An Application of Integer Programming Solvers to an Item Collecting Problem in Directed Bipartite Graphs

Yoshiyuki Karuno^a and Seiya Yoshikawa^b

^aFaculty of Mechanical Engineering, Kyoto Institute of Technology, Kyoto 606-8585, Japan

^bGraduate School of Science and Technology, Kyoto Institute of Technology, Kyoto 606-8585, Japan

* Corresponding Author: karuno@kit.ac.jp

Abstract

In this paper, a weighted item collecting problem in directed bipartite graphs is considered, where a finite set of items with non-negative weights and a finite set of players are contained. No budget constraints are imposed on the players, and the problem is viewed as a weighted version of the minimal switching graph problem, which was introduced as a constrained via minimization in integrated circuit design. In this paper, the weighted item collecting problem is interpreted as a special case of the weighted maximum clique problem, and an application of integer programming solvers is investigated in order to obtain an exact solution for an instance of the weighted item collecting problem. Numerical experiments are conducted to demonstrate the exact solutions, and the results are reported.

Keywords: Engineering optimization, Item collecting, Directed bipartite graphs, Maximum clique, Integer programming.

1. Introduction

In this paper, we consider a weighted item collecting problem in directed bipartite graphs, where a finite set of items with non-negative weights and a finite set of players are contained. No budget constraints are imposed on the players (i.e., the case with a sufficiently large budget), and the weighted item collecting problem is viewed as a weighted version of the minimal switching graph problem⁽¹⁾, which was introduced by Tang (2005), for a constrained via minimization in integrated circuit design⁽²⁾. Existing algorithms for the minimal switching graph and related problems are heuristic ones^(1,3-6). In this paper, we transform the weighted item collecting problem into a conventional problem of combinatorial optimization, known as the weighted maximum clique problem, and investigate an application of integer programming (IP) solvers in order to obtain an exact solution for an instance of the weighted item collecting problem. The (weighted) maximum

clique problem has been known to be NP-hard⁽⁷⁾, while some IP solvers have considerably been improved to find an exact solution for a larger size instance of many integer programming problems.

We review the mathematical description of the weighted item collecting problem (without budget constraints on the players), following our last paper with respect to the problem⁽⁶⁾. Let $I = \{i \mid i = 1, 2, \dots, m\}$ denote a set of m items, and let $J = \{j \mid j = 1, 2, \dots, n\}$ denote a set of n players. Two non-negative weights w_i and w_{m+i} such that $w_i + w_{m+i} > 0$ are associated with each item $i \in I$ as the rewards. There are two kinds of signals, by which the items and the players are connected. For convenience, we assign a color, either blue or red, to each connection signal. For each player $j \in J$, let $B_j \subseteq I$ (resp., $R_j \subseteq I$) denote the set of items, each of which is initially connected with the player $j \in J$ by a blue signal (resp., by a red signal). It must be satisfied that $B_j \cap R_j = \emptyset$ for any player $j \in J$. On the other hand, without loss of generality, it is assumed that each player $j \in J$ meets $B_j \cup R_j \neq \emptyset$. It is also assumed that $B_1 \cup R_1 \cup \dots \cup B_n \cup R_n = I$.

We say that a player $j \in J$ is *reversed* when changing every color of the initial connection signals associated with the player $j \in J$ to the opposite one (i.e., either from blue to red, or from red to blue). As a feasible solution of the weighted item collecting problem, a reversing vector $x = (x_1, x_2, \dots, x_n)$ is defined, where for each $j = 1, 2, \dots, n$,

$$x_j = \begin{cases} 1 & \text{if player } j \text{ is reversed,} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

That is, for a reversed player $j \in J$ in a reversing vector x (i.e., for a player $j \in J$ with $x_j = 1$), an item $i \in B_j$ (resp., an item $i \in R_j$) is connected with the player by a red signal (resp., by a blue signal). A signal color (i.e., either blue or red) between an item and an associated player with the item can be represented by a direction of an arc when regarding the set of m items, the set of n players, and the connection

between them as a directed bipartite graph. We are going to illustrate an example of such a directed bipartite graph in the following section.

For a reversing vector x , let $I_B(x) \subseteq I$ (resp., $I_R(x) \subseteq I$) denote the set of items, each of which has only blue connection signals (resp., only red connection signals). We regard an item $i \in I_B(x) \cup I_R(x)$ with common color connection signals as a successful result of the *cooperation* of players associated with the item, and the n players as a team can get the weight w_i (resp., w_{m+i}) as the reward when $i \in I_B(x)$ holds (resp., when $i \in I_R(x)$ holds). The total weight as the objective function to be maximized is represented by

$$f(x) = \sum_{i \in I_B(x)} w_i + \sum_{i \in I_R(x)} w_{m+i}. \quad (2)$$

The weighted item collecting problem asks to find a reversing vector $x = x^*$ which maximizes the total weight $f(x)$ defined in Eq. (2). We may call the x^* an *optimal solution*.

2. Preliminaries

2.1. Directed Bipartite Graph Representation

Let $G(x) = (I \cup J, A_B(x) \cup A_R(x))$ denote a directed bipartite graph, called a *reversing graph*, with respect to a feasible solution x , where $A_B(x) \subseteq I \times J$ and $A_R(x) \subseteq J \times I$ are disjoint arc sets. More precisely, an arc $(i, j) \in A_B(x)$ (resp., an arc $(j, i) \in A_R(x)$) indicates a blue signal (resp., a red signal) between item $i \in I$ and player $j \in J$. In particular, let $G(0) = (I \cup J, A_B(0) \cup A_R(0))$ denote the initial reversing graph with respect to the *zero reversing vector* $x = 0$.

In Fig. 1, we provide an illustration of the initial reversing graph $G(0)$ and another reversing graph $G(x)$ with respect to $x = (1, 0, 1, 1)$, where the number of items and the number of players are $m = n = 4$. In the initial $G(0)$, each item $i \in I$ has connection signals with both colors, and there is no item with common color signals, i.e., $I_B(0) = I_R(0) = \emptyset$. Hence, the total weight of $x = 0$ is $f(0) = 0$. On the other hand, in the $G(x)$ with respect to $x = (1, 0, 1, 1)$, we see $I_B(x) = \{1, 2\}$ and $I_R(x) = \{3\}$. Hence, the total weight of the x is $f(x) = w_1 + w_2 + w_7$.

2.2. The Weighted Maximum Clique Problem

As mentioned in the first section, we are going to transform the weighted item collecting problem into a conventional problem of combinatorial optimization, known as the weighted maximum clique problem, and to apply an IP solver to an integer programming formulation of the weighted maximum clique problem in order to obtain an optimal solution for a given instance of the weighted item collecting problem. Although the (weighted) maximum clique problem is one of famous combinatorial optimization problems, we here provide a

formal description of the weighted maximum clique problem for the self-containment of this paper.

Let $H = (V, E, p)$ denote a weighted undirected graph, where V is a finite set of vertices, E is a finite set of edges with $E \subseteq V \times V$, and $p : V \rightarrow \mathbb{Z}_+ \cup \{0\}$ is a vertex-weight function, which we call a *profit* function for convenience, and we express a profit of each vertex $v \in V$ by $p[v] \geq 0$. A *clique* in graph $H = (V, E, p)$ is defined to be a subset $Q \subseteq V$ of vertices such that every pair of vertices in Q is joined by an edge in E . We represent the *size* of a clique Q by the cardinality $|Q|$, and the profit sum of the Q by

$$p(Q) = \sum_{v \in Q} p[v]. \quad (3)$$

Problem WMC (Weighted Maximum Clique)

Instance: An undirected graph $H = (V, E, p)$, where V is the vertex set, E is the edge set, and p is a profit function on V .

A feasible solution: A clique $Q \subseteq V$ in graph H .

Goal: Find a clique $Q^* \subseteq V$ such that the profit sum satisfies $p(Q^*) \geq p(Q)$ for any clique Q in graph H .

We refer to the Q^* as the maximum profit sum clique (in graph H). When each vertex $v \in V$ is given the unit profit, i.e., $p[v] = 1$, the profit sum of a clique Q means the size of it, i.e., $p(Q) = |Q|$ holds. In this paper, we should include the case of $E = \emptyset$ in graph H , and we see the following property:

Observation 1. For an instance of the weighted maximum clique problem, the maximum profit sum clique Q^* satisfies either $|Q^*| \geq 2$, or $Q^* = \{v^*\}$ such that the singleton $v^* \in V$ meets

$$p[v^*] = \max_{v \in V} \{p[v]\}. \quad (4)$$

2.3. Basic Semi-solutions

Basic semi-solutions are essentially the same as *atomic building blocks* defined in an application of the genetic algorithm by Tang (2005) for the minimal switching graph problem⁽¹⁾. If a reversing vector x contains a blue basic semi-solution (resp., a red basic semi-solution) for an item $i \in I$, then we can know that the solution x gets the weight w_i (resp., w_{m+i}). We already utilized the basic semi-solutions in greedy heuristic algorithms⁽³⁾, which found quickly an optimal solution for the test instance of the minimal switching graph problem⁽¹⁾ produced by Tang (2005). In the following section, we are going to interpret each basic semi-solution of an item as a vertex of an undirected graph $H = (V, E, p)$ of the weighted maximum clique problem, and the corresponding weight of the item as a profit of the vertex. We review the definition of the basic semi-solutions below.

Fig. 1 An example of the directed bipartite representation of the connection between the items and the players.

For a reversing vector $x = (x_1, x_2, \dots, x_n)$, the *counterpart* vector (or simply called, counterpart) of the x is defined by $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$, where for each player $j = 1, 2, \dots, n$,

$$\bar{x}_j = 1 - x_j. \quad (5)$$

The definition implies that the counterpart of the $\bar{x}$ is the original reversing vector x . Further, $I_B(x) = I_R(\bar{x})$ and $I_R(x) = I_B(\bar{x})$ hold for a reversing vector x and the counterpart $\bar{x}$. For example, consider again a reversing vector $x = (1, 0, 1, 1)$ in (b) of Fig. 1. The counterpart is $\bar{x} = (0, 1, 0, 0)$. Then, we see $I_B(x) = I_R(\bar{x}) = \{1, 2\}$ and $I_R(x) = I_B(\bar{x}) = \{3\}$. The total weight of the x is $f(x) = w_1 + w_2 + w_7$, while the total weight of the counterpart $\bar{x}$ is $f(\bar{x}) = w_3 + w_5 + w_6$.

A reversing vector $x = (x_1, x_2, \dots, x_n)$ can be regarded as a *string* of n binary bits. Introducing the ordinary symbol $*$ for representing “no care”⁽¹⁾, a semi-solution is defined by $s = [s_1, s_2, \dots, s_n]$, where for each player $j = 1, 2, \dots, n$,

$$s_j = \begin{cases} 1 & \text{if player } j \text{ is reversed,} \\ 0 & \text{if player } j \text{ is not reversed,} \\ * & \text{if player } j \text{ is not cared.} \end{cases} \quad (6)$$

The counterpart of the s is defined by $\bar{s} = [\bar{s}_1, \bar{s}_2, \dots, \bar{s}_n]$, where for each $j = 1, 2, \dots, n$,

$$\bar{s}_j = \begin{cases} 1 - s_j & \text{if } s_j \in \{0, 1\}, \\ * & \text{otherwise (i.e., if } s_j = *). \end{cases} \quad (7)$$

A *blue basic* (resp., a *red basic*) semi-solution $s^{(i)}$ (resp., $s^{(m+i)}$) for each item $i \in I$ is define to summarize all the solutions getting the weight w_i (resp., the weight w_{m+i}) of the item as a completely blue-signaled one (resp., as a completely red-signaled one). Notice that the counterpart of a blue basic semi-solution is the corresponding red basic semi-solution, i.e., for each item $i \in I$,

$$s^{(m+i)} = \bar{s}^{(i)} \quad (8)$$

holds. For the problem instance provided in Fig. 1, there are the following four (i.e., $m = 4$) blue basic semi-solutions:

$$\begin{aligned} s^{(1)} &= [*, 0, *, 1], \\ s^{(2)} &= [1, 0, *, *], \\ s^{(3)} &= [0, 1, 0, *], \\ s^{(4)} &= [*, *, 0, 1], \end{aligned}$$

and the following four red basic semi-solutions:

$$\begin{aligned} s^{(5)} &= \bar{s}^{(1)} = [*, 1, *, 0], \\ s^{(6)} &= \bar{s}^{(2)} = [0, 1, *, *], \\ s^{(7)} &= \bar{s}^{(3)} = [1, 0, 1, *], \\ s^{(8)} &= \bar{s}^{(4)} = [*, *, 1, 0]. \end{aligned}$$

Further, we say that a reversing vector $x = (x_1, x_2, \dots, x_n)$ *contains* a semi-solution $s = [s_1, s_2, \dots, s_n]$ if $x_j = s_j$ holds for any player $j \in J$ with $s_j \in \{0, 1\}$, and we express it by $x \supseteq s$. For a semi-solution $s = [s_1, s_2, \dots, s_n]$ and a reversing vector $x = (x_1, x_2, \dots, x_n)$ with $x \supseteq s$, if $s_j = *$ implies $x_j = 0$, then the x is the *minimal solution* for the s . Obviously, from a semi-solution, the minimal solution can be obtained in $O(n)$ time⁽³⁾.

Observation 2. For an instance of the weighted item collecting problem, an optimal solution x^* can get at least one weight, i.e., it satisfies $|I_B(x^*) \cup I_R(x^*)| \geq 1$, and

$$f(x^*) \geq \max_{i \in I} \left\{ \max\{w_i, w_{m+i}\} \right\}. \quad (9)$$

2.4. Consistency between Two Semi-solutions

In the transformation of the weighted item collecting problem into the weighted maximum clique problem with an undirected graph $H = (V, E, p)$, we are going to interpret a certain consistency between two basic semi-solutions as an edge of the graph H . The consistency has been defined as follows⁽³⁾.

We say that two distinct semi-solutions $s = [s_1, s_2, \dots, s_n]$ and $s' = [s'_1, s'_2, \dots, s'_n]$ are *inconsistent* if there exists some player $j \in J$ who meets all the following three conditions:

$$(i) \quad s_j \neq s'_j, \quad (ii) \quad s_j \neq *, \quad (iii) \quad s'_j \neq *. \quad (10)$$

Otherwise, they are *consistent*. The definition implies that for any item $i \in I$, the blue basic and red basic semi-solutions $s^{(i)}$ and $s^{(m+i)}$ ($= \bar{s}^{(i)}$) are inconsistent. For example, in the problem instance provided in Fig. 1, two basic semi-solutions $s^{(1)}$ and $s^{(3)}$ are inconsistent (since $s_2^{(1)} = 0 \neq s_2^{(3)} = 1$), while $s^{(1)}$ and $s^{(7)}$ ($= \bar{s}^{(3)}$) are consistent.

Also, notice that the $x = (1, 0, 1, 1)$ in (b) of Fig. 1 contains the two basic semi-solutions $s^{(1)}$ and $s^{(7)}$ ($= \bar{s}^{(3)}$). Generally, for two distinct semi-solutions s and s' , there exists a reversing vector x such that it meets $x \supseteq s$ and $x \supseteq s'$ both if and only if the two distinct semi-solutions are consistent⁽³⁾. The consistency between two distinct semi-solutions can obviously be checked in $O(n)$ time by Eq. (10).

3. Integer Programming Based Approach

3.1. Transformation Procedure

For an instance of the weighted item collecting problem, we can obtain the m blue basic semi-solutions $s^{(1)}, s^{(2)}, \dots, s^{(m)}$ and the m red basic semi-solutions $s^{(m+1)}, s^{(m+2)}, \dots, s^{(2m)}$ in $O(mn)$ time⁽¹⁾. From the $2m$ basic semi-solutions, we construct the following instance of the weighted maximum clique problem: Let $V = \{v_i \mid i = 1, 2, \dots, 2m\}$ denote the set of $2m$ vertices, and let

$$E = \{\{v_i, v_k\} \mid v_i, v_k \in V, \text{ basic semi-solutions } s^{(i)} \text{ and } s^{(k)} \text{ are consistent, } i \neq k\}. \quad (11)$$

The profit of each vertex $v_i \in V$ is defined to be

$$p[v_i] = w_i, \quad i = 1, 2, \dots, 2m. \quad (12)$$

The graph $H = (V, E, p)$ of the weighted maximum clique problem is obviously constructed in polynomial time from the instance of the weighted item collecting problem. We may call the H a *transformed graph* from an instance of the weighted item collection problem. In Fig. 2, we show the transformed graph from the instance of the weighted item collecting problem provided in Fig. 1. For example, we see the edge $\{v_1, v_7\}$, since two basic semi-solutions $s^{(1)}$ and $s^{(7)}$ ($= \bar{s}^{(3)}$) are consistent. We remark that a transformed graph H does not include edge $\{v_i, v_{m+i}\}$ for any $i \in I$, since two basic semi-solutions $s^{(i)}$ and $s^{(m+i)}$ are inconsistent by the definition of Eq. (7).

Lemma 1. For an instance of the weighted item collecting problem, and the transformed instance of the weighted maximum clique problem, let x^* denote an optimal solution and let

Fig. 2 Undirected graph of the weighted maximum clique problem transformed from the instance of the weighted item collecting problem in Fig. 1.

Q^* denote the maximum profit sum clique, respectively. Then, these objective function values meet

$$f(x^*) = p(Q^*). \quad (13)$$

Proof. For an optimal solution x^* , let

$$S(x^*) = \{s^{(i)} \mid x^* \supseteq s^{(i)}, i = 1, 2, \dots, 2m\}$$

denote the set of basic semi-solutions which are contained in the optimal solution x^* . We see that the total weight of the optimal solution is

$$f(x^*) = \sum_{i: x^* \supseteq s^{(i)}} w_i = \sum_{i: s^{(i)} \in S(x^*)} w_i.$$

From Observations 1 and 2, without loss of validity, we assume $|S(x^*)| \geq 2$. Then, any pair of two distinct basic semi-solutions $s^{(i)}$ and $s^{(k)}$ in the set $S(x^*)$ is a consistent one, and hence there exists the corresponding edge $\{v_i, v_k\}$ in the transformed graph H . Further, let

$$Q(x^*) = \{v_i \in V \mid s^{(i)} \in S(x^*), i = 1, 2, \dots, 2m\}$$

denote a subset of vertices. Notice that the $Q(x^*)$ is a clique in the transformed graph H , and the profit sum is

$$p(Q(x^*)) = \sum_{v_i \in Q(x^*)} p[v_i] = \sum_{i: s^{(i)} \in S(x^*)} w_i.$$

Hence, we have

$$f(x^*) = p(Q(x^*)) \leq p(Q^*).$$

On the other hand, for the maximum profit sum clique Q^* , we can also assume $|Q^*| \geq 2$ from Observations 1 and 2, and let

$$S'(Q^*) = \{s^{(i)} \mid v_i \in Q^*, i = 1, 2, \dots, 2m\}$$

denote the set of basic semi-solutions which correspond to the vertices in the Q^* . Then, any pair of two vertices v_i and v_k in the vertex set Q^* has the edge $\{v_i, v_k\}$ joining the two vertices in the transformed graph H , and hence any two distinct basic semi-solutions $s^{(i)}$ and $s^{(k)}$ in the set $S'(Q^*)$ are consistent, which implies that there is a reversing vector x such that it satisfies $x \supseteq s^{(i)}$ for all basic semi-solutions $s^{(i)} \in S'(Q^*)$. Then, we see that

$$\begin{aligned} f(x^*) &\geq f(x) \geq \sum_{i: s^{(i)} \in S'(Q^*)} w_i \\ &= \sum_{v_i \in Q^*} p[v_i] \\ &= p(Q^*). \end{aligned}$$

Therefore, we have $f(x^*) = p(Q^*)$. $\square$

3.2. Compounding Semi-solutions

Suppose that we have obtained the maximum profit sum clique Q^* by a certain way (in the following, we are going to obtain the Q^* by utilizing an IP solver). Again, let

$$S'(Q^*) = \{s^{(i)} \mid v_i \in Q^*, i = 1, 2, \dots, 2m\} \quad (14)$$

denote the set of basic semi-solutions which correspond to the vertices in the Q^* . Then, we can construct an optimal solution x^* from the Q^* as follows.

We first explain the compounding procedure for consistent semi-solutions⁽³⁾. For two distinct semi-solutions s and s' such that they are consistent, we obtain a semi-solution $\hat{s}$ from the two semi-solutions: For each player $j = 1, 2, \dots, n$, let

$$\hat{s}_j := \begin{cases} s_j (= s'_j) & \text{if } s_j = s'_j \in \{0, 1\}, \\ s_j & \text{if } s_j \in \{0, 1\} \text{ and } s'_j = *, \\ s'_j & \text{if } s_j = * \text{ and } s'_j \in \{0, 1\}, \\ * & \text{if } s_j = s'_j = *. \end{cases} \quad (15)$$

Recall that no player $j \in J$ meets Eq. (10). For the semi-solution $\hat{s}$, let $\hat{x}$ temporarily denote the minimal solution. Then, both of $\hat{x} \supseteq s$ and $\hat{x} \supseteq s'$ hold. The $\hat{s}$ is called a *compound semi-solution* of the consistent pair of s and s' .

In Procedure 1, we describe the procedure which derives an optimal solution x^* for an given instance of the weighted item collecting problem from the maximum profit sum clique Q^* for the transformed instance of the weighted maximum clique problem. The correctness depends on Lemma 1.

Procedure 1 (Input Q^* ; Output x^*)

Input: The maximum profit sum clique Q^* for the transformed instance of the weighted maximum clique problem.

Output: An optimal solution x^* for the given instance of the weighted item collecting problem.

- 1: Obtain set $S'(Q^*)$ of basic semi-solutions by Eq. (14);
- 2: Sort the $\kappa = |Q^*|$ basic semi-solutions of the $S'(Q^*)$ in an arbitrary order, denoted by

$$s^{[1]}, s^{[2]}, \dots, s^{[\kappa]};$$

- 3: $\hat{s} := [* , * , \dots , *]$;
 - 4: **for** ($i = 1$ to κ) **do**
 - 5: $\hat{s} :=$ (a compound semi-solution of the $\hat{s}$ and the i -th basic semi-solution $s^{[i]}$ in $S'(Q^*)$); /* which is performed by Eq. (15). */
 - 6: **end for**
 - 7: Construct the minimal solution x^* from the final $\hat{s}$ of compound semi-solution;
 - 8: **return** x^* as an optimal solution.
-

The set $S'(Q^*)$ of basic semi-solutions in Eq. (14) is constructed in $O(m)$ time from the maximum profit sum clique Q^* , and an arbitrary ordering of the $\kappa = |Q^*|$ basic semi-solutions of the set $S'(Q^*)$ is also obtained in $O(m)$ time. The compounding of two consistent semi-solutions by Eq. (15) requires $O(n)$ time, and hence it takes $O(mn)$ time to perform the main for-loop. As mentioned before, the minimal solution from a semi-solution is constructed in $O(n)$ time. Therefore, the time complexity of Procedure 1 is evaluated to be $O(mn)$ time.

3.3. Integer Programming Formulation

We finally show an integer programming formulation of the transformed instance of the weighted maximum clique problem. For a subset $Q \subseteq V = \{v_1, v_2, \dots, v_{2m}\}$ of vertices in the transformed graph $H = (V, E, p)$, let $y[v]$ denote a 0-1 variable such that for each vertex $v \in V$,

$$y[v] = \begin{cases} 1 & \text{if } v \in Q, \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

Using the 0-1 variables $y[v]$, the weighted maximum clique problem is represented by the following integer programming (which can be found also in the literature):

Problem IP (Integer Programming Form of WMC)

$$\text{maximize} \quad \sum_{v \in V} p[v]y[v] \quad (17)$$

$$\begin{aligned} \text{subject to} \quad & y[u] + y[v] \leq 1 \quad \text{for } \forall u, v (\neq u) \in V \\ & \text{such that } \{u, v\} \notin E, \\ & y[v] \in \{0, 1\} \quad \text{for } \forall v \in V. \end{aligned} \quad (18)$$

Due to the space limitation, we would like to omit the discussion for the equivalency of the integer programming formulation to the weighted maximum clique problem. In the

numerical section, we are going to apply an IP solver to the above integer programming formulation of the weighted maximum clique problem, and find an exact solution y^* . Then, we obtain the maximum profit sum clique by

$$Q^* := \{v \in V \mid y^*[v] = 1\},$$

and further we construct an optimal solution x^* of the given instance of the weighted item collecting problem from the Q^* by Procedure 1.

4. Numerical Experiments

We write programs for generating test instances and for a greedy heuristic⁽⁵⁾ in C language, and utilize CPLEX⁽⁸⁾ as an IP solver to find an exact solution for an instance of the weighted maximum clique problem described by the integer programming formulation. The programs in C and the IP solver are run on a desktop computer with Windows 10 Pro Education (64bit), Intel Core i7 4790 CPU (3.60GHz) and 12GB memory. The instances of the weighted item collecting problem to be tested are randomly generated as follows:

- The number of items and the number of players: $m = n = 200$, and $m = n = 300$.
- Weights of each item $i \in I$: Uniformly random integers $w_i, w_{m+i} \in \{1, 2, \dots, 10\}$, and the unit weights $w_i = w_{m+i} = 1$.
- The number of signaling players of each item $i \in I$:

$$\delta_i = |\{j \in J \mid i \in B_j \cup R_j\}| \in \{2, 3, 4\}$$

with probabilities 0.35, 0.35 and 0.30, respectively.

- The total number of arcs between the sets I and J : $|A_B(0) \cup A_R(0)| = \sum_{i \in I} \delta_i$. The direction of each arc is chosen between the two alternatives with the same probability. It is checked whether each player is associated with at least one item.

Table 1 illustrates the performance of the IP solver on instances with $m = n = 200$ and $w_i, w_{m+i} \in \{1, 2, \dots, 10\}$. For the comparison, we show the total weights obtained by the zero reversing vector $x = 0$ (i.e., the total profit in the initial reversing graph $G(0)$), and that by a greedy heuristic algorithm presented in our previous paper⁽⁵⁾. It is obvious that the total weight is significantly improved from the initial reversing graph by the optimal solution of the IP solver. Also, we show the clique size $|Q^*|$ and the execution time of the IP solver on the desktop PC. In an actual situation, the $|Q^*|$ means the number of via holes which can be reduced from the initial design in double-sided printed circuit boards. The IP solver finds an optimal solution at most fourteen seconds for the twenty test instances. The Gray code based enumeration⁽⁹⁾ as an

exact algorithm treated only test instances with $m = n = 20$ in our previous study⁽³⁾, and the application of the IP solver based on the transformation into a conventional problem of combinatorial optimization may indicate a novel direction to find an optimal solution for a larger size of instances of the weighted item collecting problem.

Table 1 Performance of the IP solver on instances with $m = n = 200$ and $w_i, w_{m+i} \in \{1, 2, \dots, 10\}$.

No.	Total Weight, $f(x)$			cplex info.	
	$x = 0$	greedy	cplex	$ Q^* $	CPU [sec]
1	317	799	924	144	4.00
2	284	769	944	146	2.11
3	353	793	915	140	6.78
4	323	798	984	146	1.29
5	309	881	987	152	3.10
6	259	836	927	143	3.98
7	378	859	978	152	0.75
8	351	843	947	151	4.48
9	344	855	968	149	1.39
10	318	815	956	150	1.21
11	256	781	874	142	11.48
12	312	844	967	141	6.15
13	376	869	991	148	0.81
14	341	846	939	143	13.14
15	366	806	959	147	2.10
16	313	801	958	139	2.65
17	322	818	927	142	2.21
18	348	770	940	143	1.84
19	359	903	963	150	0.68
20	290	882	914	143	0.59
Ave.	326.0	828.4	948.1	145.6	3.54

Table 2 demonstrates the performance of the IP solver on instances with $m = n = 200$ and $w_i = w_{m+i} = 1$. Also in this table, the total weight is significantly improved from the initial reversing graph by the optimal solution of the IP solver. For the case with the unit weights, we actually observe that the total weight of the optimal solution is equal to the size of the maximum profit sum clique, i.e., $f(x^*) = p(Q^*) = |Q^*|$. On the other hand, we notice that the IP solver requires a larger amount of execution time than that for the case with non-unit weights (i.e., $w_i, w_{m+i} \in \{1, 2, \dots, m\}$) in Table 1. In particular, the IP solver spends over 600 seconds (i.e., over ten minutes) as the execution time for two instances, No. 3 and No. 6. The bounding procedure may be less effective for the unit weight case. We here remark again that we do not any budget constraint on the players. The total weight of an optimal solution with respect to no budget case may be utilized as an upper bound on the total weight of an optimal solution with respect to the case with some budget constraint^(3,5).

Table 3 shows the performance of the IP solver on instances with $m = n = 300$ and $w_i, w_{m+i} \in \{1, 2, \dots, 10\}$. Again, we observe that the total weight is improved from the initial

Table 2 Performance of the IP solver on instances with $m = n = 200$ and $w_i = w_{m+i} = 1$.

No.	Total Weight, $f(x)$			cplex info.	
	$x = 0$	greedy	cplex	$ Q^* $	CPU [sec]
1	59	119	155	155	114.11
2	48	119	153	153	282.83
3	63	117	150	150	731.36
4	57	124	159	159	42.45
5	58	123	157	157	105.16
6	52	117	150	150	631.67
7	67	137	162	162	67.87
8	71	120	156	156	96.46
9	59	123	154	154	126.74
10	58	122	161	161	47.27
11	55	112	151	151	330.21
12	59	126	152	152	86.24
13	64	118	155	155	328.60
14	54	127	155	155	142.84
15	67	119	157	157	91.35
16	56	123	157	157	34.68
17	57	111	151	151	148.33
18	68	120	152	152	345.22
19	65	118	158	158	47.59
20	56	120	150	150	157.03
Ave.	59.7	120.8	154.8	154.8	197.90

reversing graph by the optimal solution of the IP solver. The IP solver finds an optimal solution at most 120 seconds (i.e., at most two minutes) for the twenty test instances.

Table 3 Performance of the IP solver on instances with $m = n = 300$ and $w_i, w_{m+i} \in \{1, 2, \dots, 10\}$.

No.	Total Weight, $f(x)$			cplex info.	
	$x = 0$	greedy	cplex	$ Q^* $	CPU [sec]
1	465	1222	1379	212	77.49
2	510	1195	1360	214	93.08
3	452	1240	1411	215	34.03
4	495	1267	1412	219	86.46
5	491	1301	1453	217	76.66
6	507	1197	1418	213	27.03
7	518	1239	1430	224	38.46
8	463	1275	1405	219	78.23
9	521	1290	1398	210	83.46
10	449	1297	1433	223	6.23
11	437	1273	1399	209	64.04
12	574	1240	1437	220	118.33
13	546	1256	1461	217	13.46
14	481	1270	1488	224	4.70
15	548	1222	1408	215	77.45
16	459	1312	1469	220	23.92
17	557	1242	1441	226	11.18
18	440	1255	1384	208	73.70
19	535	1255	1394	222	21.79
20	446	1205	1381	215	12.08
Ave.	494.7	1252.7	1418.1	217.1	51.09

Table 4 illustrates the performance of the IP solver on instances with $m = n = 300$ and $w_i = w_{m+i} = 1$. In this table, we give 600 seconds as the upper limit of the execution time to the IP solver. Hence, the solution obtained by the IP solver within the upper limit may not be optimal, and $|Q_{600}|$ denote the size of a clique obtained by the IP solver within the upper limit, instead of the $|Q^*|$. For the case with the unit weights, the total weight is highly improved from the initial reversing graph by the solution of the IP solver, although the solution may not be the optimal one. As in Table 2, we observe that the total weight of the solution by the IP solver is equal to the size of its clique. Unfortunately, the IP solver uses up the allowable execution time for all the twenty test instances. For the case with the unit weights, it would be interesting to examine a certain combination with an iterative improvement procedure, e.g., the simulated annealing^(5,10).

Table 4 Performance of the IP solver on instances with $m = n = 300$ and $w_i = w_{m+i} = 1$.

No.	Total Weight, $f(x)$			cplex info.	
	$x = 0$	greedy	cplex	$ Q_{600} $	CPU [sec]
1	89	170	231	231	601.90
2	92	166	228	228	600.50
3	83	178	223	223	600.44
4	90	178	233	233	600.50
5	92	179	233	233	600.25
6	94	168	233	233	600.26
7	91	194	236	236	600.26
8	88	196	233	233	600.39
9	93	182	229	229	600.54
10	89	188	238	238	600.42
11	84	177	232	232	600.30
12	96	169	230	230	600.40
13	97	184	236	236	600.45
14	87	173	236	236	600.47
15	90	181	231	231	600.26
16	79	185	235	235	600.28
17	95	195	238	238	600.48
18	86	181	230	230	600.33
19	98	199	239	239	600.36
20	79	186	227	227	600.28
Ave.	89.6	181.5	232.1	232.1	600.46

5. Conclusions

In this paper, we considered a weighted item collecting problem in directed bipartite graphs, where a finite set of items with non-negative weights and a finite set of players are contained, and no budget constraints are imposed on the players. We interpreted the weighted item collecting problem as a special case of the weighted maximum clique problem, and examined an application of integer programming (IP) solvers to an integer programming formulation of the weighted maximum clique problem in order to obtain an optimal solution for an

instance of the weighted item collecting problem.

We conducted numerical experiments to demonstrate the optimal solutions obtained by utilizing CPLEX as an IP solver. The IP solver found an optimal solution for each randomly generated instance with up to three hundred items with non-unit weights and three hundred players within two minutes of execution time, which implies that the problem transformation and the application of the IP solver significantly improve the performance of the Gray code based enumeration as an exact algorithm, although the weighted maximum clique problem is an NP-hard problem. We would regard the approach treated in this paper as a preliminary investigation of designing an exact algorithm for the weighted item collecting problem in directed bipartite graphs with budget constraints on the players.

However, for some randomly generated instances with two hundred items with the unit weights and two hundred players, the IP solver was not able to ensure the solutions obtained within ten minutes of execution time to be optimal, although the solutions highly improved the total weight in the initial reversing graph. For future research, it would also be interesting to examine a certain combination with an iterative improvement procedure such as the simulated annealing for the case with the unit weights.

Acknowledgment

This research was partially supported by JSPS KAKENHI Grant 19K04880.

References

- (1) Tang, M., An adaptive genetic algorithm for the minimal switching graph problem, *Lecture Notes in Computer Science*, Vol. 3448, Raidl, G.R. and Gottlieb, J. (Eds.): EvoCOP 2005, Springer, (2005), pp.224–233.
- (2) Tang, M., Eshraghian, K., and Cheung, H.N., An efficient approach to constrained via minimization for two-layer VLSI routing, *Proceedings of IEEE Asia and South Pacific Design Automation Conference*, (1999), pp.149–152.
- (3) Karuno, Y. and Tanaka, S., Cooperative item collecting problems in directed bipartite structures, *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, Vol.11, No.2, (2017), 16pages, [DOI: 10.1299/jamdsm.2017jamdsm0025].
- (4) Karuno, Y. and Tanaka, S., Weighted item collecting in directed bipartite structure with a reversing cost penalty, *Proceedings of the Fifth IIAE International Conference on Industrial Application Engineering (ICIAE 2017)*, (2017), pp.62–68.
- (5) Karuno, Y. and Tanaka, S., Iterative improvement approaches for collecting weighted items in directed bipartite graphs, *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, Vol.12, No.2, (2018), 13pages, [DOI: 10.1299/jamdsm.2018jamdsm0051].
- (6) Tanaka, S. and Karuno, Y., Compounding procedures for a weighted item collecting problem with a cost penalty term in directed bipartite structures, *Proceedings of the Sixth IIAE International Conference on Industrial Application Engineering (ICIAE 2018)*, (2018), pp.227–234.
- (7) Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, (1979), Freeman.
- (8) CPLEX, [<https://www.ibm.com/jp-ja/products/ilog-cplex-optimization-studio>] (Accessed on 5 May, 2019).
- (9) Skiena, S.S., *The Algorithm Design Manual*, (2008), Springer.
- (10) Aarts, E. and Korst, J., *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, (1989), Wiley.