JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 24 No. 10, pp. 57-64, October 2019
https://doi.org/10.9708/jksci.2019.24.10.057

Notes On Inverse Interval Graph Coloring Problems

Yerim Chung*, Hak-Jin Kim*

*Assistant Professor, Yonsei School of Business, Yonsei University, Seoul, Korea
*Professor, Yonsei School of Business, Yonsei University, Seoul, Korea

[Abstract]

In this paper, we study a polynomially solvable case of the inverse interval graph coloring problem. Given an interval graph associated with a specific interval system, the inverse interval graph coloring problem is defined with the assumption that there is no proper K-coloring for the given interval graph, where K is a fixed integer. The problem is to modify the system of intervals associated with the given interval graph by shifting some of the intervals in such a way that the resulting interval graph becomes K-colorable and the total modification is minimum with respect to a certain norm. In this paper, we focus on the case $K=1$ where all intervals associated with the interval graph have length 1 or 2, and interval displacement is only allowed to the righthand side with respect to its original position. To solve this problem in polynomial time, we propose a two-phase algorithm which consists of the sorting and First Fit procedure.

▸ **Key words**: Inverse Optimization, Graph Coloring, Interval Graph, Algorithm, Scheduling

[요 약]

이 논문에서는 인터벌 그래프 컬러링 역문제 중 다항시간 안에 풀이 가능한 경우에 대해 연구한다. 인터벌 그래프의 컬러링 역문제는 주어진 인터벌 그래프를 K개의 서로 다른 색깔로 색칠할 수 없는 경우를 가정하며, 다음과 같이 정의된다. 주어진 인터벌 그래프가 K개의 색깔을 이용해서 모두 칠해질 수 있도록 인터벌 그래프와 연관되어 있는 인터벌 시스템을 최소한으로 수정하는 문제이다. 인터벌 시스템에서 두 인터벌이 부분적으로라도 서로 겹쳐있는 구간이 있을 경우 두 인터벌에 해당하는 노드들이 엣지로 연결되어 있음을 의미하고, 따라서 이 경우에는 해당 노드들을 같은 색깔을 이용해 칠할 수 없다. 따라서 겹쳐져 있는 인터벌들을 이동시켜 해당 그래프의 chromatic number를 바꿀 수 있다. 본 논문에서는 인터벌의 길이가 모두 1 또는 2이며, 인터벌의 이동이 본래 위치 대비 오른쪽으로만 가능하다는 제한이 있는 경우에 대해 집중 탐구한다. 이 문제를 해결하는 다항시간 알고리즘으로 sorting과 선입선출 방식을 사용하는 2단계 알고리즘을 제안한다.

▸ **주제어**: 역최적화, 그래프 컬러링, 인터벌 그래프, 알고리즘, 스케줄링

• First Author: Yerim Chung, Corresponding Author: Hak-Jin Kim
*Yerim Chung (yerimchung@yonsei.ac.kr), School of Business, Yonsei University
*Hak-Jin Kim (hakjin@yonsei.ac.kr), School of Business, Yonsei University
• Received: 2019. 08. 27, Revised: 2019. 09. 25, Accepted: 2019. 09. 30.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

In the minimum graph coloring problem[1], we are given a graph $G=(V,E)$, and we need to decide whether the given graph is K-colorable for a fixed positive integer K. A graph $G=(V,E)$ is said to be K-colorable if any pair of adjacent nodes (connected by an edge) can be colored with different colors[1]. The graph coloring has many applications in the context of university course timetabling and examination timetabling[2-4]. Given a list of courses (or examinations), any pair of courses should not be scheduled on a same time-slot if there is at least one student who wants to attend both courses. It is well-known that the minimum graph coloring problem and the university course/examination timetabling problem are NP-complete, and it is also hard to find an optimal solution for their optimization problems[1].

In the inverse optimization problem[5], we are given a feasible solution for an optimization problem, and we need to modify as little as possible the instance of the given problem so that the prescribed feasible solution becomes optimal in the modified instance. For many combinatorial optimization problems, its inverse problem has been studied by many researchers[5-10]. In [9], the authors studied the complexity status of the inverse graph coloring problem in interval graphs.

Given an interval graph $G=(V,E)$ associated with n intervals and a positive integer K, the inverse interval graph coloring problem [9] is defined as the problem of modifying the system of the intervals by making parallel translation of intervals in such a way that the chromatic number of the resulting interval graph does not exceed K and the total deviation between the original and new interval system is minimum under the L_1-norm.

In [9], the authors consider the inverse interval graph coloring problem with $K=1$, called the inverse booking problem, in the context of hotel reservation. The inverse booking problem and its variant are shown to be NP-hard in the strong sense even in the case $K=1$ and that no polynomial time approximation algorithm can guarantee an approximation ratio of $O(n^{1-\epsilon}), \epsilon>0$, where n denotes the number of nodes of the given interval graph[9].

In this paper, we focus on the case where the lengths of the intervals associated with the given interval graph are all restricted to 1 or 2, and interval displacement is only allowed to the righthand side with respect to its original position. We call this problem the inverse booking problem with release dates and denote it by $IBPR_{K,1,2}$.

The formal definition of the problem $IBPR_{K,1,2}$ is as follows: given a list of n intervals $I_j := [a_j, a_j + l_j)$ of length $l_j \in 1,2, j \in 1,\cdots,n$, and a positive integer $K \geq 1$, the task is to find the new left endpoint a'_j of each interval such that $a_j \leq a'_j$ and the interval graph associated with new intervals $I'_j = [a'_j, a'_j + l_j)$, $j \in 1,\cdots,n$ can be properly colored with K different colors or less. The objective is to minimize the deviation $\sum_{j=1}^{n}(a'_j - a_j)$. All numerical data are assumed to be non-negative integers.

Using the terminology of job scheduling in the literature [11-14], the problem with $K=1$ can be referred to as the minimum tardiness single machine scheduling problem with arbitrary release dates. We denote this problem by $IBPR_{1,2}$.

It is known that the generalized problem, $IBPR$, without restriction on interval lengths is NP-hard[9], while $IBPR$ with equal length intervals is polynomially solvable[14]. In this paper, we propose a polynomial time algorithm for solving $IBPR_{1,2}$.

This paper is organized as follows: In Section 2, we discuss some preliminary results on inverse interval graph coloring. In Section 3 we describe a polynomial time algorithm for solving $IBPR_{1,2}$. We prove in Section 4 the optimality of our algorithm. We then conclude the paper.

II. Preliminaries

Here we provide some preliminary remarks on interval translations. Since the objective function is measured by the sum of interval translations, the following lemmas will be useful later on.

Given an interval graph, we denote the related interval system by a position vector $I=(a_1,\cdots,a_n)$, each component a_j of which is the left endpoint of an interval $[a_j,a_j+l_j), j=1,\cdots,n$. Without loss of generality, we assume that $a_1 \leq \cdots \leq a_n$. By definition of $IBPR$, the task is to determine a new position vector for intervals in which all intervals can be placed without overlapping with others. A position vector is said to be optimal if the deviation between the original and new interval position vector is minimum.

Lemma 1. Assume that all intervals have equal length. Then, there is at least one optimal position vector for $IBPR$ preserving the original interval order.

Proof. Let $I=(a_1,\cdots,a_n)$be the initial position vector of the intervals $[a_j,a_j+l_j), j=1,\cdots,n$ such that $a_1 \leq \cdots \leq a_n$. We assume that all intervals have equal length, i.e., $l_j=l$ for any $j\in 1,\cdots,n$. We assume an optimal position vector $I^*=(a^*_1,\cdots,a^*_n)$. Let us define a set $P_{I^*}$of pairs (h,k) such that $a_h \leq a_k$ and $a^*_k \leq a^*_h$, meaning that interval $[a_h,a_h+l)$, originally placed before interval $[a_k,a_k+l)$, is placed in I^*after $[a_k,a_k+l)$. Suppose that $P_{I^*}=\phi$. Let $[a_i,a_i+l)$ and $[a_j,a_j+l)$ be two intervals such that $(i,j)\in P_{I^*}$ with $a_i=\min\{a_h \mid \exists k, (h,k)\in P_{I^*}\}$, and $a^*_j=\min\{a^*_k \mid (i,k)\in P_{I^*}\}$ for a fixed i. As the interval translations are only allowed to the right-hand side, by definition of P_{I^*}, we have $a_i \leq a_j \leq a^*_j \leq a^*_i$.

Let us now consider another position vector I' obtained from I by exchanging two intervals $[a_i,a_i+l)$ and $[a_j,a_j+l)$. All the other intervals' positions remaining the same, this interval exchange does not make any additional translation cost. In fact, the translation cost $c(I^*_{(i,j)})$ of I^* induced by the intervals $[a_i,a_i+l)$ and $[a_j,a_j+l)$ is equal to the related cost $c(I'_{(i,j)})$ of I':

$$\begin{aligned} c(I^*_{(i,j)}) &= (a^*_i - a_i)+(a^*_j - a_j) \\ &= (a^*_j - a_i)+(a^*_i - a_j) \\ &= (a'_i - a_i)+(a'_j - a_j) \\ &= c(I'_{(i,j)}) \end{aligned}$$

It means that I' is at least as good as I. So, by exchanging every pair of intervals in P_{I^*}, we can reposition all of the intervals according to the original interval order. Since $|P_{I^*}|=O(n^2)$, it requires $O(n^2)$-time to find an optimal position vector preserving the initial order of intervals.

■

Proposition 1. $IBPR$ with equal length intervals can be solved in $O(n)$ time.

Proof. Denoting by $I^*=(a^*_1,\cdots,a^*_n)$an optimal solution preserving the order of interval's left endpoint (see **Lemma 1**), it is straightforward to show that we have $a^*_1=a_1$ and $a^*_j=\max\{a_j, a^*_{j-1}+l\}$. Thus, I^* can be computed using this relation from $j=1$to $j=n$.

■

In the sequel, we assume that every interval has a length of 1 or 2 and that interval translations are only allowed to the righthand side. One can observe the following:

Observation 1. Assume two intervals intersecting each other. If they have different starting points, then it is always less expensive to move the interval with larger starting point value in order to legally place both intervals on a single line.

Observation 2. If two intervals of length 1 and 2 start at the same point, then it is always less expensive to move the longer one in order to legally place both intervals on a single line.

Consider the case where an interval of length 2 intersects an interval block, composed of several intervals of length 1 or 2.

Lemma 2. Let L be an interval block, composed of some intervals of length 1 or 2. Let $[a_j, a_j+2)$ be an interval of length 2 intersecting L. We suppose that there is an empty space of width at least 2 between the block L and an interval following L (if exists). Then, in order to legally position all intervals, moving some intervals of L is at least as expensive as moving the interval $[a_j, a_j+2)$ (without moving L) to the end of L.

Proof. Let us compare the following two solutions. In Solution 1, we shift interval $[a_j, a_j+2)$ (without moving any interval of L) to the end of L. In Solution 2, we split L into two parts (L_1 finishing at a_j and L_2 starting at a_j) and shift the second part L_2 by distance of 2 to make a space for the interval $[a_j, a_j+2)$. Assume that L_2 contains p intervals; the length $s(L_2)$ of L_2 is such that $p \le s(L_2) \le 2p$. Solution 2 yields the translation cost of $2p$ while Solution 1 yields the cost of $s(L_2)$. Two solutions have the same cost if L_2 is composed of p intervals of length 2.

■

Lemma 3. Let L be an interval block, composed of intervals of length 1 or 2. Let $[a_j, a_j+2)$be an interval of length 2 intersecting L. If L is followed by another

interval block L' with distance at least 1, then it is always less expensive to shift L'(if necessary) and insert $[a_j, a_j+2)$ between L and L' than to shift $[a_j, a_j+2)$ to the end of the second block L'.

Proof. Let p and p' be the number of intervals contained in L and L', and $s(L)$ and $s(L')$ the length of L and L', respectively. Clearly, $p \le s(L)$ and $p' \le s(L')$. If there is an empty space of width 2 (or more) between L and L', then due to Lemma 1 it is optimal to place $[a_j, a_j+2)$ at the end of L. Suppose now that the width of an empty space between L and L' is equal to 1. In this case, in order to put $[a_j, a_j+2)$ between L and L', we have to push L' to the right by distance 1. However, since there is always an empty space of width at least 1 between any two interval blocks, the translation of L' does not cause any overlap with its following block, say L''. So, the related translation cost is equal to $s(L)+p'$. On the other hand, it costs $s(L)+s(L')+1$ to shift $[a_j, a_j+2)$ to the end of L'. Since $p' < s(L')+1$, inserting $[a_j, a_j+2)$ between $L$and L' is preferred to placing $[a_j, a_j+2)$ at the end of L'.

■

III. A polynomial-time algorithm solving $IBPR_{1,2}$

An instance of $IBPR_{1,2}$ is given by n_1 intervals of length 1 and n_2 intervals of length 2. Let $I(1) = \{[a_i, a_i+1), 1 \le i \le n_1\}$ and $I(2) = \{[a_j, a_j+2), n_1+1 \le j \le n_1+n_2\}$ be the set of intervals of length 1 and 2, respectively. We assume that the endpoints of every interval have integer values, and the intervals of $I(1)$ and $I(2)$ are respectively sorted in non decreasing order of their left endpoint values;

$a_1 \le \cdots \le a_i \le \cdots a_{n_1}$a n d $a_{n_1+1} \le \cdots \le a_j \le \cdots a_n$.

Algorithm 1. A Greedy Algorithm For Solving $IBPR_{1,2}$

Input: $I(1) = \{[a_i, a_i+1), 1 \le i \le n_1\}$ and $I(2) = \{[a_j, a_j+2), n_1+1 \le j \le n_1+n_2\}$, each sorted in nondecreasing order of their left endpoint values, and a position vector $I = (a_1, \cdots, a_{n_1}, a_{n_1}+1, \cdots, a_n)$.

Output: an optimal position vector I^*.

Step 1.

1. Find an optimal position for $I(1)$;
we denote by $I_{I(1)}=(a_1,\cdots,a_{n_1})$ the sub-instance of I restricted to the set $I(1)$ of intervals of length 1, and by $J=(a'_1,\cdots,a'_{n_1})$ an optimal solution for $I_{I(1)}$ such that $a'_1\leq\cdots\leq a'_{n_1}$.

Step 2. First Fit

2. Define a position vector $I^*=(a^*_1,\cdots,a^*_{n_1},a^*_{n_1}+1,\cdots,a^*_n)$ such that $a^*_j=a'_j$ for $j=1,\cdots,n_1$ and $a^*_j=a_j$ for $j=n_1+1,\cdots,n$. The first n_1 components of I^* indicate the optimal positions of the intervals of $I(1)$ and the other n_2 components indicate the initial positions of intervals of $I(2)$.
1.
2. 3. **For** each interval $[a_j,a_j+2)\in I(2)$ **do**
3.
4. 4. Put $[a_j,a_j+2)$ in the nearest blank on its right side (if necessary, push the next interval block by distance 1);
5.
6. 5. Renew the position vector I^*.
7.
8. 6. **End For**
9.
10. 7. Return I^*.

Let us denote by $I=(a_1,\cdots,a_{n_1},a_{n_1}+1,\cdots,a_n)$ the initial position vector of the given intervals; the first n_1 components indicate the initial position of intervals of length 1 and the rest concern the initial position of intervals of length 2. In the sequel, an instance of $IBPR_{1,2}$ will be presented by means of this position vector I. Then, $IBPR_{1,2}$ can be seen as the problem of finding a new interval position vector $I^*=(a^*_1,\cdots,a^*_{n_1},a^*_{n_1}+1,\cdots,a^*_n)$ for which (i) all intervals can be legally positioned on a same line, i.e., without intersecting any other intervals, (ii) $\forall i\in 1,\cdots,n, a_i\leq a^*_j$ and (iii) the total cost of translations $\sum_{j=1}^{n}(a^*_j-a_j)$ is minimum.

We propose a greedy algorithm for solving $IBPR_{1,2}$. Our algorithm is implemented in two steps. At the first step, we restrict our attention to the set $I(1)$ of intervals of length 1. Due to **Proposition 1** and **Lemma 1,** one can find in $O(n)$ time a minimum cost legal position assignment for $I(1)$ preserving the initial interval order.

Note that once all the intervals of length 1 are optimally distributed on the line, we obtain a finite number of interval blocks; we number them in nondecreasing order of their starting point values. These blocks generate a finite number of gaps (empty spaces) between blocks of consecutive indices. We call them blanks. Since the endpoints of every interval are integers, each blank has an integral size larger than 1.

At the second step, we are given n_2 intervals of length 2 to be positioned on the line already loaded by the intervals of length 1. Using the procedure **First Fit,** we determine the new position for each interval of $I(2)$. Recall that the intervals of $I(2)$ are sorted in nondecreasing order of their left endpoint values. The procedure **First Fit** will be executed following this order until every interval of $I(2)$ is legally positioned, i.e., without intersecting any other intervals. The complexity of the proposed algorithm is of order $O(n)$.

IV. The Proof for Optimality

Given an instance $I=(a_1,\cdots,a_{n_1},a_{n_1}+1,\cdots,a_n)$ of $IBPR_{1,2}$, we denote by $I_{I(1)}=(a_1,\cdots,a_{n_1})$ the sub-instance of I restricted to the interval set $I(1)$ of length 1. Let $J=(a'_1,\cdots,a'_{n_1})$ be an optimal solution for $I_{I(1)}$ such that $a'_1\leq\cdots\leq a'_{n_1}$(the existence of J is due to **Lemma 1**). We call J the intermediate solution of $IBPR_{1,2}$. Then, one can

consider another problem to find an optimal assignment of the intervals of $I(2)$ to the line already loaded by the intervals of $I(1)$. An instance of this problem can be expressed by the following position vector $\tilde{I}=(a'_1,\cdots,a'_{n_1},a_{n_1}+1,\cdots,a_n)$ (of dimension $n=n_1+n_2$); the first n_1 components

indicate the optimal positions for $I(1)$ found at the first step, and the other n_2 components indicate the initial positions for $I(2)$. The objective is to find a new position vector for which all the intervals of $I(1)\cup I(2)$ can be legally positioned on the same line and the total translation is minimum.

We first show that **First Fit** solves this problem efficiently.

Lemma 4. The procedure **First Fit** is optimal for the instance $\tilde{I}$.

Proof. This result immediately follows from **Lemma 1**, as the procedure **First Fit** respects this lemma.

■

Let $I^*=(a^*_1,\cdots,a^*_{n_1},a^*_{n_1}+1,\cdots,a^*_n)$ be an optimal solution for the whole instance I such that $\forall j=1,\cdots,n,\ \ a_j\le a^*_j$ and $a^*_1\le\cdots\le a^*_{n_1}$and $a^*_{n_1+1}\le\cdots\le a^*_n$ (such an optimal position vector exists due to **Lemma 1**). The sub-vector $I^*_{I(1)}$ of I^* restricted to $I(1)$ indicates the position of the intervals of $I(1)$ in the final solution I^*. Below, we show that there exists an optimal solution $I^*$of $IBPR_{1,2}$ in which every interval of length 1 is located on the righthand side with respect to its position in the intermediate solution, J.

For any two vectors u and v of same dimension, we say $u\le v$ if the component values of u are, component by component, smaller than those of v.

Lemma 5. There exists an optimal position vector I^* such that $J\le I^*_{I(1)})$, i.e., $\forall i=1,\cdots,n,\ a'_j\le a^*_j$.

Proof. Suppose that there are some intervals $[a_j,a_j+1),j=1,\cdots,n$, such that $a^*_j<a'_j$ where a'_j and a^*_j respectively denote the left endpoint of the interval $[a_j,a_j+1)$ in the intermediate solution, J, and the optimal solution, I. Let $[a_l,a_l+1)$ be the one having the smallest index among such intervals: $\forall i\le l, a'_j\le a^*_j$ and $a^*_l<a'_l$. In J, the interval $[a_l,a_l+1)$ is placed at the position a'_l. It means that all positions before a_l are already taken by some intervals of smaller index than l. Otherwise, one can reduce the translation cost associated with J by placing interval $[a_l,a_l+1)$ before the position a'_l, and this contradicts the fact that J is optimal for $I_{I(1)}$. Since $a^*_l<a'_l$, there is in J an interval h with $h<l$, placed at the position a^*_l, i.e., $a'_h=\alpha=a^*_l$. Besides, since $h<l$, we have: $a'_h=\alpha\le a^*_h$. Let us now exchange in $I^*_{I(1)}$ the components $a^*_h=\beta$ and $a^*_l=\alpha$, i.e., we assign the interval $[a_h,a_h+1)$ to the position α and $[a_l,a_l+1)$ to the position β. This exchange does not yield any extra cost. So, by repeating such exchanges, we can construct an optimal solution I^* such that $\forall j=1,\cdots,n_1$, $a'_j\le a^*_j$.

■

Lemma 5 implies that every optimal solution for I (the whole instance of $IBPR_{1,2}$ can be constructed from an optimal solution for the sub-instance $I_{I(1)}$. Let us now show that Algorithm 1 optimally solves $IBPR_{1,2}$.

Proposition 2. Algorithm 1 optimally solves $IBPR_{1,2}$ in time $O(n)$.

Proof. Given an instance I, let us denote by $\beta(I)$ the optimal solution value for I. Then, the solution value returned by Algorithm 1 is equal to $\beta(I_{I(1)})+\beta(\tilde{I})$. To prove the optimality of Algorithm 1, we show that $\beta(I)=\beta(I_{I(1)})+\beta(\tilde{I})$. It

is straightforward to see that $\beta(I) \leq \beta(I_{I(1)}) + \beta(\tilde{I})$. In fact, $IBPR_{1,2}$ is a minimization problem and Algorithm 1 returns a feasible solution for it.

We now show that $\beta(I) \geq \beta(I_{I(1)}) + \beta(\tilde{I})$. Due to Lemma 5, one can construct an optimal solution I^* for I from an optimal solution $J = (a'_1, \cdots, a'_{n_1})$ for $I_{I(1)}$. So, the optimal solution value of $IBPR_{1,2}$ can be expressed as follows: $\beta(I) = \beta(I_{I(1)}) + \gamma$ where γ is the translation cost occurred when positioning the intervals of length 2. Since the obtained solution is feasible for $\tilde{I}(\gamma \geq \beta(\tilde{I}))$, we have: $\beta(I) \geq \beta(I_{I(1)}) + \beta(\tilde{I})$.

■

V. Conclusion

In this paper, we have studied the inverse booking problem with release dates, which is an interesting variant of the inverse interval graph coloring problem where each interval has length 1 or 2, and the translation of intervals is only allowed to the righthand side. We proposed a polynomial time algorithm for solving this problem and proved its optimality. This algorithm can be very useful for solving the university course timetabling problem, because many university offers the lectures of 1 and 2 hours for the courses of 3 credit. However, our algorithm does not work for the case where interval lengths are either 1 or $M \geq 3$. It will be interesting to investigate the computational complexity of $IBPR$ with another restriction on interval lengths. It also remains as a future research to devise an efficient algorithm that can solve $IBPR_{K,1,2}$ for a fixed constant $K \geq 2$.

REFERENCES

[1] M.R. Garey, and D.S. Johnson, "Computers and Intractability - A Guide to The Theory of NP-completeness," San Francisco, Freeman, January, 1979.

[2] M. Cangalovic, and J.A.M. Schreuder, "Exact Coloring Algorithms for Weighted Graphs Applied to Timetabling Problems with Lectures of Different Lengths," European Journal of Operational Research, Vol. 51, No. 2, pp. 248-258, March 1991.

[3] D. De Werra, "Some Combinatorial Models for Course Scheduling," Practice and Theory of Automated Timetabling, ser. Springer, Lecture Notes in Computer Science, Vol. 1153, pp. 296-308, March 1996.

[4] D. De Werra, "The Combinatorics of Timetabling," European Journal of Operational Research, Vol. 96, No. 3, pp. 504-513, February 1997.

[5] R.K. Ahuja, and J.B. Orlin, "Inverse Optimization," Operations Research, Vol. 49, No. 5 pp. 771-783, October 2001.

[6] R.K. Ahuja, and J.B. Orlin, "A Faster Algorithm for the Inverse Spanning Tree Problem," Journal of Algorithms, Vol. 34, No. 1, pp. 177-193, January 2000.

[7] Y. Chung, and M. Demange, "On Inverse Traveling Salesman Problems," 4OR - A Quarterly Journal of Operations Research, Vol. 10, pp. 193-209, June 2012.

[8] Y. Chung, and M. Park, "Notes On Inverse Bin-Packing Problems," Information Processing Letters, Vol. 115, pp. 60-68, January 2015.

[9] Y. Chung, J.F. Culus, and M. Demange, "Inverse Chromatic Number Problems in Interval and Permutation Graphs," European Journal of Operational Research Vol. 243, pp. 763-773, 2015.

[10] C. Heuberger, "Inverse Combinatorial Optimization: A Survey on Problems, Methods, and Results," Journal of Combinatorial Optimization, Vol. 8, No. 3, pp. 329-361, September 2004.

[11] M.R. Garey, R.E. Tarjan, and G.T. Wilfong, "One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties," Mathematics of Operations Research, Vol. 13, pp. 330–348, May 1988.

[12] A.H.G. Rinnooy Kan, "Machine Scheduling Problem: Classification, Complexity and Computation," Nijhoff, The Hague, December 1976.

[13] M. Müller-Hannemann, and A. Sonnikow, "Non-Approximability of Just-In-Time Scheduling," Journal of Scheduling, Vol. 12, No. 5, pp. 555-562, October 2009.

[14] P. Baptiste, "Scheduling Equal-Length Jobs on Identical Parallel Machines," Discrete Applied Mathematics, Vol. 103, No. 1-3, pp. 21–32, July 2000.

Authors

Yerim Chung received the B.S. degree in Business Administration from Yonsei University, Korea, in 2000. She received the M.S. and Ph.D. degree in Applied Mathematics and Computer Science from Paris 1 University, France, in 2004 and 2010, respectively. Dr. Chung joined the faculty of Business School at Yonsei University, Seoul, Korea, in 2011. She is interested in inverse optimization and network optimization, and their many application problems.

Hak-Jin Kim received the M.S. degree in Mathematics from University of Illinois, Urbana-Champaign, and the Ph.D. degree in Operations Research from Tepper Business School, Carnegie-Mellon University, U.S.A. He has been a faculty member in the School of Business, Yonsei University, since 2001. He is interested in the logic-based optimization, the constraint programming, reinforcement learning, and their many application problems.