank, 1985; Peterson and Söderberg, 1989). Such networks resemble statistical models of magnetic systems (*spin glasses*), with an atomic spin state (up or down) seen as analogous to the *firing* state of a neuron (on or off).

The archetype of a recurrent network is the Hopfield model (Hopfield, 1982), based on an energy function of the form:

$$E(s) = -\frac{1}{2}\sum_{ij} w_{ij} s_i s_j$$

in terms of binary variables (or Ising spins, as used in magnetic models), $si = 0, 1$ with symmetric weights w_{ij} . With an appropriate choice of weights determined by a set of stored patterns, the latter appear as local minima, satisfying

$$s_i = sgn(\sum_j w_{ij} s_j)$$

With a simple *asynchronous* dynamics based on iterating above equations, this system turns into a recurrent ANN, having the local minima as stationary points.

3.5.2 *Updating methods*

Two kind of updating is used in this context: *asynchronous* and *synchronous* .

3.5.3 *Mean Field Equations*

An alternative is given by MF annealing, where the stochastic SA method is approximated by a deterministic dynamics based on the MF approximation, defined as follows for a system of Ising spins. By defining $v_i \equiv< s_i >= p_i(1) - p_i(0) \in [0, 1]$, the free energy function that should be minimized is:

$$F(v) = E(v) - TS(v)$$

,where T is the temperature ,$S(v)$ is defined as $S(v) \equiv -\frac{1}{2}\sum_i[(1+v_i)log(1+v_i)+(-v_i)log(-v_i)$ is the entropy associated with the approximating distribution and $E(v) \equiv< E(s) >\equiv -\frac{1}{2}\sum_{i\neq j} w_{ij}v_iv_j$

3.6 *Scatter Search*

Scatter search was introduced by Glover (1977) and for the first time applied to this problem by Cung et al in 1997 [12] .Scatter search is an evolutionary heuristic, proposed two decades ago, that uses linear combinations of a population subset to create new solutions. As comes with its name ,it tries to keep the points as scatter as possible. A special operator is used to ensure their feasibility and to improve their quality. Basically the Scatter Search method starts with a collection of feasible solutions, which is called *reference set*. At each step some of the best solutions are extracted from the collection to be combined. A trial *point* is then created as a linear combination of the extracted points and an operator is applied to the trial point This operator has two purposes. In many cases, a linear combination of integer points will not result in an integer point. The first purpose of the operator is thus to produce an integer(feasible) solution from the trial point.

The second purpose is to improve the quality of the created solution. As a result of the operator, a new feasible solution is obtained which might be included or not, according to some criteria, in the collection.
There are some great differences between Scatter Search and Genetic Algorithms although, in both of them, a set of feasible solutions evolves. First of all, there is no metaphor with nature's behavior in Scatter Search. Its rationale is rather of a geometric or analytic type by taking a linear combination of good solutions, one might expect to obtain a new good solution. Second, this is the very first method which allows combining more than two solutions.

3.6.1 Algorithm

A general template for a Scatter search can be sketched as follow:

Procedure *Scatter Search*
1. *Generate an initial set of solutions P by using a Diversification Generation Method*
2. *Improve these solutions by an improvement method*
3. *With these solutions build an initial RefSet*
4. **repeat**
5. *Obtain all subsets of pairs from RefSet*
6. *Combine these subsets and obtain new solutions*
7. *Improve these solutions by the Improvement Methods*
8. *Update RefSet with these news solutions*
9. **until** *RefSet is stable (i.e. no new solution have been* included
10. *If max_ite iteration (steps 1-4) elapse without improvement stop else return Step1*

Scatter search algorithm can be organized in two phases outlined as follows:

Initial Phase
1.Diversification Generation Method
2.Improvement Method
3.Reference Set Update Method
4. Repeat this initial phase until producing a desirable level of high-quality and diverse solutions.

Scatter Search Phase
5. Subset Generation Method
6. Solution Combination Method
7. Improvement Method
8. Reference Set Update Method.

An application to the QAP can be found in Cung *et al.* (1997) [12].

3.7 *Greedy Random Adaptive Search Procedure*

The Greedy Randomized Adaptive Search Procedure (GRASP) meta-heuristic is a multi-start or iterative process, in which each iteration consists of two phases:*construction* and *local search*.
The construction phase builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result.
The pseudo-code in following Pseudo-code illustrates the main blocks of a GRASP procedure for minimization, in which *Max_Iterations* iterations are performed and *Seed* is used as the initial seed for the pseudorandom number

generator
.

Procedure GRASP(Max Iterations,Seed)
1 *Read Input();*
2 *for k = 1... Max Iterations do*
3 *Solution*← *Greedy_Randomized_Construction(Seed);*
4 *Solution* ← *Local_Search(Solution);*
5 *Update_Solution(Solution,Best Solution);*
6 *end;*
7 **return** *Best Solution;*
end **GRASP**.

Next Pseudo-code illustrates the construction phase with its pseudo-code. At each iteration of this phase, let the set of candidate elements be formed by all elements that can be incorporated to the partial solution under construction without destroying feasibility. The selection of the next element for incorporation is determined by the evaluation of all candidate elements according to a greedy evaluation function.
This greedy function usually represents the incremental increase in the cost function due to the incorporation of this element into the solution under construction. The evaluation of the elements by this function leads to the creation of a restricted candidate list (RCL) formed by the best elements, i.e. those whose incorporation to the current partial solution results in the smallest incremental costs (this is the greedy aspect of the algorithm). The element to be incorporated into the partial solution is randomly selected from those in the RCL (this is the probabilistic aspect of the heuristic). Once the selected element is incorporated to the partial solution, the candidate list is updated and the incremental costs are reevaluated (this is the adaptive aspect of the heuristic). This strategy is similar to the semi-greedy heuristic proposed by Hart and Shogan, which is also a multi-start approach based on greedy randomized constructions, but without local search.

Procedure*Greedy_Randomized_Construction(Seed)*
1 *Solution*← $\emptyset$*;*
2 *Evaluate the incremental costs of the candidate elements;*
3 **while** *Solution is not a complete solution* **do**
4 *BuildTheRestrictedCandidateList(RCL);*
5 *SelectAnElement_s_FromTheRCLAtRandom;*
6 *Solution*←*Solution*$\cup\{s\}$*;*
7 *ReevaluateTheIncrementalCosts;*
8 **end;**
9 *return Solution;*

end *Greedy_Randomized_Construction.*

The solutions generated by a greedy randomized construction are not necessarily optimal, even with respect to simple neighborhoods. The local search phase usually improves the constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. The pseudo-code of a basic local search algorithm starting from the solution Solution constructed in the first phase and using a neighborhood N is given in previous schema.

procedure Local Search(Solution)
1 while Solution is not locally optimal do
2 Find $s' \in N$(Solution) with $f(s') < f$(Solution);
3 Solution s0;
4 end;
5 return Solution;
end Local Search.

The effectiveness of a local search procedure depends on several aspects, such as the neighborhood structure, the neighborhood search technique, the fast evaluation of the cost function of the neighbors, and the starting solution itself. The construction phase plays a very important role with respect to this last aspect, building high-quality starting solutions for the local search. Simple neighborhoods are usually used. The neighborhood search may be implemented using either a *best_improving* or a *firstimproving* strategy .

4 Acknoledgment

The authors should thank of Professor A.Misevicius and Professor P.M Hahn for their kind and useful helps.

References

[1] Ahuja, R., Orlin, J.B. and Tiwari, A. A greedy genetic algorithm for the quadratic assignment problem. Computers and Operations Research, 27(10), 917-934. (2000).

[2] Angel, E. and Zissimopoulos, V. On the quality of local search for the quadratic assignment problem. Discrete Applied Mathematics, 82(1-3), 15-25.(1998).

[3] Angel, E. and Zissimopoulos, V. On the classification of NP-complete problems in terms of their correlation coefficient. DAMATH: Discrete Applied

Mathematics and Combinatorial Operations Research and Computer Science, 99(1-3), 261-277. (2000).

[4] Angel, E. and Zissimopoulos, V. On the landscape ruggedness of the quadratic assignment problem. Theoretical Computer Science, 263(1-2), 159-172.(2001).

[5] Balakrishnan, J., Cheng, C.H., Conway, D.G. and Lau, C.M. A hybrid genetic algorithm for the dynamic plant layout problem. International Journal of Production Economics, 86(2), 107-120. (2003).

[6] Battiti, R. and Tecchiolli, G. The reactive tabu search. ORSA Journal on Computing, 6(2), 126-140.(1994a).

[7] Bazaraa, M.S. and Sherali, H.D. Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem. Naval Research Logistics Quarterly, 27, 29-41. (1980).

[8] Boaventura-Netto, P.O. Combinatorial instruments in the design of a heuristic for the quadratic assignment problems. Pesquisa Operacional, 23(03), 383-402. (2003).

[9] Burkard, R.E and Bonniger, T. A heuristic for quadratic Boolean programs with applications to quadratic assignment problems. European Journal of Operation Research, 13, 374-386. (1983).

[10] Christofides, N. and Benavent, E. An exact algorithm for the quadratic assignment problem. Operation Research, 37(5), 760-768. (1989).

[11] Connolly, D. T. An improved annealing scheme for the QAP. European Journal of Operational Research, 46, 93-100. (1990).

[12] Cung, V-D., Mautor, T., Michelon, P. and Tavares, A. A scatter search based approach for the quadratic assignment problem. In: Proceedings of IEEE International Conference on Evolutionary Computation, 165-169. (1997).

[13] Drezner, Z. A new genetic algorithm for the quadratic assignment problem. Informs Journal on Computing, 15(3), 320-330. (2003).

[14] Drezner, Z. The extended concentric tabu for the quadratic assignment problem. To appear in European Journal of Operational Research, 160, 416-422.(2005).

[15] Edwards, C.S. A branch and bound algorithm for the Koopmans-Beckmann quadratic assignment problem. Mathematical Programming Study, 13, 35-52. (1980).

[16] Fedjki, C.A. and Duffuaa, S.O. An extreme point algorithm for a local minimum solution to the quadratic assignment problem. European Journal of Operational Research, 156(3), 566-578. (2004).

[17] Fleurent, C. and Ferland, J.A. Genetic hybrids for the quadratic assignment problem. In: Quadratic Assignment and Related Problems [edited by P.M. Pardalos and H. Wolkowicz], DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 16, 173-187, AMS, Rhode Island.(1994).

[18] Gambardella, L.M., Taillard, D. and Dorigo, M. Ant colonies for the QAP. Journal of Operational Research. Society, 50, 167-176. (1999).

[19] Gilmore, P.C. Optimal and suboptimal algorithms for the quadratic assignment problem. SIAM Journal on Applied Mathematics, 10, 305-313. (1962).

[20] Gutin, G. and Yeo, A. Polynomial approximation algorithms for TSP and QAP with a factorial domination number. Discrete Applied Mathematics, 119(1-2), 107-116. (2002).

[21] Heider, C.H. An N-step, 2-variable search algorithm for the component placement problem, Naval Research Logistics Quarterly 20, 699-724.(1973).

[22] Hillier, F.S. and Michael, M.C. Quadratic assignment problem algorithms and the location of indivisible facilities. Management Science, 13, 44-57.(1966).

[23] Hopfield ,J.J , Tank D.W, . Neural computation of decisions in optimization problems,Biolog.Cybr,52,141-152 ,(1985)

[24] Land, A.M. A problem of assignment with interrelated costs. Operational Research Quarterly, 14, 185-198. (1963).

[25] Lawler, E.L. The quadratic assignment problem. Management Science, 9, 586-599. (1963).

[26] Love, R.F. and Wong, J.Y. Solving quadratic assignment problems with rectangular distances and integer programming. Naval Research Logistics Quarterly, 23, 623-627. (1976a).

[27] Maniezzo, V. and Colorni, A. The ant system applied to the quadratic assignment problem. Knowledge and Data Engineering, 11(5), 769-778. (1999).

[28] Middendorf, M., Reischle, F. and Schmeck, H. Multi colony ant algorithms. Journal of Heuristics, 8(3), 305-320. (2002).

[29] Mills, P., Tsang, E. and Ford, J. Applying an extended guided local search to the quadratic assignment problem. Annals of Operations Research, 118(1-4), 121-135. (2003).

[30] Misevicius, A. An intensive search algorithm for the quadratic assignment problem. Informatica, 11, 145-162. (2000a).

[31] Misevicius, A. Genetic algorithm hybridized with ruin and recreate procedure: application to the quadratic assignment problem. Knowledge-Based Systems, 16(5-6), 261-268.(2003b).

[32] Nissen, V. and Paul, H. A modification of threshold accepting and its application to the quadratic assignment problem. OR Spektrum, 17(2-3), 205-210. (1995).

[33] Ramakrishnan, K.G., Resende, M.G.C., Ramachandran, B. and Pekny, J.F. Tight QAP bounds via linear programming. In: Combinatorial and Global Optimization [edited by P.M. Pardalos, A. Migdalas, and R.E. Burkard], 297-303, World Scientific Publishing, Singapore. (2002).

[34] Rodriguez, J.M., Macphee, F.C., Bonham, D.J. and Bhavsar, V.C. Solving the quadratic assignment and dynamic plant layout problems using a new hybrid meta-heuristic approach, In: Proceedings of the 18th Annual International Symposium on High Performance Computing Systems and Applications (HPCS), [edited by M.R. Eskicioglu], 9-16 (GATS homepage: http://acrl.cs.unb.ca/research/gats/) (2004).

[35] Sarker, B.R., Wilhelm, W.E. and Hogg, G.L. One-dimensional machine location problems in a multi-product flowline with equidistant locations. European Journal of Operational Research, 105(3), 401-426.(1998).

[36] Skorin-Kapov, J. Tabu search applied to the quadratic assignment problem. ORSA Journal on Computing, 2(1), 33-45.(1990).

[37] Taillard, E. Robust taboo search for the quadratic assignment problem. Parallel Computing, 17, 443-455. (1991).

[38] West, D.H. Algorithm 608: Approximate solution of the quadratic assignment problem. ACM Transactions on Mathematical Software, 9, 461-466. (1983).

[39] Tate, D.E. and Smith, A.E. A genetic approach to the quadratic assignment problem. Computers and Operations Research, 22, 73-83. (1995).

[40] Ying, K.C. and Liao, C.J. An ant colony system for permutation flow-shop sequencing. Computers and Operations Research, 31(5), 791-801. (2004).