ELSEVIER

Theoretical Computer Science 263 (2001) 159–172

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the landscape ruggedness of the quadratic assignment problem

Eric Angel[a,*], Vassilis Zissimopoulos[b]

[a] *Université de Paris Sud, L.R.I, CNRS-URA 410, Centre d'Orsay, 91405 Orsay, France*
[b] *Université Paris Nord, LIPN, CNRS-URA 1507, 93430 Villetaneuse, France*

Accepted April 2000

Abstract

Local-search-based heuristics have been demonstrated to give very good results to approximately solve the quadratic assignment problem (QAP). In this paper, following the works of Weinberger and Stadler, we introduce a parameter, called the ruggedness coefficient, which measures the ruggedness of the QAP landscape which is the union of a cost function and a neighborhood.

We give an exact expression, and a sharp lower bound for this parameter. We are able to derive from it that the landscape of the QAP is rather flat, and so it gives a theoretical justification of the effectiveness of local-search-based heuristics for this problem. Experimental results with simulated annealing are presented which confirm this conclusion and also the influence of the ruggedness coefficient on the quality of results obtained. © 2001 Published by Elsevier Science B.V.

Keywords: Quadratic assignment problem; Local search

1. Introduction

Given two $n \times n$ matrices $F = (f_{ij})$ and $D = (d_{ij})$, the quadratic assignment problem (QAP) asks to find a permutation π which minimizes the sum $\sum_{ij} f_{ij} d_{\pi(i)\pi(j)}$.

This problem is NP-hard [6], and nonapproximable [11]. One of the major applications of the QAP is in location theory where f_{ij} is the flow of materials from facility i to facility j, and d_{ij} represents the distance from location i to location j. The objective is to find an assignment of all facilities to locations which minimizes the total cost. We shall assume, as it is usually the case, that the matrices F and D

* Corresponding author.
E-mail addresses: angel@lri.fr (E. Angel), vz@ura1507.univ-paris13.fr (V. Zissimopoulos).

0304-3975/01/$ - see front matter © 2001 Published by Elsevier Science B.V.
PII: S0304-3975(00)00239-5

are symmetric with a null diagonal, and the cost function to be minimized is written $C(\pi)=\frac{1}{2}\sum_{i,j=1}^{n} f_{ij}d_{\pi(i)\pi(j)}$.

Despite it has been extensively studied since its first formulation in 1957 [9], it remains very hard to solve exactly, and problems with sizes greater than 20 are generally intractable. But, it has been noticed that heuristics give remarkably good results, and specially those based on local search i.e. simulated annealing and tabu search. In [2], the authors have presented a theoretical performance guarantee result for this problem.

Local search algorithms work in an iterative fashion. First, a *neighborhood structure* $\mathcal{N}$ has to be defined. It associates to each solution $s\in\mathcal{S}$, a subset of solutions $\mathcal{N}(s)\subset\mathcal{S}$ called its neighbors. Then, the following instruction is executed: substitute the current solution by a better one in its neighborhood, until a *locally optimum* solution is attained, that is a solution it is no more possible to improve. We consider the 2-exchange neighborhood, which is the standard one for the QAP. Given a permutation $\pi=(\pi(1),\ldots,\pi(i),\ldots,\pi(j),\ldots,\pi(n))$, its neighbors are the $n(n-1)/2$ permutations of the form $(\pi(1),\ldots,\pi(j),\ldots,\pi(i),\ldots,\pi(n))$ for $1\leqslant i<j\leqslant n$, obtained from π by a swap. It means that at each step we exchange the locations of two facilities.

Several experimental studies have reported, that local-search-based heuristics gave very good results when applied to the QAP. In [5] one can read "[...] reflects a QAP property which was not yet put into evidence in the literature: any approach based on local search is bound to be very effective as a heuristic for QAP", and in [4] "Simulated annealing is an extremely efficient heuristic for the QAP".

The next of this paper is organized as follows: In Section 2, after having explained what is meant by the ruggedness of a landscape, we show the importance of this concept for the analysis of the behavior of local search algorithms, and introduce the autocorrelation coefficient and the derived ruggedness coefficient. In Section 3 we give an exact expression, and a sharp lower bound for the autocorrelation coefficient for the QAP. In Section 4, we present a theoretical justification of the effectiveness of local-search-based heuristics for the QAP based on the ruggedness of its landscape. Section 5 is devoted to experimental evaluations.

2. The ruggedness of a landscape

The union of the cost function to be optimized, and the neighborhood forms what is called a *landscape*. It is the presence of numerous local minima in the landscape, which represents the main obstacle for local search algorithms. Various heuristics have been developed to overcome this problem, and simulated annealing and tabu search are among the most popular, yet the number of local minima still remains the main difficulty they are faced to, and even if they are no more trapped in, they slow down the search.

Strongly related to this problem, is the ruggedness of a landscape. Intuitively, it is clear that the number of local minima depends on the link between the cost of a solution and the cost of its neighbors. If the cost difference between any two neighbor-

ing solutions is on average small (respectively important), then the landscape will be flat (respectively very steep), and therefore well (respectively bad) suited for a local search algorithm. The autocorrelation functions, introduced by Weinberger [14], measure the ruggedness of a landscape. In a related study concerning the graph bipartitioning problem [1] we had previously proposed a derived parameter, called the *autocorrelation coefficient*. In the next section we study it for the QAP, and in order to obtain easy interpretable values we will perform a change of scale on this coefficient specially designed for this problem, obtaining in this way a new coefficient which we call the ruggedness coefficient.

In the following we always have symmetric neighborhood, i.e. $s \in \mathcal{N}(t) \iff t \in \mathcal{N}(s)$, for any two solutions $s, t \in \mathcal{S}$. Let the *distance* between any two distinct solutions s and t, noted $d(s,t)$, be the smallest integer $k \geqslant 1$ such that there exists a sequence of solutions $s_0, \ldots, s_k$ with $s_0 = s$, $\forall i \in \{0, \ldots, k-1\}$, $s_{i+1} \in \mathcal{N}(s_i)$ and $s_k = t$. In the sequel, we always have $d(s,t) = d(t,s)$. By definition, the landscape *autocorrelation function* [14] is

$$\rho(d) = 1 - \frac{\langle (C(s) - C(t))^2 \rangle_{d(s,t)=d}}{\langle (C(s) - C(t))^2 \rangle},$$

with $\langle (C(s) - C(t))^2 \rangle$ the average value of $(C(s) - C(t))^2$ over all solutions pairs $\{s,t\}$, and $\langle (C(s) - C(t))^2 \rangle_{d(s,t)=d}$ the average value of $(C(s) - C(t))^2$ over all solutions pairs $\{s,t\}$ which are at distance d. It is not difficult to see that

$$\rho(d) = 1 - \frac{\langle (C(s) - C(t))^2 \rangle_{d(s,t)=d}}{2(\langle C^2 \rangle - \langle C \rangle^2)},$$

with $\langle C \rangle$ (respectively $\langle C^2 \rangle$) the average value of $C(s)$ (respectively $C^2(s)$) over $\mathcal{S}$.

Function $\rho(d)$ shows the level of correlation between any two solutions which are at distance d from each other. The most important value to know is $\rho(1)$, because the link between two adjacent solutions is of first importance for any local-search-based heuristic. A value close to 1 indicates that costs of any two neighboring solutions are in average very close. In contrary, a value close to 0 indicates that the cost of any two neighboring solutions are almost independent.

We define the autocorrelation coefficient ξ by $\xi = 1/(1 - \rho(1))$. The larger ξ is, the more flat is the landscape, and so the more suited is the landscape for any local-search-based heuristic.

3. The autocorrelation coefficient for the QAP

We shall have to calculate various sums over four indices i, j, k, l. It will be convenient to decompose a sum, in several subsums according the number of identical values among the indices i, j, k and l.

The notation $i \neq j \neq k \neq l$ means that the indices i, j, k and l have distincts values, i.e. $(i \neq j) \wedge (i \neq k) \wedge (i \neq l) \wedge (j \neq k) \wedge (j \neq l) \wedge (k \neq l)$. The notation

$i=j,k,l$ means $(i=j)\wedge(k\neq l)\wedge(k\neq i)\wedge(l\neq i)$. The notation $i=j=k,l$ means $(i=j=k)\wedge(l\neq i)$, and $i=j,k=l$ means $(i=j)\wedge(k=l)\wedge(i\neq k)$.

The decomposition is represented as

$$\sum_{i,j,k,l}=\sum_{i\neq j\neq k\neq l}+\sum_{i=j,k,l}+\sum_{i=k,j,l}+\sum_{i=l,j,k}+\sum_{j=k,i,l}+\sum_{j=l,i,k}+\sum_{k=l,i,j}$$

$$+\sum_{i=j,k=l}+\sum_{i=k,j=l}+\sum_{i=l,j=k}+\sum_{i=j=k,l}+\sum_{i=j=l,k}+\sum_{i=k=l,j}+\sum_{j=k=l,i}+\sum_{i=j=k=l}.$$

We will sometimes write $\sum_{i,j,k=l}$ instead of $\sum_{k=l,i,j}$.

We note F_k for the sum $\sum_{i,j} f_{ij}^k$, f_i for the sum $\sum_j f_{ij}$, D_k for the sum $\sum_{i,j} d_{ij}^k$, and d_i for the sum $\sum_j d_{ij}$. The cost function is $C(\pi)=\frac{1}{2}\sum_{i,j=1}^{n} f_{ij}d_{\pi(i)\pi(j)}$.

Lemma 1. *We have the following combinatorial identities*:

$$\sum_{i\neq j\neq k} f_{ij}f_{ik}=\sum_i f_i^2-F_2,$$

$$\sum_{i\neq j\neq k\neq l} f_{ij}f_{kl}=F_1^2+2F_2-4\sum_i f_i^2.$$

Proof. For the first equality, we have

$$\sum_{i\neq j\neq k} f_{ij}f_{ik}=\sum_{i,j} f_{ij}(f_i-f_{ij})$$

$$=\sum_{i,j} f_i f_{ij}-\sum_{i,j} f_{ij}^2$$

$$=\sum_i f_i^2-F_2.$$

For the second equality, we have

$$\left(\sum_{i,j} f_{ij}\right)^2=\sum_{i\neq j} f_{ij}^2+\sum_{(i,j)\neq(k,l)} f_{ij}f_{kl}$$

$$=\sum_{i\neq j} f_{ij}^2+\sum_{i\neq j\neq k\neq l} f_{ij}f_{kl}+4\sum_{i\neq j\neq k} f_{ij}f_{ik}+\sum_{i\neq j} f_{ij}^2.$$

Therefore,

$$\sum_{i\neq j\neq k\neq l} f_{ij}f_{kl}=\left(\sum_{i,j} f_{ij}\right)^2-2\sum_{i\neq j} f_{ij}^2-4\left(\sum_i f_i^2-\sum_{i,j} f_{ij}^2\right)$$

$$=F_1^2+2F_2-4\sum_i f_i^2. \qquad \square$$

Lemma 2. *The average cost is given by* $\langle C\rangle = 1/(2n(n-1))F_1D_1$.

Proof. By definition the average cost is $\langle C\rangle = \sum_\pi C(\pi)/n!$. Therefore, $\langle C\rangle = (1/2n!)\sum_\pi \sum_{i,j} f_{ij}d_{\pi(i)\pi(j)} = (1/2n!)\sum_{i,j} f_{ij}\sum_\pi d_{\pi(i)\pi(j)}$. With i and j fixed we have $\sum_\pi d_{\pi(i)\pi(j)} = (n-2)!\,D_1$ as there are $(n-2)!$ permutations such that $\pi(i)=k$ and $\pi(j)=l$ with k and l fixed. So, $\langle C\rangle = (1/2n!)\sum_{i,j} f_{ij}(n-2)!\,D_1 = F_1D_1/(2n(n-1))$. □

Lemma 3. *The average-squared cost is given by*

$$\langle C^2\rangle = \frac{1}{4n!}\left\{(n-4)!\left(F_1^2+2F_2-4\sum_i f_i^2\right)\left(D_1^2+2D_2-4\sum_i d_i^2\right) +4(n-3)!\left(\sum_i f_i^2-F_2\right)\left(\sum_i d_i^2-D_2\right)+2(n-2)!F_2D_2\right\}.$$

Proof. We have $C^2(\pi) = \frac{1}{4}\sum_{i,j,k,l=1}^n f_{ij}f_{kl}d_{\pi(i)\pi(j)}d_{\pi(k)\pi(l)}$.

Hence, $\sum_\pi C^2(\pi) = \frac{1}{4}\sum_{i,j,k,l=1}^n f_{ij}f_{kl}\sum_\pi d_{\pi(i)\pi(j)}d_{\pi(k)\pi(l)}$.

We distinguish several cases, according to the values of indices i,j,k and l. By using Lemma 1 we obtain:

Case 1: $i\neq j\neq k\neq l$

$$\sum_{i\neq j\neq k\neq l} f_{ij}f_{kl} = F_1^2+2F_2-4\sum_i f_i^2,$$

$$\sum_\pi d_{\pi(i)\pi(j)}d_{\pi(k)\pi(l)} = (n-4)!\sum_{o\neq p\neq q\neq r} d_{op}d_{qr} = (n-4)!\left(D_1^2+2D_2-4\sum_i d_i^2\right).$$

Case 2: $(i=k,j,l)$ or $(i=l,j,k)$ or $(j=k,i,l)$ or $(j=l,i,k)$

$$\sum_{i\neq j\neq l} f_{ij}f_{il} = \sum_i f_i^2 - F_2,$$

$$\sum_\pi d_{\pi(i)\pi(j)}d_{\pi(i)\pi(l)} = (n-3)!\left(\sum_i d_i^2 - D_2\right).$$

Case 3: $(i=k,j=l)$ or $(i=l,j=k)$

$$\sum_{i\neq j} f_{ij}^2 = F_2,$$

$$\sum_\pi d_{\pi(i)\pi(j)}^2 = (n-2)!D_2.$$

Case 4: All others possibilities

$$\sum f_{ij} f_{kl} = 0,$$

$$\sum_{\pi} d_{\pi(i)\pi(j)} d_{\pi(k)\pi(l)} = 0.$$

Thus, we obtain by using the decomposition above

$$\langle C^2 \rangle = \frac{1}{4n!} \left\{ (n-4)! \left(F_1^2 + 2F_2 - 4 \sum_i f_i^2 \right) \left(D_1^2 + 2D_2 - 4 \sum_i d_i^2 \right) + 4(n-3)! \left(\sum_i f_i^2 - F_2 \right) \left(\sum_i d_i^2 - D_2 \right) + 2(n-2)! F_2 D_2 \right\}. \qquad \square$$

Lemma 4. *The average-squared difference cost between two neighboring solutions is given by*

$$\begin{aligned} \langle (C(x) - C(x'))^2 \rangle = 4 \Bigg(& F_2 D_2 n^3 - \left(2F_2 \left(2D_2 + \sum_i d_i^2 \right) + \sum_i f_i^2 \left(2D_2 - \sum_i d_i^2 \right) \right) n^2 \\ & + \left(F_1^2 \left(D_2 - \sum_i d_i^2 \right) + F_2 \left(D_1^2 + 5D_2 + 4 \sum_i d_i^2 \right) - \sum_i f_i^2 \left(D_1^2 - 4D_2 - 3 \sum_i d_i^2 \right) \right) n \\ & + F_1^2 \left(D_1^2 - D_2 - \sum_i d_i^2 \right) - \left(D_1^2 + 2 \left(D_2 + \sum_i d_i^2 \right) \right) \left(F_2 + \sum_i f_i^2 \right) \Bigg) \Big/ (n^2 (n-1)^2 (n-2)(n-3)). \end{aligned}$$

Proof. Let $\delta_{ij}(\pi)$ be the cost of the new solution obtained by exchanging facilities i and j in the current solution π, minus the cost of this current solution. We have $\langle (C(x) - C(x'))^2 \rangle = (\sum_{\pi} \sum_{i \neq j} \delta_{ij}^2(\pi))/(n! n(n-1))$, and so we are going to develop this expression.

We have $\delta_{ij}(\pi) = \sum_{k=1, k \neq i,j}^{n} (f_{ik} d_{\pi(j)\pi(k)} + f_{jk} d_{\pi(i)\pi(k)} - f_{ik} d_{\pi(i)\pi(k)} - f_{jk} d_{\pi(j)\pi(k)})$, and so $\delta_{ij}^2(\pi) = \sum_{k,l=1, k \neq i,j;\, l \neq i,j}^{n} (f_{ik} d_{\pi(j)\pi(k)} + f_{jk} d_{\pi(i)\pi(k)} - f_{ik} d_{\pi(i)\pi(k)} - f_{jk} d_{\pi(j)\pi(k)})$ $(f_{il} d_{\pi(j)\pi(l)} + f_{jl} d_{\pi(i)\pi(l)} - f_{il} d_{\pi(i)\pi(l)} - f_{jl} d_{\pi(j)\pi(l)})$.

We note (1) for $\sum_{\pi} \sum_{i \neq j \neq k \neq l} f_{ik} d_{\pi(j)\pi(k)} f_{il} d_{\pi(j)\pi(l)}$, (2) for $\sum_{\pi} \sum_{i \neq j \neq k \neq l} f_{ik} d_{\pi(j)\pi(k)}$ $f_{jl} d_{\pi(i)\pi(l)}$, (3) for $\sum_{\pi} \sum_{i \neq j \neq k \neq l} f_{ik} d_{\pi(j)\pi(k)} f_{il} d_{\pi(i)\pi(l)}, \ldots,$ (16) for $\sum_{\pi} \sum_{i \neq j \neq k \neq l} f_{jk}$ $d_{\pi(j)\pi(k)} f_{jl} d_{\pi(j)\pi(l)}$.

We note $(1')$ for $\sum_\pi \sum_{i,j,k=l} f_{ik} d_{\pi(j)\pi(k)} f_{il} d_{\pi(j)\pi(l)}$, $(2')$ for $\sum_\pi \sum_{i,j,k=l} f_{ik} d_{\pi(j)\pi(k)} f_{jl} d_{\pi(i)\pi(l)}$, $(3')$ for $\sum_\pi \sum_{i,j,k=l} f_{ik} d_{\pi(j)\pi(k)} f_{il} d_{\pi(i)\pi(l)}, \ldots, (16')$ for $\sum_\pi \sum_{i,j,k=l} f_{jk} d_{\pi(j)\pi(k)} f_{jl} d_{\pi(j)\pi(l)}$.

It it easy to check the following equalities, by using the proof of Lemma 3.

$$\begin{aligned}
\sum_\pi \sum_{i\neq j\neq k\neq l} f_{ik} d_{\pi(j)\pi(k)} f_{il} d_{\pi(j)\pi(l)} &= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{jk} d_{\pi(i)\pi(k)} f_{jl} d_{\pi(i)\pi(l)} \\
&= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{ik} d_{\pi(i)\pi(k)} f_{il} d_{\pi(i)\pi(l)} \\
&= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{jk} d_{\pi(j)\pi(k)} f_{jl} d_{\pi(j)\pi(l)} \\
&= (n-3)(n-3)! \left(\sum_i f_i^2 - F_2 \right) \\
&\quad \times \left(\sum_i d_i^2 - D_2 \right).
\end{aligned}$$

In other words $(1) = (6) = (11) = (16) = (n-3)(n-3)!(\sum_i f_i^2 - F_2)(\sum_i d_i^2 - D_2)$.

$$\begin{aligned}
\sum_\pi \sum_{i\neq j\neq k\neq l} f_{ik} d_{\pi(j)\pi(k)} f_{jl} d_{\pi(i)\pi(l)} &= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{jk} d_{\pi(i)\pi(k)} f_{il} d_{\pi(j)\pi(l)} \\
&= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{ik} d_{\pi(i)\pi(k)} f_{jl} d_{\pi(j)\pi(l)} \\
&= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{jk} d_{\pi(j)\pi(k)} f_{il} d_{\pi(i)\pi(l)} \\
&= (n-4)! \left(F_1^2 + 2F_2 - 4 \sum_i f_i^2 \right) \\
&\quad \times \left(D_1^2 + 2D_2 - 4 \sum_i d_i^2 \right).
\end{aligned}$$

In other words $(2) = (5) = (12) = (15) = (n-4)!(F_1^2 + 2F_2 - 4\sum_i f_i^2)(D_1^2 + 2D_2 - 4\sum_i d_i^2)$.

$$\begin{aligned}
\sum_\pi \sum_{i\neq j\neq k\neq l} f_{ik} d_{\pi(j)\pi(k)} f_{jl} d_{\pi(j)\pi(l)} &= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{jk} d_{\pi(i)\pi(k)} f_{il} d_{\pi(i)\pi(l)} \\
&= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{ik} d_{\pi(i)\pi(k)} f_{jl} d_{\pi(i)\pi(l)} \\
&= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{jk} d_{\pi(j)\pi(k)} f_{il} d_{\pi(j)\pi(l)}
\end{aligned}$$

$$= (n-3)!\left(F_1^2 + 2F_2 - 4\sum_i f_i^2\right) \times \left(\sum_i d_i^2 - D_2\right).$$

In other words $(4)=(7)=(10)=(13)=(n-3)!(F_1^2+2F_2-4\sum_i f_i^2)(\sum_i d_i^2 - D_2)$.

$$\begin{aligned}\sum_\pi \sum_{i\neq j\neq k\neq l} f_{ik} d_{\pi(j)\pi(k)} f_{il} d_{\pi(i)\pi(l)} &= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{jk} d_{\pi(i)\pi(k)} f_{jl} d_{\pi(j)\pi(l)} \\ &= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{ik} d_{\pi(i)\pi(k)} f_{il} d_{\pi(j)\pi(l)} \\ &= \sum_\pi \sum_{i\neq j\neq k\neq l} f_{jk} d_{\pi(j)\pi(k)} f_{jl} d_{\pi(i)\pi(l)} \\ &= (n-3)(n-4)!\left(\sum_i f_i^2 - F_2\right) \\ &\quad \times \left(D_1^2 + 2D_2 - 4\sum_i d_i^2\right).\end{aligned}$$

In other words $(3)=(8)=(9)=(14)=(n-3)(n-4)!(\sum_i f_i^2 - F_2)(D_1^2+2D_2-4\sum_i d_i^2)$.

$$\begin{aligned}\sum_\pi \sum_{i,j,k=l} f_{ik} d_{\pi(j)\pi(k)} f_{il} d_{\pi(j)\pi(l)} &= \sum_\pi \sum_{i,j,k=l} f_{jk} d_{\pi(i)\pi(k)} f_{jl} d_{\pi(i)\pi(l)} \\ &= \sum_\pi \sum_{i,j,k=l} f_{ik} d_{\pi(i)\pi(k)} f_{il} d_{\pi(i)\pi(l)} \\ &= \sum_\pi \sum_{i,j,k=l} f_{jk} d_{\pi(j)\pi(k)} f_{jl} d_{\pi(j)\pi(l)} \\ &= (n-2)(n-2)! F_2 D_2.\end{aligned}$$

In other words $(1')=(6')=(11')=(16')=(n-2)(n-2)!F_2D_2$.

$$\begin{aligned}\sum_\pi \sum_{i,j,k=l} f_{ik} d_{\pi(j)\pi(k)} f_{jl} d_{\pi(i)\pi(l)} &= \sum_\pi \sum_{i,j,k=l} f_{jk} d_{\pi(i)\pi(k)} f_{il} d_{\pi(j)\pi(l)} \\ &= \sum_\pi \sum_{i,j,k=l} f_{ik} d_{\pi(i)\pi(k)} f_{jl} d_{\pi(j)\pi(l)} \\ &= \sum_\pi \sum_{i,j,k=l} f_{jk} d_{\pi(j)\pi(k)} f_{il} d_{\pi(i)\pi(l)} \\ &= (n-3)!\left(\sum_i f_i^2 - F_2\right)\left(\sum_i d_i^2 - D_2\right).\end{aligned}$$

In other words $(2')=(5')=(12')=(15')=(n-3)!(\sum_i f_i^2 - F_2)(\sum_i d_i^2 - D_2)$.

$$\begin{aligned}\sum_{\pi}\sum_{i,j,k=l} f_{ik}d_{\pi(j)\pi(k)}f_{jl}d_{\pi(j)\pi(l)} &= \sum_{\pi}\sum_{i,j,k=l} f_{jk}d_{\pi(i)\pi(k)}f_{il}d_{\pi(i)\pi(l)}\\ &= \sum_{\pi}\sum_{i,j,k=l} f_{ik}d_{\pi(i)\pi(k)}f_{jl}d_{\pi(i)\pi(l)}\\ &= \sum_{\pi}\sum_{i,j,k=l} f_{jk}d_{\pi(j)\pi(k)}f_{il}d_{\pi(j)\pi(l)}\\ &= (n-2)!\left(\sum_i f_i^2 - F_2\right)D_2.\end{aligned}$$

In other words $(4')=(7')=(10')=(13')=(n-2)!(\sum_i f_i^2 - F_2)D_2$.

$$\begin{aligned}\sum_{\pi}\sum_{i,j,k=l} f_{ik}d_{\pi(j)\pi(k)}f_{il}d_{\pi(i)\pi(l)} &= \sum_{\pi}\sum_{i,j,k=l} f_{jk}d_{\pi(i)\pi(k)}f_{jl}d_{\pi(j)\pi(l)}\\ &= \sum_{\pi}\sum_{i,j,k=l} f_{ik}d_{\pi(i)\pi(k)}f_{il}d_{\pi(j)\pi(l)}\\ &= \sum_{\pi}\sum_{i,j,k=l} f_{jk}d_{\pi(j)\pi(k)}f_{jl}d_{\pi(i)\pi(l)}\\ &= (n-2)(n-3)!F_2\left(\sum_i d_i^2 - D_2\right).\end{aligned}$$

In other words $(3')=(8')=(9')=(14')=(n-2)(n-3)!F_2(\sum_i d_i^2 - D_2)$.

We obtain finally,

$$\begin{aligned}\sum_{\pi}\sum_{i,j}\delta_{ij}^2(\pi) &= (1)+(2)-(3)-(4)+(5)+(6)-(7)-(8)-(9)-(10)\\ &\quad +(11)+(12)-(13)-(14)+(15)+(16)\\ &\quad +(1')+(2')-(3')-(4')+(5')+(6')-(7')-(8')-(9')\\ &\quad -(10')+(11')+(12')-(13')-(14')+(15')+(16')\\ &= 4((1)+(1')+(2)+(2')-(3)-(3')-(4)-(4')),\end{aligned}$$

and the lemma is proved. □

We have proved the following proposition with Lemmas 2–4.

Proposition 1. For the quadratic assignment problem, let $F_k = \sum_{i,j} f_{ij}^k$, f_i for the sum $\sum_j f_{ij}$, D_k for the sum $\sum_{i,j} d_{ij}^k$, and d_i for the sum $\sum_j d_{ij}$. Then, the variance of the cost function and the average-squared cost difference between two neighboring

solutions, for the 2-exchange neighborhood, are given by

$$\begin{aligned}\mathrm{Var}(C) = \Bigg\{ & F_2 D_2 n^4 - 2\left(F_2\left(2D_2 + \sum_i d_i^2\right) + \sum_i f_i^2\left(D_2 - \sum_i d_i^2\right)\right) n^3 \\ & + \left(F_1^2\left(D_2 - 2\sum_i d_i^2\right) + F_2\left(D_1^2 + 5D_2 + 4\sum_i d_i^2\right)\right. \\ & \left. -2\sum_i f_i^2 (D_1^2 - 2D_2)\right) n^2 \\ & + \left(F_1^2\left(2D_1^2 - D_2 + 2\sum_i d_i^2\right) - F_2\left(D_1^2 + 2\left(D_2 + \sum_i d_i^2\right)\right)\right. \\ & \left. +2\sum_i f_i^2\left(D_1^2 - D_2 - \sum_i d_i^2\right)\right) n - 3F_1^2 D_1^2 \Bigg\} \\ & /(2n^2(n-1)^2(n-2)(n-3)),\end{aligned}$$

and

$$\begin{aligned}\langle (C(x) - C(x'))^2 \rangle = 4 \Bigg(& F_2 D_2 n^3 - \left(2F_2\left(2D_2 + \sum_i d_i^2\right)\right. \\ & \left. + \sum_i f_i^2\left(2D_2 - \sum_i d_i^2\right)\right) n^2 \\ & + \left(F_1^2\left(D_2 - \sum_i d_i^2\right) + F_2\left(D_1^2 + 5D_2 + 4\sum_i d_i^2\right)\right. \\ & \left. - \sum_i f_i^2\left(D_1^2 - 4D_2 - 3\sum_i d_i^2\right)\right) n \\ & + F_1^2\left(D_1^2 - D_2 - \sum_i d_i^2\right) - \left(D_1^2 + 2\left(D_2 + \sum_i d_i^2\right)\right) \\ & \times \left(F_2 + \sum_i f_i^2\right)\Bigg) \Bigg/ (n^2(n-1)^2(n-2)(n-3)).\end{aligned}$$

From this proposition, the autocorrelation coefficient can be calculated exactly in polynomial time. Moreover, the following theorem can be derived.

Theorem 1. *The autocorrelation coefficient of any instance of the QAP, with the 2-exchange neighborhood, verifies* $\xi \geqslant n/4$.

Proof. By definition

$$\xi = \frac{1}{1-\rho(1)} = \frac{2\,\mathrm{Var}(C)}{\langle (C(x)-C(x'))^2 \rangle},$$

and so

$$\xi \geqslant \frac{n}{4} \Leftrightarrow 2\,\mathrm{Var}(C) - \frac{n}{4}\langle (C(x)-C(x'))^2 \rangle \geqslant 0.$$

It follows from Proposition 1 that this is equivalent (by using mathematical softwares such as Mathematica or Maple) to

$$\frac{(n\sum_i f_i^2 - F_1^2)(n\sum_i d_i^2 - D_1^2)}{n^2(n-1)^2(n-2)} \geqslant 0,$$

which is always true. □

To verify that this bound is sharp, it suffices to consider the matrix F with $f_{ij} = 1/(n-1)$, and $f_{ii} = 0$ otherwise, for $1 \leqslant i \neq j \leqslant n$.

For the upper bound we have no formal proof, but based on numerous experimental results we conjecture that $\xi \leqslant n/2$.

4. Application

As it was reported in the introduction, usually local-search-based heuristics give very good results when applied to the QAP. The autocorrelation coefficient can be used to explain this fact. The low autocorrelation binary string problem has been reported to be, by Beenker et al. in [3], a notorious very hard problem for local search. By results of Stadler in [12] it is straightforward to prove that the autocorrelation coefficient for all its instances is asymptotically $n/8$, which is rather a small value. Also, Johnson in [7] has reported that for the traveling salesman problem local search heuristics give very good results. The autocorrelation coefficient for this problem is $n/2$, a rather large value. Thus, the autocorrelation coefficient of the QAP which is minored by $n/4$ can be considered as a satisfactory value.

In order to have a more convenient parameter to read, and independent of the size of the instance, we define a *ruggedness coefficient*, noted ζ, lying between 0 and 100 (under the assumption $\xi \leqslant n/2$ which is based on experimental results), by putting $\zeta = 100 - 400/n(\xi - n/4)$. If ζ is close to 100 (respectively 0) it means that the autocorrelation coefficient is weak (respectively large) and so the landscape is very steep (respectively flat).

The computational results presented in the next section confirm the link between the ruggedness of a landscape and its hardness for local search algorithms.

5. Computational results

We have chosen to generate QAP instances with known optimal solution. We have used a slightly modified version of the test problem generator GEN2 [10], to obtain QAP instances with both symmetric matrices F and D.

Algorithm 1. *The symmetric test problems generator*

- Construct matrix F by setting $f_{ij} = 100$ if $i \neq j$, and $f_{ij} = 0$ otherwise. For $i < j$, generate randomly d_{ij} with an appropriate distribution (to be explained later: procedure 1), and set $d_{ji} = d_{ij}$. Set $d_{ii} = 0$.
- Sort d_{ij}, for $i < j$, ascendingly and store the rank of d_{ij} in r_{ij} in increasing order.
- Randomly generate integers x_i, $1 \leqslant i \leqslant n(n-1)/2$, with a uniform distribution in [0,100], and sort them ascendingly.
- For each f_{ij}, $i < j$, set $f_{ij} = f_{ij} - x_{r_{ij}}$ and $f_{ji} = f_{ij}$.

The identity permutation is the global optimum for this generated symmetric QAP.

For the construction of the matrix D, we have used a method of Taillard [13]. Its advantage, is that we can generate instances with a nearly complete range of values for the ruggedness coefficient. The matrix D is euclidean, and represents the integral rounded distances between n points of the plane generated according to the following procedure.

Procedure 1. *The generation of locations for the matrix D*

Repeat n/c times:

- Choose Θ randomly, uniformly between 0 and 2π.
- Choose R randomly, uniformly between 0 and M.

  Repeat c times:
  - Choose θ randomly, uniformly between 0 and 2π.
  - Choose r randomly, uniformly between 0 and m.
  - The euclidean coordinates of the next generated point are $(R\cos\Theta + r\cos\theta,\ R\sin\Theta + r\sin\theta)$.

In other words, we generate clusters of c points that are uniformly distributed in a circle of radius M, the points in the clusters being uniformly distributed in a circle of radius m. This nonuniform distribution of distances makes the QAP problems closer to real-life problems [12]. For our experiments the size of instances is set to 20, and we have chosen $M = 40$, $m = 10$, and $c = 5$.

For our local-search-based heuristic, we have chosen the simulated annealing implementation of Johnson et al. [8]. Its robustness allows us to avoid the problem of tuning numerous parameters, otherwise some instances would have been penalized and others favored.

Table 1
Performance of simulated annealing on various instances of size 20.

ζ	$l = 50$		$l = 100$	
	% Rel. error	nbr steps	% Rel. error	nbr steps
$10 \leqslant \zeta < 20$	0.2	50 500	0.1	101 395
$20 \leqslant \zeta < 30$	0.3	53 300	0.2	106 890
$30 \leqslant \zeta < 40$	0.3	58 700	0.2	118 760
$40 \leqslant \zeta < 50$	0.5	62 700	0.3	126 395
$50 \leqslant \zeta < 60$	0.7	66 100	0.4	133 055
$60 \leqslant \zeta < 70$	1.0	75 300	0.6	151 870
$70 \leqslant \zeta < 80$	1.3	76 800	1.0	155 230
$80 \leqslant \zeta < 90$	1.9	79 700	1.4	159 840
$90 \leqslant \zeta < 100$	2.0	82 400	1.8	165 610

At each step two facilities i and j are chosen at random, and the change δ_{ij} in the cost function, of swapping them is computed (in linear time). The swap is accepted if, either $\delta_{ij} \leqslant 0$, or $X \leqslant e^{-\delta_{ij}/T}$ with X being a random real number drawn from the uniform [0,1] distribution, and T is a parameter, called the temperature, which decreases every fixed number (called the temperature length) of steps in a geometric way, i.e. $T \leftarrow rT$, with r the geometric cooling ratio.

The initial temperature is experimentally fixed in such a way that the fraction of accepted moves is between 35% and 45%. The temperature length is set to be $l \times$ instance size, with $l = 50$ and 100, and the geometric cooling ratio is 0.95. When at the end of a temperature the percentage of accepted moves is less than 2%, it means that the search is going to stop soon, because no moves will be accepted. If such an observation occurs five times, then we consider the search process as being "frozen", and the simulated annealing stops. There is an exception if a solution better than the previous best one is found, in that case we wait again for five new low-acceptance temperature completions, to stop the algorithm.

The final result is the best solution found during the entire search. For each instance, we have performed 10 trials, starting from random permutations, and for each category of problems we have generated 100 instances. The results are reported in Table 1. The suitability of the landscape is measured in terms of the quality of obtained solutions, and time spent.

Several conclusions can be drawn from these results. First, we were not able to generate instances with $0 \leqslant \zeta < 10$. It supports our belief that $\zeta \geqslant 0$, or in an equivalent way $\xi \leqslant n/2$. Notice, that both relative error and number of steps are clearly increasing with regard to the ruggedness coefficient and this for both cases with l=50 and 100. The performance of simulated annealing is the direct function of the ruggedness coefficient. The relative error monotonically increases in a very important way, when the landscape becomes more and more rugged. This degradation of results cannot be imputed on a less number of iterations, indeed notice that this number increases along with the ruggedness

coefficient, and so these results are effectively an evidence of the well (respectively bad) suitability of a flat (respectively rugged) landscape for simulated annealing.

6. Conclusion

The ruggedness coefficient introduced in this paper gives a theoretical justification of the effectiveness of local-search-based heuristics for the QAP.

We think that this parameter could be further exploited in order to define appropriate cooling schedules in the simulated annealing algorithm, and also in conjunction with the well-known dominance parameter would allow us to design a complexity measure for characterizing the precise difficulty of QAP instances.

References

[1] E. Angel, V. Zissimopoulos, Autocorrelation coefficient for the graph bipartitioning problem, Theoret. Comput. Sci. 191 (1998) 229–243.
[2] E. Angel, V. Zissimopoulos, On the quality of local search for the quadratic assignment problem, Discrete Appl. Math. 82 (1998) 15–25.
[3] G.F.M. Beenker, T.A.C.M. Claasen, P.W.C. Hermens, Binary sequences with a maximally flat amplitude spectrum, Philips J. Res. 40 (1985) 289–304.
[4] D.T. Connolly, An improved annealing scheme for the qap, European J. Oper. Res. 46 (1990) 93–100.
[5] M. Dorigo, V. Maniezzo, A. Colorni, Algodesk: an experimental comparison of eight evolutionary heuristics for the quadratic assignment problem, European J. Oper. Res 81 (1995) 188–204.
[6] M.R. Garey, D.S. Johnson, Computers and Intractability – a Guide to the Theory of NP-Completeness, W.H. Freeman and Company, SanFrancisco, CA, 1979.
[7] D.S. Johnson, Local optimization and the traveling salesman problem, ICALP 90 Automata, Languages and Programming, 1990, pp. 446–461.
[8] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning, Oper. Res. 37 (6) (1989) 865–892.
[9] T.C. Koopmans, M. Beckmann, Assigment problems and the location of economic activities, Econometrica 25 (1957) 53–76.
[10] Y. Li, P.M. Pardalos, Generating quadratic assignment test problems with known optimal permutations, Comput. Optim. Appl. 1 (1992) 163–184.
[11] S. Sahni, T. Gonzalez, P-complete approximation problems, J. ACM 23 (1976) 555–565.
[12] P.F. Stadler, Landscapes and their correlation functions Tech. Rep. 95-07-067, Santa Fe institute, Santa Fe, NM, 1995.
[13] E.D. Taillard, Comparison of iterative searches for the quadratic assignment problem, Location Sci. 3 (1995) 87–105.
[14] E.D. Weinberger, Correlated and uncorrelated fitness landscapes and how to tell the difference, Biol. Cybernet. 63 (1990) 325–336.