Solution Methods for the Balancing of Jet Turbines

Andrew Mason
Mikael Rönnqvist
Department of Engineering Science
The University of Auckland
PB 92019 Auckland, New Zealand

December 7, 1998

Abstract

Turbine balancing is an important and regular maintenance operation at airline companies. Because of manufacturing inaccuracies, variations occur in the weights of the blades that can, in turn, lead to significant out-of-balance forces on the engine. The overall time required for balancing can be significantly decreased if the best placement of turbine blades is first determined mathematically. This problem is formulated as a variation of the quadratic assignment problem and a number of solution schemes are investigated. A neighbourhood search algorithm is found to significantly outperform the other solution approaches when applied to data from a major South Pacific airline. The neighbourhood search algorithm can be combined with various strategies to initialize starting points. The use of starting points obtained from a Lagrangean dual scheme is shown to improve results for large problems.
Scope and Purpose

The balancing of turbines is a regular maintenance operation at airlines. Typically an engine will be dismantled and tested for defects, and any faulty blades replaced by new ones. The turbine then needs to be reassembled and balanced, an operation complicated by differences in blade weights. The overall balancing can be a time consuming task, particularly for those engines where the turbine must be dismantled and reassembled whenever blades are interchanged. The problem can be formulated as a variation of the quadratic assignment problem, a problem known to be NP-hard. Furthermore, experiments have shown that finding solutions to this balancing problem is complicated by the presence of many local optima. We propose a number of local search methods which all outperform the current method used by a major South Pacific airline when tested on a number of problems provided by the airline.
1 Introduction

As part of the maintenance operations of a major South Pacific airline, jet engines must be dismantled, inspected, repaired when required, and then reassembled. During the reassembly process, turbine blades have to be sequenced in some order around the rotational axis of the turbine. Because of manufacturing inaccuracies, variations occur in the weights of the blades. These variations can lead to significant out-of-balance forces on the engine. It is therefore important that an optimal ordering for the blades be found in which blade weight variations combine and cancel to minimize the difference between the center of mass of the blades and the rotational axis of the turbines.

Balancing of high speed machinery has been given some attention in the literature. However, interest has mainly focussed on the underlying theoretical mechanics behind turbine balancing. The literature typically focusses on the measurement of out-of-balance forces and discussion of methods for determining the size and location at which correcting weights should be applied. It is more difficult to find literature describing how blades can be relocated to other positions by using mathematical programming. One reason may be the commercial advantage of such a system. General descriptions about balancing can be found in a number of text books, see for example Goodwin [9], Darlow [4], and Ehrich [6].

The airline is today using a program developed by one of the manufacturers of the engines (and therefore likely to be used by a number of airlines). This system begins by finding an initial placement of the blades and then uses an interchange heuristic to improve the solution. The initial placement is driven by rules of thumb which suggest that heavy weights should be placed on opposite sides, and that heavy and light blades should be placed alternately around the turbine. The interchanging heuristic then considers swapping of blades until no more improvements can be found.

The remaining part of this paper has the following outline. In Section 2 we discuss the overall maintenance operation and balancing problem faced by the airline. In Section 3 we discuss the Quadratic Assignment Problem (QAP), and show that the balancing problem can be modelled as a variation of the standard QAP formulation. In Section 4 we describe various local search algorithms and in Section 5 we discuss how these are related to the current software used by the airline. In Section 6 we discuss two different Lagrangean relaxation strategies for the model and show how these can be used to initialize the local search algorithms. We then in Section 7 discuss a branch and bound solution method which we have tested but gave disappointing results. In Section 8 we make a comparison between different solution methods. In Section 9 we discuss two practical extensions that can easily be handled by the proposed model. Finally, we draw some conclusions in Section 10.

2 Problem description

Although a modern jet engine is a complex piece of engineering, the principle of the jet is a relatively simple application of Newton’s third law of motion: “Every
action has an opposite and equal reaction”. In the jet engine it is the reaction to the exhaust gases shooting backwards that propels the engine forward. A standard jet engine is made up of an air intake, a series of compressors, a combustion chamber, a series of turbines and an exhaust nozzle, as shown in Figure 1.

In a typical jet turbine, air comes into the front of the engine and is compressed by the compressors. The air then passes into a combustion chamber where fuel is added and ignited. This expanding air is then used to drive a series of turbines. These turbines are connected to compressors by shafts and hence drive them. The air then travels out of the back of the engine to provide thrust. A turbine consists of a number of blades placed around a shaft. There are two essential types, fans and compressors. In a fan the blades are quite long and slender while in a compressor they are short and compact, see Figure 2.

When an engine has been in use for a certain maximum time it needs an inspection. At this inspection turbines are broken down and individual turbine blades are checked. If any show sign of wear they are removed and replaced by new blades. Blades are also removed if the blade has been in use close to or more than the limit recommended by the manufacturer. Instead of using a specific time between maintenance operations, an ‘as required’ maintenance scheme can be implemented by
monitoring the exhaust gas temperature to detect engine wear. If this temperature increases above a certain limit it shows that the compressor and turbine fans are wearing out. When this occurs the engine is brought in for an overhaul.

The balancing is a major part of the maintenance operation. When maintenance is carried out on fans or ‘spools’ (banks of blades on a shaft) they must be dynamically balanced by the addition of small correction weights before being reassembled into their modules. This dynamic balancing requires that the spools be loaded into a test bed which then spins the spools up to normal running speed. Sensors are used to register any imbalance or vibration. From the vibration patterns, it is possible to calculate the location of the mass center and to indicate where correction weights should be placed. Once the engine has been fully reassembled it too must be dynamically balanced. If such an imbalance is found, it is generally solved by rotating the modules with reference to each other. This allows the individual imbalance forces to be aligned to reduce the overall imbalance.

The blades on an individual turbine are all slightly different in weight due to manufacturing variation. As a consequence, different orders of placement of the blades around the turbine will give rise to different mass centers. The determination of a correct ordering of blades is an important engineering problem for a number of reasons. Firstly, there is a limit on the number of correction weights that can be used in the subsequent dynamical balancing of the turbine. Indeed, failure to balance the turbine within this additional weight limit requires that the turbine be disassembled and the balancing operation restarted. Secondly, in many cases, the blades can only be put onto the shaft at one point, and must be slid around the shaft in only one direction. This means that to get to a given blade all the blades that were inserted after it must first be removed. Consequently, the reordering of blades can be a very time consuming task. Indeed, we note that in the case of the fan, all the blades must be removed even to place a correction weight. Given these requirements, the overall balancing of one engine typically takes between six to eight hours to complete. Note that as each engine includes approximately 10 turbines, only a fraction of this time is available to solve each turbine balancing problem.

### 3 Mathematical model

In this section we derive a mathematical model for minimizing the out of balance of the mass center, and show that the problem can be represented as a variation of the standard QAP. QAP was first formulated by Koopmans and Beckman [12] in 1957 in the context of facility location problems. As well as its continued use in facility location problems (e.g. see Armour and Buffa [1]), QAP has proved successful in areas such as data analysis [11] and task scheduling on parallel computers [5]. The turbine balancing has an underlying QAP structure that distinguishes it from facility location problems, and, as we will show at the end of this section, it cannot be solved directly using standard QAP codes such as GRASP [14, 20].

QAP is NP-hard, and so the generation of optimal solutions is likely to be computationally prohibitive for large problems. Branch and bound approaches have been suggested by Gilmore [8] and Lawler [13], with more recent work by Ramakr-
ishman et al. [19]. However, these have proved successful for problems of only limited size, and so are not considered in this work. (Note, however, that we do report results on an interesting problem-specific branch and bound implementation.) Consequently, the QAP has a history of solution by heuristic, with pairwise interchange approaches proving most popular; see Nugent et al. [16] and Ritzman [21] for a survey of solution procedures. Recent approaches also include the application of techniques such as simulated annealing and genetic algorithms [10]. An excellent survey and description on QAP can be found in Pardalos et al. [17]. A number of recent articles can also be found in [18]. The results we report in this paper support the use of heuristic solution methods in solving practical QAP formulations.

The airline’s turbine balancing problem can be considered as one of assigning \( n \) point masses (blades) to \( n \) equally spaced positions around the turbine. If unconstrained, an object will naturally rotate about its center of moment. However, a turbine is physically constrained to rotate about its geometrical center. Consequently, if the center of moment and the geometrical centre are not coincident, then an out-of-balance force will be generated that acts on the bearings constraining the turbine’s motion. This force increases as the distance between the turbine’s center of moment and its geometrical centre increases. If this distance is minimised then the turbine is said to be well balanced, and the loads on the bearings are minimised.

As part of the balancing process, the airline measures each blade \( i \) to determine its ‘moment weight’ \( w_i \). These moment weights are calculated such that when blade \( i \) is located at position vector \( \mathbf{p}_i \), the moment vector generated by blade \( i \) about the origin (taken as the geometrical centre of the turbine) is given by \( \mathbf{m}_i = w_i \mathbf{p}_i \).

Let the positions at which blades can be placed be numbered from \( j = 1 \) to \( n \), and let \( \mathbf{p}_j, j = 1, 2, \ldots, n \) be the position vectors of these \( n \) positions. Now, if blade \( i \) is located in position \( j(i) \) (i.e. at position vector \( \mathbf{p}_{j(i)} \)), where \( j(\cdot) \) is some permutation of the positions \( \{1, 2, \ldots, n\} \), then the moment of the turbine about its geometrical centre is given by the vector

\[
\sum_{i=1}^{n} w_i \mathbf{p}_{j(i)}
\]  

We seek a solution that minimises the magnitude of the resulting moment:

\[
\text{minimise } \left| \sum_{i=1}^{n} w_i \mathbf{p}_{j(i)} \right|
\]  

We note that the angle between two consecutive positions is constant, namely \( \Delta \theta = 2\pi/n \). Therefore, for any position \( j \) with angle \( \theta_j = \Delta \theta \ast (j - 1) \), we have \( \mathbf{p}_j = (\cos(\theta_j), \sin(\theta_j)) \) assuming a normalised turbine in which the blades are located at a radius of 1. This gives:

\[
\text{minimise } \sqrt{\left( \sum_{i=1}^{n} w_i \cos(\theta_{j(i)}) \right)^2 + \left( \sum_{i=1}^{n} w_i \sin(\theta_{j(i)}) \right)^2}
\]  

Taking the x and y components separately we see that when blade \( i \) with moment weight \( w_i \) is placed in position \( j \), then its contribution in (3) in the vertical (y) and
horizontal (x) directions respectively is given by

\[ c_{ij} = w_i \cos(\theta_j) \]  \hspace{1cm} (4)

\[ d_{ij} = w_i \sin(\theta_j) \]  \hspace{1cm} (5)

The decision problem we need to solve is to determine where each blade should be allocated. We model this by introducing 0/1 variables \( x_{ij} \) defined as

\[ x_{ij} = \begin{cases} 
1 & \text{if blade } i \text{ is allocated to position } j, \\
0 & \text{otherwise.} 
\end{cases} \]  \hspace{1cm} (6)

Now, for any solution \( \mathbf{x} \) we can determine the moment vector \((m_x, m_y)\) of the moment weight as

\[ m_x = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \] \hspace{1cm} (7)

\[ m_y = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}. \] \hspace{1cm} (8)

Thus we wish to solve the following:

\[ \text{minimise } \sqrt{m_x^2 + m_y^2} \] \hspace{1cm} (9)

Minimising the out-of-balance is equivalent to minimising its square, and so the overall problem can be formulated as follows.

\[ [\mathbf{P}] \quad \text{min } \quad mc = m_x^2 + m_y^2 \] \hspace{1cm} (10)

\[ \text{s.t. } \quad m_x = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \] \hspace{1cm} (11)

\[ m_y = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \] \hspace{1cm} (12)

\[ \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \] \hspace{1cm} (13)

\[ \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \] \hspace{1cm} (14)

\[ x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n. \] \hspace{1cm} (15)

The assignment constraints (13) and (14) restrict each blade to one position and each position to have one blade, respectively. Problem \([\mathbf{P}]\) is a Non-Linear Integer Programming (NLIP) problem. Note that while we minimize \(m_x^2 + m_y^2\), the solutions we report are the actual out-of-balance distance being the square root of the objective.

Problem \([\mathbf{P}]\) is not a standard QAP formulation as it includes additional constraints defining \(m_x\) and \(m_y\). These can be eliminated through substitution, giving
the new objective function

\[
\min \left( \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \right)^2 + \left( \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \right)^2 \\
= \left( \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} (c_{ij} c_{kl} + d_{ij} d_{kl}) x_{ij} x_{kl} \right).
\]

(16) \hspace{1cm} (17) \hspace{1cm} (18)

It is important to compare this with the literature-standard QAP application, the facility location problem. In this problem, an assignment of facilities to locations is sought which minimises the product of material flow and distance between the facilities. This gives an objective function

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} (f_{ik} d_{jl}) x_{ij} x_{kl}.
\]

(19) \hspace{1cm} (20)

where $D = (d_{jl})$ represents the distance between location $j$ and $l$, and $F = (f_{ik})$ is the predetermined flow between locations $i$ and $k$.

Comparing the two above objectives, we see that the facility location and turbine balancing objectives are not equivalent. In particular, the turbine balancing objective cannot be represented in terms of the flow matrix $F$ and the distance matrix $D$ used in standard QAP, and thus the turbine balancing problem is not amenable to standard QAP heuristics such as GRASP [14, 20]. Both are instances of the most general QAP objective [13]:

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} e_{ijkl} x_{ij} x_{kl}.
\]

(21) \hspace{1cm} (22)

Note that this general objective introduces $n^4$ coefficients, and is clearly more general than would be wanted for the solution of turbine balancing problems.

A further practical complication in the solution of turbine balancing problems is the range of numerical values that need to be handled. While the actual blade weights can be considerable, the out of balance objective that is then achieved will typically be very small, often a factor of $10^{-9}$ less than the input weights. As is discussed in Section 7, this can lead to difficulty in numerical schemes such as branch and bound. This range of magnitudes would also cause problems if the data were to be integerised, such as for use in a modified GRASP package.

### 4 Local search

A well known method for improving solutions for large scale scheduling problems (e.g., the travelling salesman problem) is the so-called 2-swap (or 2-interchange) method, see e.g. Nemhauser and Wolsey [15]. Applying this method to our problem results in testing the change in objective function value that occurs when the
positions of two blades are interchanged. The number of swaps to be evaluated in every iteration is \(n(n - 1)/2\). All such blade swaps can be tested and then the one giving the best reduction of objective function value can be chosen; this gives a steepest descent (SD) algorithm. A second approach is to simply accept each improvement as it is found, giving a next descent (ND) method. Either of these approaches can then be repeated until no more improvements are possible, at which stage we have a ‘2-swap optimal’ solution.

An extension to the 2-swap improvements is to consider a 3-rotation in which blades \(i_1\), \(i_2\) and \(i_3\) assumed to be in positions \(j(1)\), \(j(2)\) and \(j(3)\) are allocated first to positions \(j(2)\), \(j(3)\) and \(j(1)\) respectively, and then to \(j(3)\), \(j(1)\) and \(j(2)\) respectively, with the better improvement (if any) being recorded. We note that a method combining repeated 2-swaps and 3-rotations considers all 3! possible arrangements for each set of 3 blades.

An important feature of any local search procedure is the efficiency with which the objective change can be calculated after making a swap or other move. The structure of the turbine balancing objective makes it particularly suited to partial recalculation, and indeed this allows candidate improvements to be costed more efficiently than is possible in the standard facility location QAP. When two blades are swapped, it is easy to see from (11) and (12) that \(m_x\) and \(m_y\) can be updated using simple addition and subtraction operations. The actual quadratic objective can then be updated using a simple difference expansion of (10).

A number of computer runs were made in which random solutions to a 38-blade example provided by the airline were repeatedly generated and then improved using either a steepest or next descent algorithm incorporating either 2-swaps, 3-rotations, or a combination of these, until a local optimum was found. (The 2-swap/3-rotation combination approach used 2-swap descents until no further improvements could be made, then switched to 3-rotation and ran this until no further improvements were made, then switched back to 2-swap and so on until neither the 2-swap or 3-rotation searches generated any new solutions.) We found that all descents terminated within very few improvement steps (typically 7 or less). Figure 3 shows the best solutions found as a function of the time allocated to the search; all results are averaged over ten repetitions of each experiment.

Firstly, we note that all 6 approaches were successful in generating improved solutions upon repeated application of the descent method. This suggests that the solution space contains a large number of local optima for all of the neighbourhood definitions. We note, however, that the next- and steepest-descent 2-swap algorithms appear to be best for this problem. A comparison of the number of ‘generate random starting solution and descend’ iterations varied significantly between the different solution methods, with the algorithms using 3-rotations performing significantly fewer such iterations per second. The increased computational requirements of searching the 3-rotation neighbourhood do not appear to be rewarded with improved solution quality. It is interesting to note that an overall best solution of 0.00052 was found during one of the 2-swap steepest descent experiments performed; this is substantially better than the average value obtained, giving further evidence for the large number of local minima and their range of quality.
The original airline solution method was based upon a solution generated as follows. Firstly the blades were sorted by weight so that $w_i \geq w_{i+1}$. Then, blade 1 (the heaviest) was placed in position 1 while blade 2 (the second heaviest) was placed opposite in position $n/2$. (Note that this method assumes an even number of blades.) Next blade $n$ (the lightest) was placed in position 2, and blade $n-1$ was placed opposite in position $(n/2) + 1$. The process was then repeated with blade 3 being placed in position 3 and so on. A partial solution is shown in Figure 4 for a 12 blade problem. The solution so generated was then improved using a 2-swap SD algorithm. The airline solution for the 38 blade problem is 0.051385, about 100 times worse than the best solution generated above.

Figure 4: The initial distribution used by the airline.

6 Lagrangean relaxation

Lagrangean relaxation has proven to be a successful approach for many large scale integer programming problems, see e.g. Fisher [7]. This is because many problems have a structure that, when constraints are relaxed, separates into smaller well defined problems that are solvable with efficient methods. In our problem we can relax the constraints for the calculation of the mass center, i.e. constraints (11)
and (12). The so-called Lagrangean subproblem now becomes the problem of minimizing the Lagrangean function $L(x, m, \lambda)$,

\[
\begin{align*}
[\text{LS}] \\
\min & \quad L(x, m, \lambda) = m_x^2 + m_y^2 + \lambda_1 \left( \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} - m_x \right) + \lambda_2 \left( \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} - m_y \right) \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \\
& \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\
& \quad x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n.
\end{align*}
\]

Here $\lambda_1$ and $\lambda_2$ are the Lagrangean multipliers associated with constraints (11) and (12). $[\text{LS}]$ separates into three subproblems, namely

\[
\begin{align*}
[\text{LS-mx}] & \quad \min m_x^2 - \lambda_1 m_x \\
[\text{LS-my}] & \quad \min m_y^2 - \lambda_2 m_y \\
[\text{LS-x1}] & \quad \min \sum_{i=1}^{n} \sum_{j=1}^{n} \left( \lambda_1 c_{ij} + \lambda_2 d_{ij} \right) x_{ij} \\
\text{s.t.} \quad & \quad \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \\
& \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\
& \quad x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n.
\end{align*}
\]

The solutions for $[\text{LS-mx}]$ and $[\text{LS-my}]$ are uniquely determined by $m_x = \frac{1}{2} \lambda_1$, and $m_y = \frac{1}{2} \lambda_2$. (We thus see that the Lagrangean multipliers can be interpreted as specifying the current center of mass.) Problem $[\text{LS-x1}]$ is a linear assignment problem, and so can be easily solved using standard solution techniques, see e.g. Nemhauser and Wolsey [15].

When the subproblems are solved the Lagrangean multipliers (or dual variables) are updated with an updating formula

\[
\begin{align*}
\lambda_1 &= \lambda_1 + t \left( \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}(\lambda) - m_x(\lambda) \right) \\
\lambda_2 &= \lambda_2 + t \left( \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}(\lambda) - m_y(\lambda) \right),
\end{align*}
\]

where $x_{ij}(\lambda)$, $m_x(\lambda)$, and $m_y(\lambda)$ are the solutions obtained from the subproblem $[\text{LS}]$, and $t$ is a step size. Various selection of step sizes can be found in Bazarra et al. [2]. The updating can be viewed as taking a step in the dual space in order to solve
the Lagrangean dual problem of maximising $\Phi(\lambda)$ where $\Phi(\lambda) = \min L(x, m, \lambda)$ is the solution to the Lagrangean subproblem.

This Lagrangean approach generates a lower bound on the optimal objective function value, and normally a heuristic is used to find a corresponding upper bound because the subproblem solution is normally infeasible. In our case each solution from the subproblem is feasible, and hence we do not need to employ a heuristic. However, our experience with this relaxation is that it produces poor solutions; Table 1 (columns LBD1 and UBD1) show results for a 10 blade example and the 38 blade example considered earlier. The 10 blade problem was generated with random blade moment-weights averaging around 100 units. It is easy to give a practical lower bound of 0.0 since we have a quadratic objective function. The lower bounds generated are therefore very poor. The upper bound is even worse when compared to the solutions generated by the existing airline method and the local search methods. This observation can be explained by the fact that [LS-x1] generates a solution which makes an attempt to move the mass center to the opposite side of the current mass center. Roughly speaking it tries to put the heaviest blades on the opposite side of the mass centre, and all the light blades on the same side as the mass centre. The sub-problems hence generate a series of oscillating 'bang-bang' solutions. Also, because the objective function is linear, there is no possibility of limiting the change in the solution as is possible in a non-linear optimisation with a line search or trust region approach. This is therefore a classic example of ineffective control of the sub-problem by the dual multipliers. Convergence occurs only as a consequence of the reducing step size in the updating of the multipliers.

<table>
<thead>
<tr>
<th>Example</th>
<th>LBD1</th>
<th>UBD1</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 blades</td>
<td>-5.453</td>
<td>87.33</td>
</tr>
<tr>
<td>38 blades</td>
<td>-156.7</td>
<td>3871.7</td>
</tr>
</tbody>
</table>

Table 1: Lower and upper bounds generated using relaxation strategy 1 after 2000 iterations.

Table 1 shows that the lower bounds provided by the relaxation were all negative, and consequently of little practical value. In fact, as the following proof shows, subproblem [LS-x1] can never generate a non-negative objective.

**Theorem:** The objective function for subproblem [LS-x1] is bounded from above by zero.

**Proof:** Firstly, let $a_{ij} = \lambda_1 c_{ij} + \lambda_2 d_{ij}$, and note that from the symmetry in the definition of the problem, $\sum_j c_{ij} = \sum_j d_{ij} = 0$, $i = 1, \ldots, n$, $j = 1, \ldots, n$, and hence $\sum_j a_{ij} = 0$. Now, the LP relaxation of subproblem [LS-x1] is naturally integer, and so is equivalent to [LS-x1]. The dual to this LP is given by

$$\text{max } \sum_{i=1}^n \alpha_i + \sum_{j=1}^n \gamma_j \quad \text{s.t. } \alpha_i + \gamma_j \leq a_{ij} \quad i = 1, \ldots, n, \ j = 1, \ldots, n$$
But, [LS-dual] is bounded above by its aggregated relaxation [Dual-relax]:

\[
\text{max } \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{n} \gamma_j \quad \text{s.t.} \quad n \sum_{i=1}^{n} \alpha_i + n \sum_{j=1}^{n} \gamma_j \leq \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}
\]

But, as discussed above, \(\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} = 0\), and so [Dual-relax] has an obvious upper bound of zero. Consequently, subproblem [LS-x1] is also bounded from above by zero. [QED]

Note that we can achieve the upper bound of zero by setting \(\lambda_1 = \lambda_2 = 0\), and then solving [LS-x1]. The solution to this is an arbitrary assignment, and is a further example of the limited control offered by the dual multipliers.

We can make a second relaxation where one set of the semi-assignment constraints is also relaxed. In this relaxation we explicitly keep the constraints stating that each position must have exactly one blade assigned to it. Subproblem [LS-x1] is now replaced by

[LS-x2]

\[
\min \quad n^2 m_x + n^2 m_y - \lambda_1 m_x - \lambda_2 m_y + \sum_{i=1}^{n} \sum_{j=1}^{n} (\lambda_1 c_{ij} + \lambda_2 d_{ij}) x_{ij} + \\
\sum_{i=1}^{n} \mu_i \left( \sum_{j=1}^{n} x_{ij} - 1 \right)
\]

\[
\text{s.t. } \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n
\]

\[
x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n
\]

This problem separates into [LS-mx] and [LS-my] plus \(n\) smaller problems

[LS-x2i]

\[
\min \sum_{j=1}^{n} (\lambda_1 c_{ij} + \lambda_2 d_{ij} + \mu_i) x_{ij}
\]

\[
\text{s.t. } \sum_{j=1}^{n} x_{ij} = 1
\]

\[
x_{ij} \in \{0, 1\}, \quad j = 1, \ldots, n.
\]

Each of these can be solved by inspection. For each \(i\) we just have to determine the smallest reduced cost, i.e.

\[
\tau_i = \min_{j} \{ \lambda_1 c_{ij} + \lambda_2 d_{ij} + \mu_i \}.
\]

The corresponding \(x\) solution for each \(i\) is then found by

\[
x_{ij} = \begin{cases} 
1 & \text{if } \tau_i = \lambda_1 c_{ij} + \lambda_2 d_{ij} + \mu_i, j = 1, \ldots, n, \\
0 & \text{otherwise.}
\end{cases}
\] (25)

Here we generate a subproblem solution which, in most cases, will not be feasible. This can, however, be rectified using a simple heuristic. One example which has worked well in practice for our problem is the following. All blades which have been allocated one position from the subproblem solution are fixed. All other blades are then sorted by their reduced costs. The blades are then taken in a sequential order and each blade assigned to that un-assigned position with the smallest reduced cost. Clearly, a feasible assignment is always generated by this algorithm.
This further relaxation provides a better lower bound than the first Lagrangian relaxation. In general, a more relaxed problem will provide a bound that is no better, and at best the same as the first relaxed problem. However, in this case, the additional relaxation is applied to the linear programming component of the first relaxation, and thus does not worsen the quality of the bound provided. However, in practice, the numerical lower bound is better for the second relaxation because of the improved response of the sub-problem solutions to changes in the dual multipliers.

Table 2 (columns LBD2 and UBD2) gives results for this method. We can clearly see an improvement over relaxation 1. A modification to this approach is to use each of the solutions found as a starting point for the 2 swap SD routine; column Local2 in Table 2 shows the result of this local search. Clearly the 2-swap improved solution is significantly better than the upper bound.

<table>
<thead>
<tr>
<th>Example</th>
<th>LBD2</th>
<th>UBD2</th>
<th>Local2</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 blades</td>
<td>-0.08169</td>
<td>0.36434</td>
<td>0.05573</td>
</tr>
<tr>
<td>38 blades</td>
<td>-3.91801</td>
<td>5.35869</td>
<td>0.00309</td>
</tr>
</tbody>
</table>

Table 2: Solutions obtained with Relaxation 2 after 2000 iterations and the solution obtained after also using a two-swap steepest descent local search.

7 Integer Programming model

Problem $[P]$ is a NLIP problem and cannot be solved using a standard package for IP problems. However, if we state an approximation of the original problem $[P]$ we may be able to solve it as an IP problem. One such approximation is to modify the problem by replacing the objective function with the linear function $m_y$ and introducing the additional constraints

\[ m_y \geq \frac{n}{2}m_x \]  
\[ m_y \leq -\frac{n}{2}m_x. \]

As is shown in Figure 5, these constraints exploit the rotational symmetry to restrict the position of the mass centre to a much reduced feasible region, thereby removing the multiple optimal solutions arising from the symmetries in the problem. Furthermore, under these constraints, $m_x$ will typically be small, and so $\sqrt{m_x^2 + m_y^2} \approx m_y$, i.e. the objective can be approximated by $m_y$. Note that the quality of this approximation increases as the number of blades is increased.

Using a standard IP code as a black box, we managed to solve the 10 blade problem. However, extending this to larger problems led to problems with numerical accuracy (recall that we can find solutions with very small objective function values), and the resulting difficulties with bounding compounded an already excessively large search tree to prevent termination of the code. Clearly, such an
approach would need substantial refinement before being suitable for problems of any size.

8 Numerical results

Our earlier limited tests suggested that local search methods were necessary to generate high quality solutions, with 2-swap SD and 2-swap ND appearing to be the best of the methods tried. To find initial solutions we either randomly generated solutions or we used the feasible solutions generated in the Lagrangean dual scheme. In this section we rigorously test these procedures by using them to solve a large set of test problems.

We have implemented the proposed solution procedures in a combination of FORTRAN and Pascal codes compiled for a VAX Alpha workstation. We have tested the methods on two sets of data. The first set contains 100 problems each with 40 blades, while the second set, also of 100 problems, has 100 blades per problem. These data sets have been generated using the same mean and variances in blade weights as observed in data provided by the airline. The number of constraints in the first set is 82 and the number of decision variables is 1,602. Corresponding numbers for the second set is 202 constraints and 10,002 variables.

Figure 6 gives the results for the six local search methods applied to the first set of problems. As in the earlier experiments, the neighbourhood descent algorithms were run repeatedly from randomly generated starting solutions, and the cumulative time and (possibly improved) best solution so far recorded after each descent. To compensate for problem specific variations, the results at each plotted time were averaged over the 100 different example problems. This averaged overall-best quality is given as a function of CPU time (in Alpha ticks of 0.01 seconds each).
Note that the first few solutions are between 10 and 100 times worse than the final solutions, and so these are not visible on the graphs.

![Figure 6: Plot showing the average objective function value for the 100 test problems in set one (40 blades) with respect to the computational time for the six local search methods.](image)

It is clear that the ND algorithms dominate the others at all the solution times considered. Although the results are very close, it does not appear that the extra computational effort required to find a steepest descent direction gives sufficient gains to justify its use. Furthermore, enlarging the neighbourhood search space by using 3-rotations does not appear to improve the solution quality for the computational times shown. Note that the larger neighbourhood schemes may eventually outperform the 2-swap procedures if the number of allowed iterations is substantially increased. The delays that would be incurred by such increased solution times are not considered acceptable by the airline.

The problems in problem set one were also solved using the airline solution method. This gave an average solution quality over the 100 problems of $2.62 \times 10^{-3}$, about 100 times worse than the best solutions shown in Figure 6. It is interesting to note that the starting solutions generated by the airline heuristic before employing the 2 SD descent had a significantly worse average quality of $3.91 \times 10^{-1}$.

The results obtained from the first set are confirmed by the experiments on the second set of data. Figure 7 gives the results for the six local search methods applied to the second set of problems. Again we can clearly see that the 2-swap methods
outperform the 3-rotation and 2-swap/3-rotation methods. The best method overall is the 2-swap ND. The airline solution method gave an average solution quality over the 100 problems of $3.31 \times 10^{-4}$, about 10 times worse than the best solutions shown in Figure 7. The starting solutions generated by the airline heuristic averaged $3.45 \times 10^{-1}$ in quality before the 2 SD descent was applied.

Figure 7: Plot showing the average objective function value for the 100 test problems in set two (100 blades) with respect to the computational time for the six local search methods.

In the previous examples we have used randomly generated starting solutions. We can also start from feasible solutions generated by the second Lagrangean relaxation. Figure 8 gives a comparison for problem set one between the two best local search methods when using randomly generated starting solutions, and a 2-swap SD method using the solutions from the second relaxation as starting points. Here, we can see that using the Lagrangean starting solutions does not give any improvement in solution efficiency. However, when we consider the 100 blade examples in problem set two, the results are not so clear. Figure 9 shows that the Lagrangean scheme produces starting solutions which improve, at least for low solution times, the efficiency of the descent procedure. This is despite the computational overhead associated with generating the Lagrangean starting solutions, and the corresponding decrease in the total number of descents performed. However, as solution times are increased, the local search methods outperform the Lagrangean method. The Lagrangean scheme can be viewed as an “intelligent” way to construct starting
solutions. In solving the dual problem the multipliers are adjusted so they are close to the final solution. While this is occurring there is a rapid improvement in the objective value. Thereafter the scheme will not produce any better solution because the subproblem will repeatedly generate the same starting solutions. The local search methods on the other hand will continue to search over a large space of starting solutions and, as we observe, eventually find better solutions.

Figure 8: Plot showing the average objective function value for 100 test problems in set one (40 blades) with respect to the computational time using both randomly generated starting solutions and those generated by the Lagrangean dual scheme.

Table 3 compares the average number of iterations completed per second for each of the different solution methods. (Remember that each of these iterations generates a single locally-optimal solution.) We note that the number of descents completed per second for problem set one is substantially greater than that for problem set two with its greater number of blades. This is true even for the next descent methods where the fixed per-step overhead is not proportional to the problem size. This suggests that, as we might expect, the number of swaps required to reach a local minimum is larger for larger problems. We also note the substantial reduction in descents per second when using the Lagrange relaxation to initialise the 2 SD descent method. Given this reduction, it is clear that the Lagrange starting solutions are very good starting points for the descent algorithm. This is despite the earlier results suggesting that these subproblem solutions are not themselves of high quality.
9 Extensions

There are two interesting practical extensions of the model. The first case is to increase the number of available blades above the number of blade positions on the turbine. The second extension is to assume some initial off-balance exists which needs to be compensated for in the blade allocation.

The first case may occur when we have an old set of blades (after removing any which have been defective) and an additional set of new blades. In this case we will typically have a set of more than \( n \) blades, say \( N \) in total, to choose from. When this situation occurs, we will, of course, find a solution that is better than or at least as good as that achievable using a subset of the blades. To incorporate this in the model we introduce \( N - n \) pseudo-positions in the turbine center. A blade placed in any of these positions makes no contribution to the total moment weight; i.e. \( c_{ij} = d_{ij} = 0 \ \forall \ j > n \). The limited number of these centre positions means that we are still forced to locate one blade to each of the other positions. No other changes to the formulation are required.

The second case occurs when we start with a turbine that has a natural non-zero moment weight, denoted by \((m_x, m_y)\), even when unpopulated by blades. Such an imbalance may typically occur as a series of turbines are being successively balanced.
Table 3: Average number of completed descent iterations performed per centisecond for the two problem sets over 250 centiseconds running time under each solution method. A completed descent iteration includes the generation of a starting solution (either randomly or using the Lagrangian scheme), the descent from this solution to a local optimum, and for the Lagrange scheme, the updating of the multipliers.

on the same shaft. In this case we want to balance out this initial imbalance. To incorporate this in the model we simply redefine our objective function to be $\min(m_x - \bar{m}_x)^2 + (m_y - \bar{m}_y)^2$. (Note that if we have more blades than positions, then we need to locate the pseudo positions discussed above at the starting moment weight $(\bar{m}_x, \bar{m}_y)$. The constraint set is the same as before.

10 Conclusions

We have studied the problem of balancing turbines faced by airline companies and suggested a model which can be formulated as a modified quadratic assignment problem that is not amenable to standard (facility-location based) QAP solution packages. We have shown that this problem contains a large number of local optimal solutions, many of which are very close to optimal. We have tested several approaches for searching this space to find high quality solutions.

The existing technique used at the airline is based on an ordering and then a local search strategy using 2-swap. We have shown that this is typically not powerful enough to find a high quality solution. In our examples we have shown that repeated descents can improve solutions by at least an order of magnitude. Steepest- and Next-descent 2-swap approaches appear to be the most efficient. These improvements in solution quality lead to reductions in the overall time taken for turbine balancing.

The first proposed Lagrangean approach fails to produce good solutions; this is explained by a closer look at the different subproblems. A better performance was found by the second relaxation which can be used as an “intelligent” way to generate good starting solutions from which a local search is applied.

We also tested the a formulation of the problem as an IP problem using an interesting problem-specific approximation. However, the use of an IP code as a
black box was very disappointing. This does not appear to be a viable solution method.

11 Acknowledgements

The authors would like to acknowledge the contribution made by Glenn Carter in conducting the initial numerical experiments.

12 About the Authors

A. Mason is a lecturer in the Engineering Science Department in the School of Engineering at the University of Auckland, New Zealand. His current research interests include rostering applications, genetic algorithms, and management decision support systems.

M. Rönqvist is a lecturer in the Engineering Science Department in the School of Engineering at the University of Auckland, New Zealand. His current research interests are applied operations research, real time optimization and scheduling applications.

References


