IE Interfaces
Vol. 14, No. 2, pp. 140-147, June 2001.

A Turbine-Blade-Balancing Problem with Some Locking Blades†

WonJoon Choi
School of Industrial Engineering, University of Ulsan

고정형 블레이드가 있는 터빈의 블레이드 균형화 문제

최 원 준

In the turbine-blade manufacturing industry, turbine-blades are machined and then are assembled to form a circular roll of blades. The roll of blades should be balanced as much as possible, since otherwise the efficiency of the turbine generator might be damaged. A locking blade is a blade whose location is fixed and a non-locking blade is a blade whose location can be freely changed. In this paper, we study methods for balancing the weights of the rotating blades for a turbine where some blades are locking blades. The turbine-blade balancing problem is formulated into a mixed-integer programming problem, which turns out to be NP-hard. A heuristic method based on the number partitioning algorithm is developed and the computational experiments show very promising results.

1. Introduction

A steam turbine may be defined as a form of the heat engine in which the energy of the steam is transformed into kinetic energy by means of expansion through nozzles, and the kinetic energy of the resulting jet is in turn converted into force doing work on rings of blades mounted on a rotating part. In the turbine-blade manufacturing industry, turbine-blades are machined and then are assembled to form a circular roll of blades, as illustrated in <Figure 1>. The roll of blades should be balanced as much as possible, since otherwise the efficiency of the turbine generator might be damaged. However, the blades to be assembled into the same roll are not normally identical in weights and lengths, which makes the balancing problem tedious and difficult. And in some situations, some blades are fixed to pre-specified locations over the rotation axis. A blade is called either a locking blade or a non-locking blade. A locking blade is a blade whose location is fixed and a non-locking blade is a blade whose location can be freely changed. Even though the number of locking blades is normally just a small portion of the whole set of blades, even a single blade can affect the quality of the balancing of the turbine blades, which makes

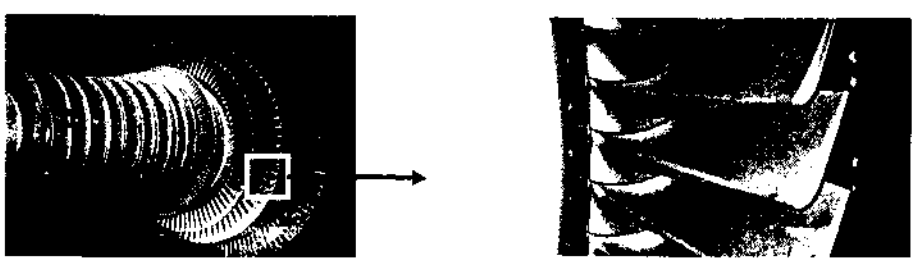
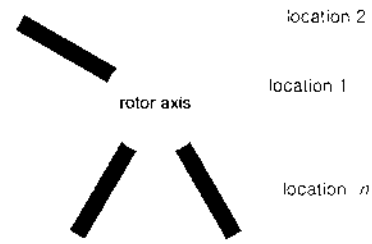


Figure 1. A Turbine and Turbine Blades.

† This research has been financially supported by University of Ulsan made in the program year of 2001.

the balancing problem an important issue.

In this paper, we study methods for balancing the weights of the rotating blades for a turbine where some blades are locking blades. In Section 2, we formulate this blade-balancing problem into a mixed-integer programming problem. In Section 3, we review the literature on topics of the turbine blade balancing problem. In Section 4, we propose a heuristic method for solving the blade-balancing problem, followed by the exposition of the computational experiences in Section 5.



(Colored blades are locking blades.)

Figure 2. Blade Balancing Problem.

2. Problem formulation

Suppose we are given a set of blades whose weights are known. The problem which we should solve is to determine the location of each blade around the rotor axis so as to minimize the *residual unbalance* (its definition will be given later) in weight distribution. A blade is either a locking blade or a non-locking blade. A locking blade is a blade whose location is fixed and a non-locking blade is a blade whose location can be freely changed. As in Amiouny *et al.* (2000), we assume that the centers of gravity of all blades are at the same distance r from the center of the rotor axis. We define the following notation:

n : total number of blades, equivalently total number of locations
i : blade index ($i = 1, 2, ..., n$)
j : location index ($j = 1, 2, ..., n$)
w_i : weight of blade i.
r : distance between the center of gravity of a blade and the center of rotor axis.
F : the set of the locking blades.

For notational simplicity, we assume that a locking blade i in F is fixed at location i. Then F also stands for the set of locations where the locking blades are positioned.

The balancing problem with some locking blades can be stated as follows: Given n blades with weight w_i, the set of locking blades F, and a circle of radius r with n equally spaced locations on its periphery, find an assignment of the non-locking blades to the locations that minimizes residual unbalance about the center. The residual unbalance is the magnitude of the vector sum of the moments created by the individual blades (non-locking blades and locking blades) about the center.

Without loss of generality, we assume that n is even. For convenience, we assume a coordinate system in which the origin is at the center of the circle and the positive x axis goes through one of the n locations as illustrated in <Figure 2>. The locations are numbered in counterclockwise order starting with the one coincident with the positive x axis. So, the coordinates of location j are

$$\left(r\cos\left(\frac{2(j-1)\pi}{n}\right),\ r\sin\left(\frac{2(j-1)\pi}{n}\right)\right),$$
$$j = 1, \cdots, n$$

The decision variables x_{ij} are defined as

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

Let
$$c_{ij} = w_i \cdot r\cos\left(\frac{2(j-1)\pi}{n}\right),$$
$$d_{ij} = w_i \cdot r\sin\left(\frac{2(j-1)\pi}{n}\right)$$

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

$$m_x = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$
$$m_y = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij}$$

Then the balancing problem can be stated as follows :

$$\text{Minimize} \quad \sqrt{m_x^2 + m_y^2} \qquad \text{(Eq. 1)}$$

subject to

$$m_x = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad \text{(Eq. 2)}$$

$$m_y = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \qquad \text{(Eq. 3)}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \cdots, n, \qquad \text{(Eq. 4)}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \cdots, n, \qquad \text{(Eq. 5)}$$

$$x_{ii} = 1, \qquad i \in F \qquad \text{(Eq. 6)}$$

$$x_{ij} = 0, 1 \qquad \text{(Eq. 7)}$$

(Eq. 1) with (Eq. 2) and (Eq. 3) defines the residual unbalance. (Eq. 4) and (Eq. 5) mean the assignment constraints that a blade can be assigned to a single location. And (Eq. 6) stands for the constraints of the locking blades.

It is known that the balancing problem with $F = \phi$ is NP-hard (Mason and Rönnqvist (1997)) and thus the balancing problem with a general form of F is NP-hard.

3. Literature Review

To the knowledge of the author, the turbine blade balancing problem with some locking blades has not been studied in the open literature. So far the turbine blade balancing problem without locking blades has been studied by several authors. Mosevich(1986) presented an algorithm which consisted of selecting the best of a large number of randomly generated solutions. Laporte and Mercure (1988) modeled the problem as a quadratic assignment problem and presented a solution procedure based on Or's TSP heuristic which outperformed that of Mosevich. Fathi and Ginjupalli (1993) also modeled the problem as a quadratic assignment problem and two families of heuristics for it. The first family of heuristics is based on the Placement Heuristic, which places the blades in order of weight the heaviest blade first choosing for each blade the available location that brings the resulting center of gravity as close as possible to the center of rotor axis. The second family of heuristics, based on a divide-and conquer approach called the Rotational Heuristic, which divides the blades into equal-sized subsets, finds good sequences for the smaller problems of balancing with only the blades in each subset and then interleaves the sequences. Mason and Rönnqvist(1997) tested several local search techniques including pairwise interchange and three-way interchange algorithms. They found that pairwise interchange heuristic was most efficient. They also proposed a Lagrangean relaxation approach but the results were not satisfactory.

Amiouny, Bartholdi and Vande Vate (2000) developed several constructive heuristics for the problem of which two seem to dominate, Ordinal Pairing and Greedy Pairing. Both algorithms begin by sorting the blades from heaviest to lightest. Next after forming the pairs of consecutive blades in the sorted list, sort the pairs in the descending order of the difference in weights in a pair. Then pairs in the finally sorted list are placed across from each other on the rotor axis. The two algorithms differ as to how the locations of each pair of blades are determined. Ordinal pairing places the blades in a fixed pattern. Greedy pairing places the blades by a greedy algorithm. For each pair, all possible open positions on the circle are examined, and the position which yields the center of gravity closest to the center of the circle is chosen. Choi *et al.*(1999) presented several heuristic methods for the turbine blade balancing problem formulated as a minimax unbalance problem. Storer (1999) proposed a heuristic which uses an embedded number partitioning algorithm, which turned out to outperform the existing heuristics. Choi (2000) developed an iterative version of Storer's heuristic and found that his heuristic improved the solution quality substantially.

4. Heursitics

In this section, we propose two heuristic methods for the balancing problem: one is based on the number partitioning algorithm and the other is a pairwise interchange heuristic. The first heuristic extends the Storer's embedded number partitioning algorithm to the case of the locking blades. For expositional simplicity, we begin with brief explanation of the Storer's embedded number partitioning algorithm.

Storer's embedded number partitioning algorithm:

Storer(1999) proposed a heuristic which uses an embedded number partitioning algorithm. Initially, the blades are placed in random locations on the circle. Then two perpendicular axes of symmetry as shown in <Figure 3> are chosen. Next the center of gravity around "the X-axis" is balanced, and then

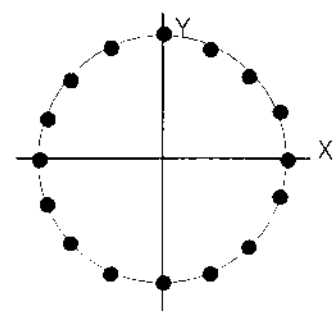


Figure 3. Perpendicular Axes of Symmetry(Storer (1999)).

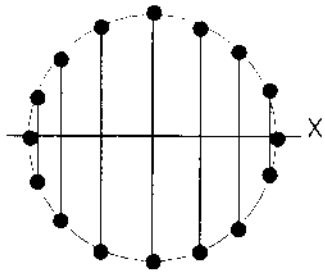


Figure 4. Weight Pairs Symmetric with respect to the X Axis(Storer(1999)).

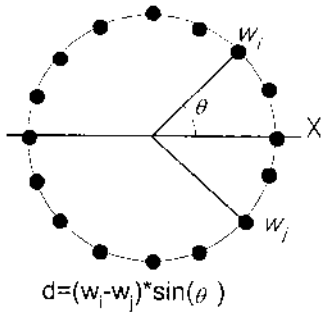


Figure 5. Calculation of d Values(Storer(1999)).

around "the Y-axis."

To balance the center of mass around an axis, pairs of weights which are symmetric with respect to the axis as shown in <Figure 4> are considered. Next a single number d_i for each pair of weights is created (i = 1 to n/2) as illustrated in <Figure 5>. More specifically speaking, we get d_i as follows:

d_i = difference in weights of the pair × sin (difference in angles of the pair / 2).

This number d_i is the center of gravity of the pair with respect to the axis of symmetry. Next a number partitioning algorithm (whose description will be given below) is applied to the set $\{|d_1|, |d_2|, \cdots, |d_{n/2}|\}$ of the pairwise center of mass. The result of partitioning is two sets of pairs. The final step is to arbitrarily select one of the two sets of pairs, and interchange the weights of each pair in that set. The result will be that the center of gravity with respect to the axis of symmetry will be nearly balanced and equal to the objective function found by the number partitioning algorithm. The final step of the algorithm is to balance the center of gravity with respect to the second perpendicular (Y) axis of symmetry. The same algorithm as for the X-axis is applied. A key observation is that applying the algorithm to produce balance around the Y-axis does not affect the balance around the X-axis in the first step.

Partially pre-fixed number partitioning problem:

The number partitioning problem is defined as follows(Karmarkar and Karp (1982)) : Partition a set of numbers into two mutually exclusive sets minimizing the absolute difference of the sum of the two sets. Letting a_i for $i = 1, \cdots, n$ represent n numbers to be partitioned, and S and S' represent the two sets after partitioning, then the problem can be stated as :

$$\text{Min} \left| \sum_{i \in S} a_i - \sum_{i \in S} a_i \right|.$$

Among the algorithms developed for the number partitioning problem, the differencing algorithm proposed by Karmarkar and Karp (1982) is known to be simple and elegant. Letting $a_i = |d_i|$, Storer applied a version of Karmarkar and Karp's algorithm to the turbine balancing problem.

We define the partially pre-fixed number partitioning problem by a number partitioning problem where for a subset of the numbers their belonging sides are pre-specified and cannot be changed.

Example: Suppose that we need partition the numbers in the set {1, 2, 5, 10, 16, 25, 40, 56, 100} under the restriction that {2,10} and {40} must belong to the different sides.

We can get a solution to the partially pre-fixed number partitioning problem by a simple modification of the algorithm for the number partitioning problem. We first sum up the numbers to be pre-fixed for each side and then create an artificial number whose value is the difference between the two sums. Next, we remove the pre-fixed numbers from the number set and insert the artificial number into the number set. Then we apply the number partitioning algorithm to the resulting number set. In this example, we create 40−(2+10)=28, remove 2, 10, 40 and insert 28. And we apply the Karmarkar and Karp's algorithm to {1, 5, 16, 25, 28, 56, 100} and we get the following partition:

{16, 100} & {1, 5, 25, 28, 56}

Then we plug the pre-fixed numbers in the partition by replacing the artificial number with the pre-fixed numbers of the heavier sum and inserting the pre-fixed numbers of the lighter sum into the opposite side of the partition. In this example, we plug the original numbers 2, 10, 40 by replacing 28 with {40} and inserting {2, 10} into the opposite side. Thus we get the final partition

$\{2, 10, 16, 100\}$ & $\{1, 5, 25, 40, 56\}$.

Locking blades version of Storer's embedded number partitioning algorithm:

We can extend the Storer's blade balancing heuristic to the case of the blade balancing problem with some locking blades. Suppose we are given an initial placement of n blades. Then with respect to the current axis, we get the values of $\{d_1, d_2, \cdots, d_{n/2}\}$. Suppose there are k d_is which are associated with the fixed blades. That is, either one or two blades in such k pairs are locking blades. Let us call these d_is by the locking differences. Then we sum up k d_is(let's denote the sum by d_Σ) and merge them into one artificial difference with the value d_Σ. Next, we remove the k locking differences from the number set and insert the artificial difference into the number set. Then we apply the Karmarkar nd Karp's number-partitioning algorithm to the $n/2-k$ $|d_i|$s and $|d_\Sigma|$. After getting a partition of the differences, we place the blades associated with non-locking differences as follows. Suppose that the partition has the form $\{|d_\Sigma|, |d_i|$ for $i \in S_1\}$& $\{|d_i|$ for $i \in S_2\}$ where S_1 and S_2 constitute the index set of the non-locking differences. If the value d_Σ is positive, we place the heavier blades (the lighter blades) associated with non-locking differences d_i for $i \varepsilon S_1$ on the side "above"("below") the axis and reversely for non-locking differences d_i for $i \varepsilon S_2$. If the value d_Σ is negative, we place the heavier blades(the lighter blades) associated with non-locking differences d_i for $i \in S_1$ on the side "below" ("above") the axis and reversely for non-locking differences d_i for $i \varepsilon S_2$.

Example: Suppose that we are given an initial placement where $d_1 = 3.5$, $d_2 = -0.3$, $d_3 = 4$, $d_4 = 5$, $d_5 = -6$, $d_6 = 2$, and $d_7 = 3$ and the last three differences d_5, d_6, and d_7 are the locking differences as illustrated in <Figure 6a>. Then we get $d_\Sigma = d_5 + d_6 + d_7 = -1$. By applying the Karmarkar and Karp's number partitioning algorithm to $\{|d_1|, |d_2|, |d_3|, |d_4|, |d_\Sigma|\}$, we get a partition $\{|d_2|, |d_4|, |d_\Sigma|\}$& $\{|d_1|, |d_3|\}$ as shown in <Figure 6b>. Then since $d_\Sigma = -1 < 0$, we place the heavier blades (the lighter blades) associated with non-locking differences d_2 and for d_4 on the side below (above) the axis and reversely for non-locking differences d_1 and d_3. (Or equivalently, we flip two blades associated with d_5 horizontally, leaving the other blades intact.)

Iterative number partitioning algorithm(locking blades version):

The algorithm to be proposed begins with an arbitrary placement with its centroid (W_x, W_y), and improves the solution iteratively.

While Storer's algorithm first balances the center of gravity around "the X-axis", we select the axis around which the center of gravity will be balanced. The axis will be chosen among axes with angles $\frac{(i-1)\pi}{\pi}$, $i = 1, 2, \cdots, n$. (We will call the axis with the angle $\frac{(i-1)\pi}{\pi}$ by Axis i, for $i = 1, 2, \cdots, n$.) We choose the axis which is nearest to the separating line perpendicular to the line segment linking the center (0, 0) and the centroid (W_x, W_y), as illustrated in <Figure 7>.

To balance the center of mass around the selected axis, pairs of weights which are symmetric with respect to the axis are considered. Next a single number d_i for each pair of weights is created. This number d_i is the center of gravity of the pair with respect to the axis of symmetry. Next the locking blades version of Storer's heuristic is applied to get the partition. The result will be that the center of gravity with respect to the axis of symmetry will be nearly balanced.

The rationale behind the selection of the axis is as follows: The nearer the axis is to the perpendicular separating line, the larger the unbalance of the

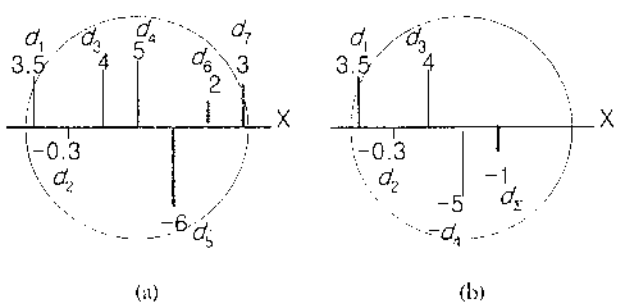


Figure 6. Illustration of the Locking Blades Version Algorithm.

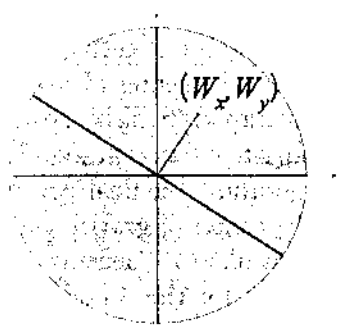


Figure 7. The Separating Line Perpendicular to the Line Segment (0, 0)$-(W_x, W_y)$.

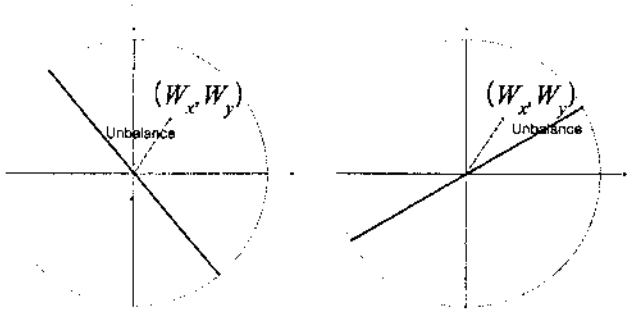


Figure 8. Examples for Illustrating the Relationship between an Axis and Size of Unbalance.

existing placement with respect to the axis is, since the unbalance with respect to an axis is equal to the distance from the center of gravity to the axis. An example is shown in <Figure 8>. One can easily see that the axis of <Figure 8a> is nearer to the perpendicular separating line than the axis of <Figure 8b> and that the unbalance in <Figure 8a> is larger than the unbalance in <Figure 8b>. However it is normally easier to improve a placement with large unbalance than a placement with small unbalance.

After applying the above process, we get a new placement and update the centroid of the placement. Then we select the axis around which the center of gravity will be balanced. If the solution was improved and the newly selected axis is different from the previous axis, then repeat the above process. (For expository simplicity, we call the newly selected axis the anchor axis. Initially the anchor axis is the first selected axis.) Otherwise, we explore other pairings by selecting an axis next to the current axis and then repeat the above process. If we fail again to improve the solution, then we try another untried axis nearest to the anchor axis. The algorithm stops when all axes are tried but the anchor axis is not changed.

The proposed algorithm can be stated in a more formal way as follows:

First we denote the weight of the blade at location i by W_i, for $i=1,2,\cdots, n$.
Get an arbitrary placement $\{W_i\}$ with the centroid (W_x, W_y).
Find out the axis nearest to the line segment linking (0, 0) and (W_x, W_y), say axis φ.
Set successful $\leftarrow$ *true*.
While (successful) do {
  Set $\varphi_{old} \leftarrow \varphi$.
  Get $D[i]$ for each $i=1,2,\cdots, n/2$.
    $D[i] = (W[\varphi+i] - W[\varphi - i+1]) \cdot \sin_i$
      (Note: $W[k] \equiv W[k+n]$ if $k \leq 0$.)
    where $\sin_i = \sin(2\pi/n^*(i-1)+\pi/n)$
  Apply the locking blades version of Storer's heuristic to $\{D[i]\}$ and update the placement $\{W_i\}$.
  Get the new centroid (W_x, W_y) and the nearest axis φ.
  If the solution was improved and φ is different from φ_{old} then
    Set *Alternate* $\leftarrow 1$ & *Explorecount* $\leftarrow 0$.
  Else
    If *Explorecount* $\geq n$ then
      Set *Successful* $\leftarrow$ *false*.
    Else
      Set *Explorecount* $\leftarrow$ *Explorecount* $+ 1$,
      $\varphi \leftarrow \varphi_{old} +$ *Alternate* $*$ *Explorecount*,
      Alternate $\leftarrow$ *Alternate* $*(-1)$.
    Endif
  Endif
}

Here, *Explorecount* denotes the counter that counts the number of axes tried since the last improving axis and *Alternate* is a parameter for regulating the axis number to be explored.

We will call this proposed algorithm the Iterative Method.

Pairwise interchange heuristic(locking blades version):
Pairwise interchange heuristic was known to be very efficient method for the turbine blade balancing problem without a locking blade (Mason and Rönnqvist (1997)). For verifying the performance of the Iterative Method, we use the pairwise interchange heuristic of the locking blades version as a bench mark method. The pairwise interchange heuristic of the locking blades version can be stated as follows:

Get an arbitrary placement $\{W_i\}$.
For each pair of non-locking blades, check if the residual unbalance can be reduced by switching the locations of two blades in the pair. If so, switch them and continue this process. If there is no such pair, then stop.

We will call this pairwise interchange heuristic the Swap Method.

5. Computational Experiences

Following Amiouny, Bartholdi, and Vande Vate

(2000), we generated blade weights from a Normal distribution with a mean of 100 and standard deviation of 5/3. We generated problems over a range of sizes from 20 blades to 200 blades. We tested the problems with the ratio of the number of locking blades to the total number of blades(we will call this ratio the locking ratio from now on) over 0% to 50%. Also following Amiouny, Bartholdi, and Vande Vate (2000), we assume that the circle radius is 100 and the objective function is the Euclidean distance between the center of gravity and the center of the circle. For each problem size, 1000 instances were generated.

<Figures 9> and <Figures 10> show the objective function values averaged over 1,000 problem instances for each problem size with Iterative Method and Swap Method respectively. Iterative Method improves on Swap Method by several orders of magnitude in residual unbalance with a negligible increase in the running time. For a clear-cut comparison between two methods, we extract the results in case of the locking ratio of 10% from <Figures 9> and <Figures 10> and show the comparison in <Figure 11>. For $n \geq 100$, Iterative Method improves on Swap Method by more than order of 3. Also from <Figures 9> and <Figures 10>, we can observe that the quality of a solution gets better as the number of blades increases while the quality of a solution gets degraded as the locking ratio increases.

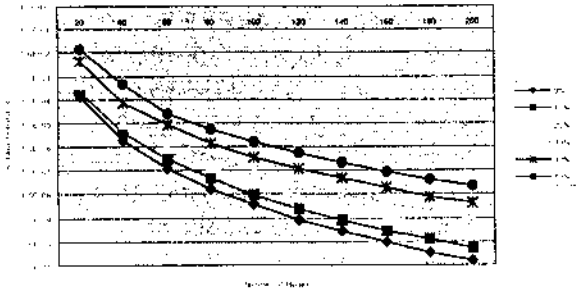


Figure 9. The Residual Unbalance with Iterative Method.

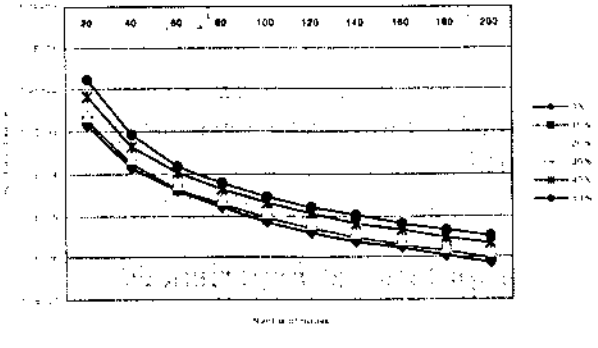


Figure 10. The Residual Unbalance with Swap Method.

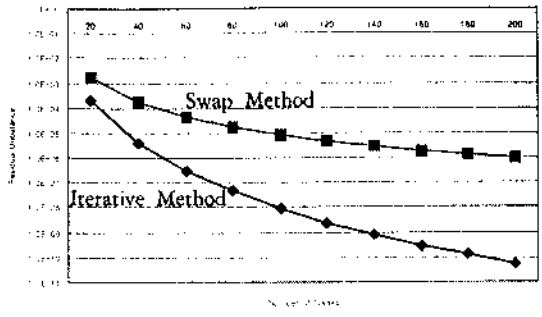


Figure 11. Comparision of Iterative Method and Swap Method for Locking Ratio of 10%.

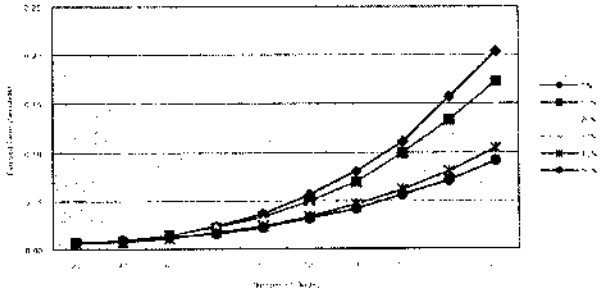


Figure 12. The Elapsed Time with Iterative Method.

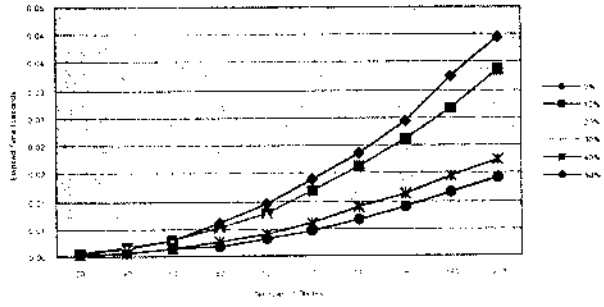


Figure 13. The Elapsed Time with Swap Method.

<Figures 12> and <Figures 13> show the elapsed times in seconds averaged over 1,000 problem instances for each problem size on a PC with a Pentium III 500 MHz processor with Iterative Method and Swap Method respectively. Iterative Method takes larger amount of time than Swap Method. However, the time consumed by Iterative Method is not really a problem in the practical sense. In reality, the typical number of blades in a roll is no greater than 200 and it is not likely that the locking ratio is greater than 10%. For $n = 200$ and the locking ratio = 10%, Iterative Method took just an average of 0.17 seconds.

In summary, Iterative Method shows quite a significant improvement over Swap method and it is handy enough for the interactive use in the real world since it runs fast on a PC.

6. Conclusions

In the turbine-blade manufacturing industry, turbine-blade-balancing problem is an important issue, since poorly-balanced turbines suffer loss in the efficiency and experience the shortening of the economic life. In this paper, we have dealt with a turbine-blade-balancing problem with some locking blades. A locking blade is a blade whose location is fixed and a non-locking blade is a blade whose location can be freely changed. Even though the number of locking blades is normally just a small portion of the whole set of blades, even a single blade can affect the quality of the balancing of the turbine blades.

We proposed an algorithm based on the number partitioning heuristic for a turbine-blade balancing problem with some locking blades. There has been no research reported in the open literature. It turned out that the proposed algorithm improved on a pairwise interchange based heuristic by significant orders of magnitude in the residual unbalance with a negligible increase in the running time. The experimental result that the proposed algorithm gives much smaller residual unbalance than the pairwise interchange based heuristic implies that the objective function surface contains lots of local optimal solutions and the proposed algorithm is effective in exploring the feasible region for searching for good local optimal solutions.

최 원 준
서울대학교 경영학과 학사
한국과학기술원 산업공학과 석사
University of Florida 박사
현재: 울산대학교 공과대학 산업정보경영공학부 교수
관심분야: 물류시스템 설계, 생산일정계획, 시뮬레이션 응용

References

Amiouny, S. V., Bartholdi, III, J. J. and Vande Vate, J. H. (2000), Heuristics for Balancing Turbine Fans, *Operations Research*, 48, 591-602.

Choi, W. (2000), A Heuristic Algorithm for a Turbine-Blade-Balancing Problem, *Proceedings of 2000 Spring Joint Conference of KIIE and KORMS*, Kyungnam University, Korea.

Choi, W., Kang, H. and Baek, T. (1999), Turbine Blade Balancing Problems, *International Journal of Production Economics*, 60-61, 405-410.

Darlow, M. S. (1989), *Balancing of High-Speed Machinery*, Springer-Verlag, New York.

Fathi, Y. and Ginjupalli, K. K. (1993), A Mathematical Model and a Heuristic for the Turbine Balancing Problem, *European Journal of Operational Research*, 63, 336-342.

Karmakar, N. R. M. and Karp, R. M. (1982), The Differencing Method for Set Partitioning, Report No. UCB/CSD 82 /113, Computer Science Division, University of California, Berkley.

LaPorte, G. and Mercure, H. (1988), Balancing Hydraulic Turbine Runners: A Quadratic Assignment Problem., *European Journal of Operational Research*, 35, 378-381

Mason, A. and R nnqvist, M. (1997), Solution Methods for the Balancing of Jet Turbines, *Computers and Operations Research*, 24(2), 153-167.

Mosevich, J. (1986), Balancing Hydraulic Turbine Runners: A Discrete Combinatorial Optimization, *European Journal of Operational Research*, 26, 202-204.

Papadimitriou, C. H. and Steiglitz, K. (1982), *Combinatorial Optimization*, Prentice-Hall, Inc.

Storer, R. H. (1999), Extensions of and Uses for the Differencing Algorithm for Number Partitioning, Report No. 99T-09, Department of Industrial and Manufacturing Systems Engineering, Lehigh University, Bethlehem, Pennsylvania.

Storer, R. H., Flanders, S. W. and Wu, S. D. (1996), Problem Space Search for Number Partitioning, *Annals of Operations Research*, 10, 465-487.