Nonlinear Assignment Problems in Manufacturing

Panos M. Pardalos and Leonidas S. Pitsoulis

Department of Industrial and Systems Engineering
University of Florida, Gainesville, FL 32611 USA
(pardalos@ophelia.ise.ufl.edu), 352-392-9011.

Department of Civil Engineering and Operations Research
Princeton University, Princeton, NJ 08543 USA
(leonidas@dragon.princeton.edu), 609-258-2161.

Nonlinear Assignment Problems (NAPs) are combinatorial optimization problems for which no exact algorithm exists that can solve them in reasonable computational time. They enjoy applications in diverse areas, such as location theory, data association problems, physics, manufacturing and many others. This talk is divided into two parts. First an overview of NAPs will be presented, where different formulations, exact and heuristic solution methods, and other characteristics of NAPs will be discussed. Secondly we will focus on a nonlinear assignment problem which has applications in the manufacturing of jet engines, and namely the Turbine Balancing Problem. An algorithm to find approximate solutions to this problem will be presented, together with computational experience from real data acquired from a major jet engine manufacturer.

Keywords: *Combinatorial optimization, assignment problems, heuristics, turbine balancing problem, data association problem.*

1 Introduction

Nonlinear Assignment Problems (NAP's) are NP-hard combinatorial optimization problems that are natural extensions of the classical Linear Assignment Problem (LAP). NAPs have numerous applications in different fields, ranging from the assignment of facilities to locations in location theory, to the tracking of elementary particles in high energy physics. We begin this section with the most basic of all assignment problems, the linear assignment problem. Consider that we are given n facilities and locations, and a matrix $\mathbb{R}^{n\times n} \ni C := (c_{ij})$ where c_{ij} represents the cost of placing facility i to location j. The objective is to find an optimal assignment of facilities to locations such that the total placement cost is minimized. Using the decision variables $x_{ij} \in \{0,1\}$ defined by

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

we can formulate the LAP as an integer $0-1$ program:

$$\begin{aligned}
\min \quad & \sum_{i,j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n, \\
& \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n, \\
& x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \ldots, n.
\end{aligned}$$

Equivalently using permutations we can formulate the LAP as

$$\min_{p \in \Pi_N} \sum_{i=1}^{n} c_{ip(i)}, \tag{1}$$

where Π_N is the set of all permutations $p : N \to N$, and $N = \{1, 2, \ldots, n\}$. The LAP is a classical optimization problem and can be solved very efficiently. Nonlinear assignment problems are extensions of the LAP, and they can be divided into two major classes, M-adic assignment problems, and Multidimensional Assignment Problems (MPA's).

In the M-adic assignment problem class, the dimension of the cost array in (1) becomes $2M, M \geq 2$, and we permute over M indices of the cost array C in the objective function. For example the 3-adic assignment problem can be formulated as

$$\min_{p \in \Pi_N} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijkp(i)p(j)p(k)}, \tag{2}$$

where $C := (c_{ijklms}) \in \mathbb{R}^{n^{2M}}, M = 3$. In a similar way we can define the quadratic ($M = 2$), and the biquadratic ($M = 4$) assignment problems. In the MAP class, the dimension of the cost array in (1) becomes $M \geq 3$, and we sum over n values of C. However C is permuted over $M-1$ indices by different permutations. For example the three dimensional assignment problem can be formulated as

$$\min_{p,q \in \Pi_N} \sum_{i=1}^{n} c_{ip(i)q(i)}, \tag{3}$$

where $C := (c_{ijk}) \in \mathbb{R}^{n^M}, M = 3$.

In section 2 we will present two representatives of the M-adic assignment problem class, specifically the Quadratic Assignment Problem (QAP) and the BiQuadratic Assignment Problem (BiQAP). Formulations and applications for both problems will be mentioned, and a recent application of the QAP which involves the balancing of turbine engines will be described in detail.

2 The Quadratic Assignment Problem

Consider the 2-adic assignment problem, where given a four-dimensional cost array $C = (c_{ijkl})$ the problem is

$$\min_{p\in\Pi_N} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijp(i)p(j)}. \tag{4}$$

The problem in (4) is the general formulation of the QAP. However the above formulation is rarely used since it has few applications. The QAP is mostly known by its Koopmans and Beckman [3] formulation which is a special case of (4). Consider the problem of allocating a set of n facilities to a set of n locations, with the cost being a function of the distance and flow between the facilities. The objective is to assign each facility to a location such that the total cost is minimized. Specifically, we are given two $n \times n$ input matrices with real elements $F = (f_{ij})$ and $D = (d_{kl})$, where f_{ij} is the flow between the facility i and facility j and d_{kl} is the distance between the location k and location l. The Koopmans-Beckmann version of the QAP can be formulated as follows:

$$\min_{p\in\Pi_N} \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{p(i)p(j)}, \tag{5}$$

where Π_N is the set of all permutations $p : N \to N$. Each individual product $f_{ij}d_{p(i)p(j)}$ is the cost of assigning facility i to location $p(i)$ and facility j to location $p(j)$. In the context of facility location the matrices F and D are symmetric with zeros in the diagonal, and both matrices are nonnegative. We will refer to the Koopmans-Beckman formulation as the *standard* QAP. Clearly, a standard QAP can be formulated as a general QAP by setting $c_{ijkl} := f_{ij}d_{kl}$ for all i, j, k, l. Although extensive research has been done for more than three decades, the QAP, in contrast with its linear counterpart the LAP, remains one of the hardest optimization problems and no exact algorithm can solve problems of size $n > 20$ in reasonable computational time. In fact, the QAP is NP-hard and that even finding an approximate solution within some constant factor from the optimal solution cannot be done in polynomial time unless P=NP.

For a detailed treatment of the QAP the reader is referred to the recent survey by Burkard, Çela, Pardalos and Pitsoulis [1].

2.1 Different Formulations

In this section we will present several formulations of the QAP for the purpose of providing an insight to the nature of the problem. Almost all of the formulations presented are the result of the effort of different researchers to view the structural characteristics of the problem from a different angle.

In the formulation (5), we can associate with each $p \in \Pi_N$ a 0-1 matrix $X = (x_{ij})_{n\times n}$, such that

$$x_{ij} = \begin{cases} 1 & \text{if } p(i) = j, \\ 0 & \text{otherwise.} \end{cases}$$

In other words the elements of X must satisfy the following *assignment constraints*

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, 2, \ldots, n,$$
$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, 2, \ldots, n,$$
$$x_{ij} \in \{0, 1\}, \qquad i, j = 1, 2, \ldots, n,$$

which define the vertices of the well known assignment polytope (other names include: the Birkhoff polytope, the transportation polytope, the perfect matching polytope of $K_{n,n}$ etc.). The matrix X is also called a permutation matrix, and we denote the set of all such permutation matrices by $\mathbf{X}_n \subseteq \{0, 1\}^{n \times n}$. Observe that there is one-to-one correspondence between Π_N and $\mathbf{X}_n$, and the cardinality of both sets in $n!$.

Koopmans and Beckmann introduced the standard QAP as a *quadratic 0-1 integer* program in [3]. The problem can be represented then as an integer program with a quadratic objective function (hence the name quadratic assignment problem) as follows

$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} f_{ij} d_{kl} x_{ik} x_{jl} \tag{6}$$
$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n, \tag{7}$$
$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n, \tag{8}$$
$$x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \ldots, n. \tag{9}$$

The standard QAP can be equivalently formulated in a more compact form if we define an *inner product*. More specifically, let the inner product between matrices $A, B \in \mathbb{R}^{m \times n}$ be defined as

$$\langle A, B \rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij}.$$

We can formulate the standard QAP alternatively as

$$\begin{aligned} \min \quad & \langle F, XDX^T \rangle \\ \text{s.t.} \quad & X \in \mathbf{X}_n. \end{aligned} \tag{10}$$

Another formulation of the QAP which reflects the combinatorial nature of the problem utilizes the *Kronecker Product*. If we let $vec(X) \in \mathbb{R}^{n^2}$ be the vector formed by the columns of a permutation matrix X, the standard QAP can be formulated as

$$\begin{aligned} \min \quad & vec(X)^T (F \otimes D) vec(X) \\ \text{s.t.} \quad & X \in \mathbf{X}_n. \end{aligned} \tag{11}$$

Although operations using the Kronecker product have been studied in detail, the above formulation is rarely used for studying the QAP.

3 The Turbine Balancing Problem

In this section we present the Turbine Balancing Problem (TBP), and show how this problem can be formulated as a QAP. Computational results of a proposed algorithm in [12] are also presented. Turbine engines, either hydraulic, gas or steam powered, are composed of "fans" which rotate at high speeds around the central axis of the turbine. Each fan (see Figure 1) is composed of a number of blades placed around the periphery of a disk at equally spaced points. The masses of the blades are not identical due to errors in the manufacturing process and wear through use. These masses can vary as much as $\pm 5\%$ ([7]). Therefore, the center of mass of the assembled fan depends on the positioning of the blades around its periphery. When the center of gravity of a fan does not coincide with its central axis of rotation then static unbalance occurs. The deviation of the center of gravity of the fan from the axis of rotation is referred as the unbalance of the turbine. Static unbalance is not desirable since it results in vibrations and shortens the life of the equipment.

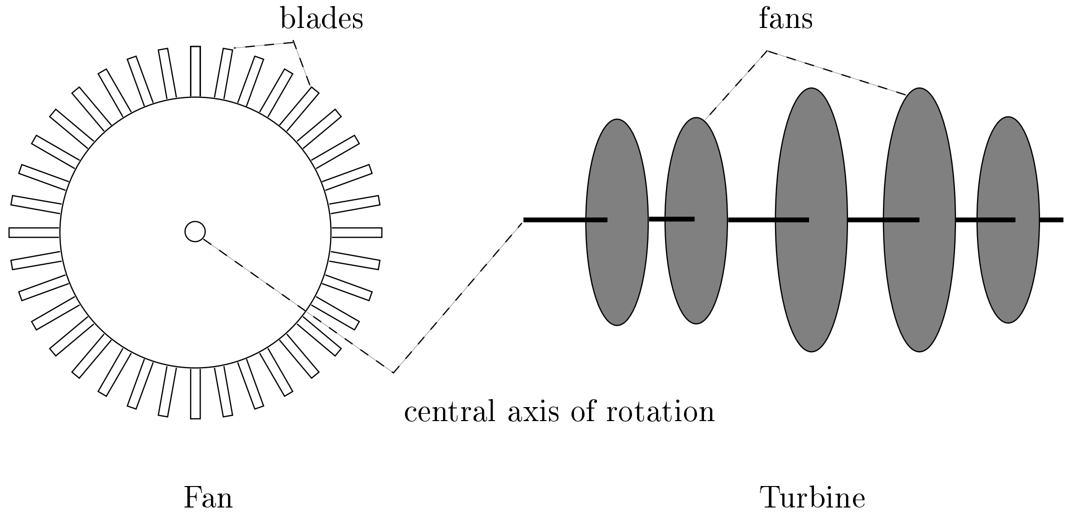

Figure 1: A turbine engine

The current practice in balancing turbines is to find a "good" arrangement of the blades based on a trial and error process and intuition, and correct any residual unbalance by adding weights. This process is time consuming, and it may take even weeks depending on the type of the turbine and the number of fans it contains. The process of balancing a turbine has to be performed in the assembly line, where the turbine is manufactured, and thereafter every time it is maintained. Therefore an efficient way to balance a turbine fast is considered to be very beneficial.

3.1 The TBP as a standard QAP

In this section it will be shown how the TBP can be formulated as a minimization problem which is equivalent to a standard QAP. Before we formulate the TBP as a minimization problem certain assumptions regarding the physical characteristics of the blades and the fan as a whole need to be made. Specifically:

- The weights of the blades are known. There are various methods used in the industry to measure the blade weights.
- The center of gravity for each blade lies at a distance r from the geometrical center of the fan and in the same plane when the blade is placed on the periphery of the fan.
- The center of gravity of the fan (without the blades) lies in its geometrical center and it has zero weight.

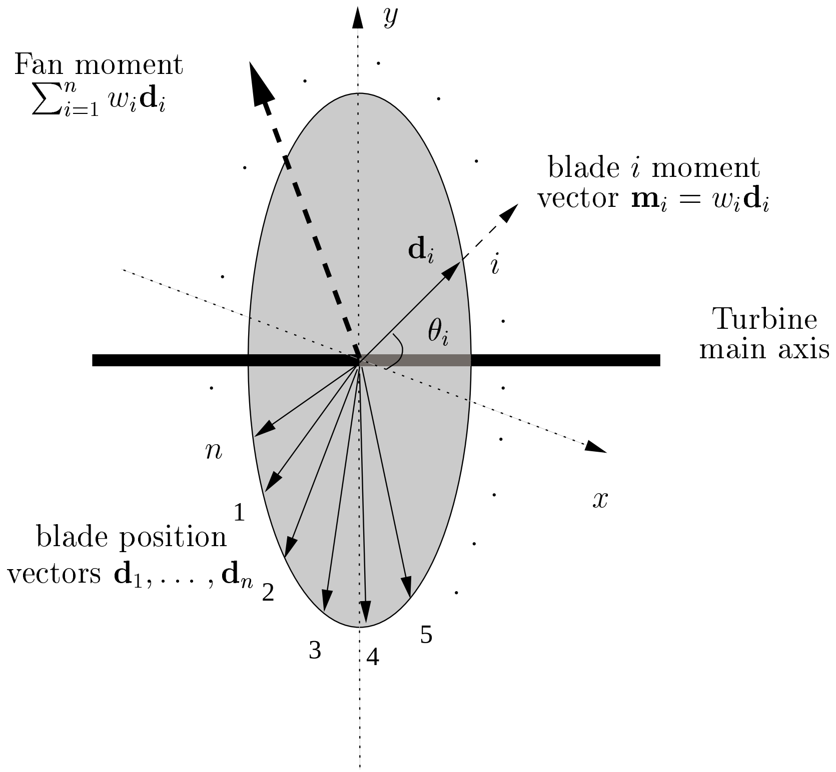

Figure 2: Fan moment vector with n blades

The above assumptions allow us to consider a two-dimensional model of the fan, where the blades are treated as point masses. It is noted that the above assumptions are in accordance with the current method used in the industry to balance turbines.

Consider that we have n blades (see Figure 2), each with a weight $w_i, i = 1, \ldots, n$, to be placed in n positions where each position is described by a vector $\mathbf{d}_i = \binom{r\cos\theta_i}{r\sin\theta_i}, \theta_i = 2\pi(i-1)/n, i = 1, \ldots, n$. Let Π_N be the set of all permutations of the set of integers $N := \{1, 2, \ldots, n\}$. In the TBP we want to find a permutation that minimizes the Euclidean norm of the moment of the fan about its geometrical center

$$\min_{p \in \Pi_N} \left\| \sum_{i=1}^{n} w_i \mathbf{d}_{p(i)} \right\|, \tag{12}$$

where for a given $\mathbf{x} \in \mathbb{R}^n$ the Euclidean norm is defined as $\|\mathbf{x}\| := \sqrt{\sum_{i=1}^{n} x_i^2}$. Each blade position i in Figure 2 is represented by a position vector $\mathbf{d}_i$, and each blade has a given mass w_i. Therefore, the moment vector generated by placing blade with mass w_i in the position described by the vector $\mathbf{d}_i$ is given by $\mathbf{m}_i = w_i \mathbf{d}_i$. The moment vector of the whole disk containing the blades, in other words the fan itself, would be the sum of the individual moment vectors of the blades. The problem is to assign the blades in such a way such that the magnitude of the sum of the individual moment vectors is minimized. A large fan moment vector translates into an off-balance turbine which results in vibrations when the turbine rotates about its central axis.

Recall that given two symmetric matrices in $\mathbb{R}^{n\times n}$, $F = (f_{ij})$ and $D = (d_{kl})$ with nonzero diagonal, the standard QAP is

$$\min_{p\in\Pi_N} \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{p(i)p(j)}. \tag{13}$$

The TBP can be formulated as a standard QAP where the matrices $F = (f_{ij})$ and $D = (d_{kl})$ are formed by the weights and the angles of the blade positions, as shown in the following result [12]

Remark 3.1 *The problem in (12) is equivalent to (13) with $d_{ij} = w_i w_j, f_{ij} = \cos(\theta_i - \theta_j)$ for $i, j = 1, 2, \ldots, n$.*

The turbine balancing problem is NP-Hard.

3.2 Computational Results

A Greedy Randomized Adaptive Search Procedure (GRASP) is developed in [12] for solving the QAP that results from the TBP. Two sets of problems were used for testing the performance of the proposed algorithm. The first set was provided by Pratt & Whitney (P&W) which is a major turbine engine manufacturer. This problem set contains data for the NASA Space Shuttle Main Engine Advanced Turbopump High Pressure Oxygen rotor. Each turbine contains 5 discs (fans) each carrying 54 blades. The results are shown in table 1. The first column contains the name of the turbine. The second column contains the disc number for each turbine, and the fourth column the radius r of the disc. The solution obtained from the current method employed in P&W, and that obtained by GRASP are shown in columns 5 and 7 respectively. For turbine P022-02 the solution obtained by PWBAL, a commercial turbine balancing software distributed by P&W, is given in column 6. In column 8 the number of iterations needed by GRASP to reach the given solution is reported. The maximum number of iterations that GRASP was allowed to run for all the problems was 1000, which required approximately 11 CPU minutes on an Intel Pentium 200MHz processor with 64MBytes of memory. A PC was used for performing the computational experiments, since the balancing of turbines usually occurs in the assembly floor where PC's are available. Comparison between the solution values of GRASP and the current method used at P&W demonstrates that GRASP outperforms by far the current method.

The second set of problems used, comes from the literature and do not correspond to actual data of turbine engines. Introduced by Korenjak and Batagelj [4] it is called the integer turbine balancing problem (ITBP) problem set, and we choose to include it in our computational results since it will illustrate how GRASP compares to other algorithms for solving the

Table 1: Statistics summarizing GRASP runs for the P&W problem set

Turbine	disc	n	r	current	PWBAL	GRASP	iter.
P022-02	1	54	4.5286	0.0059	6.40×10^{-6}	5.16×10^{-6}	874
	2	54	4.2725	0.0057	9.80×10^{-6}	3.50×10^{-6}	91
	3	54	4.5640	0.0165	2.93×10^{-5}	3.67×10^{-6}	416
	4	54	4.2725	0.0022	9.80×10^{-6}	4.69×10^{-6}	276
	5	54	4.5927	0.0040	2.32×10^{-5}	4.94×10^{-6}	40
P022-01	1	54	4.5286	0.0006		6.29×10^{-6}	116
	2	54	4.2725	0.0005		7.07×10^{-6}	652
	3	54	4.5640	0.0164		1.08×10^{-5}	616
	4	54	4.2725	0.0044		8.65×10^{-6}	925
	5	54	4.5927	0.0064		1.07×10^{-5}	103
P024-01	1	54	4.5286	0.0086		1.75×10^{-5}	763
	2	54	4.2725	0.0043		5.28×10^{-6}	175
	3	54	4.5640	0.0048		9.09×10^{-6}	97
	4	54	4.2725	0.0005		9.97×10^{-6}	201
	5	54	4.5927	0.0064		1.15×10^{-5}	904
P030-01	1	54	4.5286	0.0039		5.95×10^{-6}	995
	2	54	4.2725	0.0002		1.30×10^{-6}	165
	3	54	4.5640	0.0043		1.00×10^{-5}	185
	4	54	4.2725	0.0022		1.35×10^{-6}	395
	5	54	4.5927	0.0121		4.94×10^{-6}	773

TBP. An instance of ITBP of size $n \geq 2$ is generated by setting $w_i = i, i = 1, \ldots, n$. Note that by the definition of the TBP, the angles $\theta_i, i = 1, \ldots, n$ are the same for every instance of size n since the blades are placed at equally spaced points. The computational results for the ITBP problem set are shown in table 2. The first column corresponds to the size n of the problem, while the radius of the disc for every problem was set to $r = 1000$. The second column corresponds to the solution provided by the algorithm of Korenjak and Batagelj [4] which we call KBH. Columns 3 through 8 report the solution given by the algorithms by Fathi and Ginjupalli [2]. They propose six heuristic methods based on two general schemes, the placement heuristic (PHk, $k = 1, 2, 3$) and the rotational heuristic (RHk, $k = 2, 3, 4$). The solution found by GRASP and the number of iterations needed are given in columns 9 and 10 respectively. The maximum number of iterations was set at 10000. In column 11 the CPU seconds needed for GRASP to complete 10000 iterations, or find a solution which results in 0 unbalance (the theoretical lower bound) is given. A zero value of CPU seconds means that the CPU time was less than a second. The rotational heuristic RHk only works for ITBPs with n divisible by k, while Korenjak and Batagelj [4] report solutions for ITPB instances with $n \leq 20$. From table 2 we can see that GRASP clearly outperforms the heuristics PHk, $k = 1, 2, 3$. The KBH heuristic produces the same solutions as GRASP for instances with $n \leq 12$, but as n gets larger its solutions deteriorate with respect to the ones given by GRASP. In comparing the RHk heuristics with GRASP we have to mention the following result regarding a combinatorial characteristic of the ITBP. Specifically Fathi and Ginjupalli [2] have shown that an ITBP of size n, has a perfect configuration if n is divisible by 2 but not divisible by 4, or if n is divisible by 3 but not divisible by 9. Furthermore they conjecture that an ITBP of size n has a perfect configuration if n is divisible by k but not divisible by k^2, for any prime number $k \geq 2$. A perfect configuration is an assignment of blades into positions that results in 0 unbalance. The RHk heuristics justify the above conjecture, but perform poorly for values of n that do not satisfy the above condition. GRASP finds a perfect configuration for all the values of n that satisfy the above conditions, except for $n = 15, 20$. Overall GRASP performs better than all of the heuristics mentioned and it also provides best known values for the ITBP instances with $n = 13, 17, 19, 23$. The RHk heuristic seems to be the only competitor, but it can only be applied to ITBPs with n divisible by k. Furthermore the existing algorithm can be modified to solve sparse TBP instances, and it can also be parallelized [10, 11].

Table 2: Statistics summarizing GRASP runs for the ITBP problem set

		Fathi and Ginjupalli						GRASP		
n	KBH	PH1	PH2	PH3	RH2	RH3	RH4	soln	iter.	CPU sec.
5	29.93	29.93	29.93	29.93				29.93	1	1
6	0.00	0.00	0.00	0.00	0.00	0.00		0.00	3	0
7	2.73	2.73	2.73	2.73				2.73	2	7
8	3.64	30.06	3.64	3.64	30.06		30.06	3.64	24	10
9	0.63	13.94	0.63	0.63		20.48		0.63	6	16
10	0.00	0.00	0.00	0.00	0.00			0.00	1	0
11	0.02	2.09	2.09	0.89				0.02	1335	35
12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	46	9
13	0.04	8.00	2.74	1.79				0.03	3438	58
14	0.00	0.00	0.00	0.00	0.00			0.00	1019	40
15	0.26	15.68	0.84	0.38		0.00		0.01	8431	107
16	0.02	7.49	0.54	0.23	0.37		3.10	0.02	3375	143
17	0.05	5.03	0.14	0.11				0.03	3575	154
18	0.03	4.06	0.99	0.92	0.00	0.00		0.00	4037	77
19	0.05	0.73	0.72	0.40				0.01	1799	257
20	0.04	1.94	0.30	0.00	0.00		0.00	0.01	1765	296
21		3.05	0.82	0.75		0.00		0.00	624	20
22		2.95	0.74	0.56	0.00			0.00	3427	141
23		1.63	0.31	0.13				0.00	9848	465
24		2.46	0.87	0.49	0.00	0.00	0.00	0.00	203	11

References

[1] R.E. Burkard, E. Çela, P.M. Pardalos, and L.S. Pitsoulis. The quadratic assignment problem. In Ding-Zhu Du and P.M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 3, pages 241–337. Kluwer Academic Publishers, 1998.

[2] Y. Fathi and K. K. Ginjupalli. A mathematical model and a heuristic for the turbine balancing problem. *European Journal of Operations Research*, 63:336–342, 1993.

[3] T. C. Koopmans and M. J. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

[4] S. Korenjak and V. Batagelj. Turbine balancing problem. Technical report, Department of Mathematics, University of Edvard Kardelj of Eljubljana, Yugoslavia, 1987.

[5] G. Laporte and H. Mercure. Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operations Research*, 35:378–381, 1988.

[6] A. Mason and M. Rönnqvist. Solution methods for the balancing of jet turbines. *Computers and Operations Research*, 24(2):153–167, 1997.

[7] J. Mosevich. Balancing hydraulic turbine runners: A discrete combinatorial optimization problem. *European Journal of Operations Research*, 26:202–204, 1986.

[8] R. A. Murphey, P. M. Pardalos, and L. Pitsoulis. A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. In P. M. Pardalos and Ding-Zhu Du, editors, *Network Design: Connectivity and Facility Location*, volume 40 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 277–302. American Mathematical Society, 1997.

[9] R. A. Murphey, P. M. Pardalos, and L. Pitsoulis. A parallel grasp for the data association multidimensional assignment problem. In *Parallel Processing of Discrete Problems*, volume 106 of *IMA Volumes in Mathematics and its Applications*, pages 159–180. Springer-Verlag, 1998.

[10] P. M. Pardalos, L. S. Pitsoulis, and M. G. C. Resende. A parallel GRASP implementation for solving the quadratic assignment problem. In A. Ferreira and José D.P. Rolim, editors, *Parallel Algorithms for Irregular Problems: State of the Art*, pages 115–133. Kluwer Academic Publishers, 1995.

[11] P. M. Pardalos, L. S. Pitsoulis, and M. G. C. Resende. Algorithm 769: FORTRAN subroutines for approximate solution of sparse quadratic assignment problems. *ACM Transactions on Mathematical Software*, 23:196–208, 1997.

[12] L. S. Pitsoulis. *Algorithms for Nonlinear Assignment Problems.* PhD thesis, Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA, 1998.