Cooperative item collecting problems in directed bipartite structures

Yoshiyuki KARUNO* and Seiya TANAKA**

* Faculty of Mechanical Engineering, Kyoto Institute of Technology
Goshokaido-cho, Matsugasaki, Sakyo-ku, Kyoto-shi, Kyoto 606-8585, Japan
E-mail: karuno@kit.ac.jp

** Graduate School of Science and Technology, Kyoto Institute of Technology
Goshokaido-cho, Matsugasaki, Sakyo-ku, Kyoto-shi, Kyoto 606-8585, Japan

Abstract
In this paper, a variant of 0-1 knapsack problems in graph structures is considered. Given a directed bipartite structure with a set of items and a set of players, the problem asks to find an arc reversing strategy of the players which collects items with a budget constraint so that the total weighted profit of the collected items is maximized. More players than one are associated with an item by their directed arcs, and in order to win the weighted profit, it is indispensable for the players to make their arcs with respect to the item in the same direction. The situation of the same direction of associated arcs with an item is regarded as a cooperation of the associated players with the item, and the problem is called CIC (cooperative item collecting) for short. Problem CIC is viewed as a generalization of an integrated circuit design problem, in which a certain type of the cost of changes from an initial design is newly considered. A cooperation of associated players with an item is generally adverse to another item, and problem CIC is also seen as a compromising model in conflictive states. In this paper, some complexity results of problem CIC including the NP-hardness are discussed, and greedy heuristic algorithms are designed. Numerical experiments are conducted to demonstrate the performance of the greedy heuristic algorithms, and the results are reported.

Key words : Engineering optimization, 0-1 knapsack problems, Directed bipartite graphs, Cooperative arc reversing, Greedy heuristics

1. Introduction
In this paper, we consider a variant of 0-1 knapsack problems in graph structures. Given a directed bipartite structure with a set of items and a set of players, the problem asks to find an arc reversing strategy of the players which collects items with a budget constraint so that the total weighted profit of the collected items is maximized. As formulated in the following section, in order to collect a profitable item, the variant requires the players to behave as a team, and may ask a player to change his/her initial assertion in the opposite direction (that is, to reverse arc directions associated with the player in the given directed bipartite structure). In this paper, we regard the situation of the same direction of associated arcs with an item as a cooperation of the associated players with the item, and we call the problem CIC (cooperative item collecting) for short. A cooperation of associated players with an item is generally adverse to another item.

A family of 0-1 knapsack problems (including subset sum problems) in graph structures has attracted much attention of researchers and practitioners due to their applications (e.g., see Samphaiboon and Yamada, 2000; Wilbaut et al., 2008; Pferschy and Schauer, 2009). For example, a hopper selection issue of double-layered automatic combination weighers in actual food packaging can be formulated as a special case of 0-1 knapsack problems in precedence graphs (e.g., see Karuno et al., 2010). Problem CIC is viewed as a generalization of the minimal switching graph problem treated by Tang (2005). The minimal switching graph problem was introduced by Tang et al. (1999) to model a constrained via minimization in the context of integrated circuit design automation. Also in problem CIC, the set of items can represent the set of given via
candidates in a double-sided circuit, and the set of players can do the set of given wiring clusters. In the generalization, we can take into account some type of the cost of changes from an initial design by newly introducing a cost function defined on the players and the budget constraint. As mentioned in the above, a cooperation of associated players with an item is generally adverse to another item, and problem CIC is also seen as a compromising model in conflictive states, e.g., the appearance of a curative effect in a product development may depend on the interactive selection of several functional elements whether each of them is in or not in the product.

Tang (2005) presented an application of the genetic algorithm to the minimal switching graph problem, in which no budget constraint is imposed and all items have the unit profit. We do not directly treat the practical application of integrated circuit design automation, while we utilize a part of discussion on atomic building blocks by Tang (2005) to establish some mathematical properties of the cooperative item collecting problem. In this paper, we first discuss some complexity results of problem CIC including the NP-hardness. Then, referring the mathematical properties, we design greedy heuristic algorithms for problem CIC. We also conduct numerical experiments to demonstrate the performance of the greedy heuristics, and report the results.

2. Problem Description
2.1. Basic Notations and Terminologies

Let $I = \{i \mid i = 1, 2, \ldots, m\}$ denote a set of $m$ items, and let $J = \{j \mid j = 1, 2, \ldots, n\}$ denote a set of $n$ players. A positive profit $p_i$ is associated with each item $i \in I$, and a positive cost $c_j$ is associated with each player $j \in J$. On the $n$ players, a budget $b$ such that $0 < b \leq \sum_{j \in J} c_j$ and $b \geq c_j$ for each $j \in J$ is imposed. In particular, we call an instance of problem CIC (i.e., cooperative item collecting) with $b = \sum_{j \in J} c_j$ an enough budget instance.

Initially, each player $j \in J$ possesses his/her own assertion such that for each item $i \in I$, that is, the player wants to (i) pull it; (ii) push it; or (iii) make no response to it. For convenience, we assign a color to each kind of the first two relationships between an item and a player as follows. We refer to an item $i \in I$ as a blue-signaled item of player $j \in J$ if the player wants to pull it. On the other hand, we refer to an item $i \in I$ as a red-signaled item of player $j \in J$ if the player wants to push it. For each player $j \in J$, let $B_j$ denote the initial set of blue-signaled items, and let $R_j$ denote the initial set of red-signaled items, where $B_j \cap R_j = \emptyset$ must be satisfied. Without loss of generality, it is assumed that each player $j \in J$ meets

$$B_j \cup R_j \neq \emptyset,$$

which implies that each player sends at least one item a colored signal. Note that for an item $i \in I$ and for two distinct players $j, k \in J$, it is possible that both $i \in B_j$ and $i \in R_k$ hold. It is also assumed that for each item $i \in I$, there exists some player $j \in J$ with $i \in B_j \cup R_j$, i.e., at least one player sends a colored signal to each item.

We say that a player $j \in J$ is reversed when making his/her initial assertion (i.e., with respect to pulling and pushing items) change in the opposite direction. That is, for a reversed player $j \in J$, an item $i \in B_j$ is regarded as a red-signaled item of the player, while an item $i \in R_j$ as a blue-signaled item of the player. As a solution of the problem to be discussed in this paper, we define a reversing vector $x = (x_1, x_2, \ldots, x_n)$, where for each $j = 1, 2, \ldots, n$,

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

When the $n$ players accept a reversing vector $x$ as a cooperative arc reversing strategy of them, they pay the reversing cost

$$c(x) = \sum_{j=1}^{n} c_j x_j.$$  

A reversing vector $x$ is referred to as feasible if it satisfies the budget constraint, i.e., $c(x) \leq b$. Notice that the zero reversing vector $x = (x_1, x_2, \ldots, x_n)$ with $x_j = 0$ for all players $j \in J$ is feasible, since the $x = 0$ meets $c(0) = 0 \leq b$.

For each item $i \in I$, the player set $J$ is partitioned into the following three disjoint subsets with respect to a reversing vector $x$:

$$J_B(x, i) = \{j \in J \mid x_j = 0 \text{ and } i \in B_j\} \cup \{j \in J \mid x_j = 1 \text{ and } i \in R_j\},$$

$$J_R(x, i) = \{j \in J \mid x_j = 0 \text{ and } i \in R_j\} \cup \{j \in J \mid x_j = 1 \text{ and } i \in B_j\},$$

and $J \setminus (J_B(x, i) \cup J_R(x, i))$, where $J_B(x, i)$ (resp., $J_R(x, i)$) is the set of players who send the item $i$ blue signals (resp., red signals) in the solution $x$. We refer to a player $j \in J_B(x, i)$ as a blue-signaling player of item $i$ (i.e., the player wants to pull
the item) in the solution \( x \), and to a player \( j \in J_R(x, i) \) as a red-signaling player of item \( i \) (i.e., the player wants to push the item) in the \( x \). As explained later, an item \( i \in I \) such that it meets either \( J_R(x, i) = \emptyset \) or \( J_B(x, i) = \emptyset \) (that is, an item to which all the signaling players send the same color signals) is desired to win the weighted profit.

In the context of double-sided circuit design, a reversed player \( j \in J \) with \( x_j = 1 \) represents a wiring cluster to be moved from the initial side of the circuit board to the other side, and an item \( i \in I \) meeting \( J_B(x, i) \neq \emptyset \) and \( J_R(x, i) \neq \emptyset \) both corresponds to an undesirable via (see Tang et al., 1999). Recall that the same color signals of some players to an item means a cooperation of the players in this paper. The given \( c_j \) of each player \( j \in J \) indicates a cost of the change on the corresponding wiring cluster from the initial design.

### 2.2. Bipartite Representation of Assertions of the Players

In this subsection, we introduce a directed bipartite graph to represent blue and red signals from the players to the items (i.e., pulling and pushing assertions of the players), where the items and the players are regarded as vertices of the graph. Let \( G(x) = (I \cup J, A_B(x) \cup A_R(x)) \) denote a reversing graph with respect to a solution \( x \), where \( A_B(x) \) and \( A_R(x) \) denote disjoint arc sets such that \( A_B(x) \subseteq I \times J \) and \( A_R(x) \subseteq J \times I \). More precisely, an arc \((i, j) \in A_B(x)\) implies that for the item \( i \in I \), the player \( j \in J \) meets \( j \in J_B(x, i) \) (i.e., the \( j \) is a blue-signaling player of the item \( i \)), while an arc \((j, i) \in A_R(x)\) implies that for the item \( i \in I \), the player \( j \in J \) meets \( j \in J_R(x, i) \) (i.e., the \( j \) is a red-signaling player of the item \( i \)). The definition of the reversing graph \( G(x) \) is essentially equivalent to that of the bipartite graph treated by Tang (2005). 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 which the arcs in \( A_B(0) \cup A_R(0) \) represent the initial assertions of the players.

We provide an illustration of a problem instance by a reversing graph in Fig. 1, where we set the number of items by \( m = 6 \), the number of players by \( n = 8 \), the profit by \( p_i = 1 \) for each item \( i = 1, 2, \ldots, 6 \), the cost by \( c_j = 1 \) for each player \( j = 1, 2, \ldots, 8 \), and the budget by \( b = 4 \). We set the initial reversing graph \( G(0) \) as shown in (a) of the figure. For the first player (i.e., for \( j = 1 \)), two subsets \( B_1 = \{2\} \) and \( R_1 = \{4, 5\} \) of items are defined as the initial assertion of the player. Also, the second item (i.e., \( i = 2 \)) is a blue-signaled item of the first player (i.e., \( 2 \in B_1 \)), while it is a red-signaled item of the second player (i.e., \( 2 \in R_2 \)). We say that for the second item, the first and second players are not in their cooperation. In the cooperative sense of an arc reversing strategy of the players, they are in their cooperation for the first and sixth items in the initial \( G(0) \).

On the other hand, the reversing graph \( G(x) \) by \( x = (0, 1, 0, 1, 1, 1, 0, 0) \) is shown in (b) of the figure. The players pay the reversing cost \( c(x) = c_2 + c_4 + c_5 + c_6 = 4 \leq b = 4 \), and hence the \( x \) is a feasible solution. The second player (i.e., \( j = 2 \)) is reversed in the \( x \), and the second item (i.e., \( i = 2 \)) becomes a blue-signaled item of the second player. The first and second players send the same color signals to the second item, and we say that the two players are in their cooperation for the second item. In the cooperative sense of an arc reversing strategy of the players, they are in their cooperation for the first, second, third, fifth and sixth items in the \( G(x) \). For the calculation of the objective function value of a feasible solution \( x \), we continue to describe problem CIC a few more in the remainder of this section.

![Fig. 1](image-url) Directed bipartite representation: The initial reversing graph \( G(0) \) in (a), and another reversing graph \( G(x) \) by \( x = (0, 1, 0, 1, 1, 1, 0, 0) \) in (b), where an asterisk indicates a reversed player.
2.3. The Objective Function

Let
\[ I_B(x) = \{ i \in I \mid J_B(x, i) \neq \emptyset \text{ and } J_B(x, i) \neq \emptyset \}, \]
\[ I_R(x) = \{ i \in I \mid J_R(x, i) = \emptyset \text{ and } J_R(x, i) \neq \emptyset \}, \]
denote the subsets of items. The definitions imply that signals of an item \( i \in I_B(x) \) receiving from players are all blue in the solution \( x \), while signals of an item \( i \in I_R(x) \) receiving from players are all red in the \( x \). In other words, the players are in their cooperation for any item \( i \in I_B(x) \cup I_R(x) \). Again in (a) of Fig. 1, we observe that \( I_B(0) = \{ 6 \} \) and \( I_R(0) = \{ 1 \} \), while in (b) of the figure, \( I_B(x) = \{ 1, 2, 3, 6 \} \) and \( I_R(x) = \{ 5 \} \). We refer to an item \( i \in I_B(x) \) (resp., \( i \in I_R(x) \)) as a completely blue-signaled item (resp., a completely red-signaled item) in the solution \( x \).

In this paper, for a given coefficient \( \lambda \) of a linear combination with \( 0 \leq \lambda \leq 1 \), the following total weighted profit is maximized as the objective function:
\[ f(x) = (1 - \lambda)f_B(x) + \lambda f_R(x), \]
where we define
\[ f_B(x) = \sum_{i \in I_B(x)} p_i, \]
\[ f_R(x) = \sum_{i \in I_R(x)} p_i. \]

As mentioned in the above, completely blue-signaled items and completely red-signaled items are desired in the cooperative sense of an arc reversing strategy of the players. In the problem instance of Fig. 1, when \( \lambda = 0.5 \), the reversing vector \( x = (0, 1, 0, 1, 1, 1, 0, 0) \) in (b) of the figure takes the objective function value \( f(x) = 0.5 \times (p_1 + p_2 + p_3 + p_6) + 0.5 \times p_5 = 2.0 + 0.5 = 2.5 \), while the zero reversing vector in (a) of the figure takes \( f(0) = 0.5 \times p_6 + 0.5 \times p_1 = 0.5 + 0.5 = 1.0 \).

Let \( x = x' \) denote a feasible solution which attains the objective function value such that it satisfies \( f(x') \geq f(x) \) for any feasible solution \( x \). We refer to the \( x' \) as an optimal solution, and let \( f^- = f(x') \) denote the optimal value. Formally, problem CIC asks to find an optimal solution \( x = x' \).

As stated before, the minimal switching graph problem treated by Tang (2005) in the context of integrated circuit design automation is viewed as a special case of problem CIC, where only enough budget instances are considered, and each item has the unit profit, and further completely blue-signaled and completely red-signaled items are evenly desired, i.e., the coefficient of the linear combination is fixed to be \( \lambda = 0.5 \) in the objective function of Eq. (8).

3. Complexity Results

Before designing a heuristic algorithm for problem CIC, we see that the problem is NP-hard. For this, we show a polynomial time reduction from the traditional 0-1 knapsack problem in an extra section.

**Theorem 1.** Problem CIC is NP-hard.

**Proof.** See Appendix. ☐

In the remainder of this section, we consider polynomially solvable cases of problem CIC in order to establish some mathematical properties. The mathematical properties are going to be utilized for designing greedy heuristic algorithms in the following section.

3.1. Counterpart Vectors

For a reversing vector \( x = (x_1, x_2, \ldots, x_n) \), we define the counterpart vector (or simply, counterpart) of the \( x \) by \( \overline{x} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) \), where each player \( j \in J \) satisfies
\[ \overline{x}_j = 1 - x_j. \]

Note that the counterpart of the \( \overline{x} \) is the original reversing vector \( x \).

**Lemma 1.** For an instance of problem CIC, a reversing vector \( x \) and the counterpart \( \overline{x} \) of the \( x \) satisfy
\[ I_B(x) = I_B(\overline{x}), \]
\[ I_R(x) = I_R(\overline{x}). \]
Proven. From Eqs. (4) and (5), we have

\[ J_B(i,j) = \{ j \in J \mid x_j = 0 \text{ and } i \in B_j \} \cup \{ j \in J \mid x_j = 1 \text{ and } i \in R_j \} \]

\[ J_B(i,j) = \{ j \in J \mid x_j = 0 \text{ and } i \in B_j \} \cup \{ j \in J \mid x_j = 1 \text{ and } i \in R_j \} = J_B(\overline{x}, i). \]

From Eqs. (6) and (7), we then obtain

\[ I_B(x) = \{ i \in J \mid J_B(x, i) \neq \emptyset \} \quad \text{and} \quad J_B(x, i) = \emptyset \}

\[ I_B(x) = \{ i \in J \mid J_B(x, i) \neq \emptyset \} \quad \text{and} \quad J_B(x, i) = \emptyset \}

which complete the proof. \( \square \)

The lemma implies that if a solution \( x \) and its counterpart \( \overline{x} \) are both feasible, then from Eqs. (8)–(10), the objective function values meet

\[ f(x) + f(\overline{x}) = ((1 - \lambda)f_B(x) + Af(x)) + ((1 - \lambda)f_B(\overline{x}) + Af(\overline{x})) \]

\[ = (1 - \lambda) \sum (p_i \mid i \in I_B(x)) + \sum (p_i \mid i \in I_B(\overline{x})) \]

\[ + (1 - \lambda) \sum (p_i \mid i \in I_B(\overline{x})) + \lambda \sum (p_i \mid i \in I_B(x)) \]

\[ = (1 - \lambda) \sum (p_i \mid i \in I_B(x)) + \lambda \sum (p_i \mid i \in I_B(\overline{x})) \]

\[ + (1 - \lambda) \sum (p_i \mid i \in I_B(\overline{x})) + \lambda \sum (p_i \mid i \in I_B(x)) \]

\[ = f_B(x) + f_B(\overline{x}). \] (14)

That is, we can choose the better solution than the other between the \( x \) and the counterpart \( \overline{x} \), regarding the coefficient \( \lambda \), if the solutions are both feasible.

We now consider an important simple instance of problem CIC with exactly one item, i.e., with \( m = 1 \).

Lemma 2. For an instance of problem CIC with \( m = 1 \), an optimal solution can be obtained in \( O(n) \) time.

Proven. From the assumption of Eq. (1), each player \( j \in J \) initially sends either a blue signal or a red signal to the single item. Hence, the set \( J \) of players is initially partitioned into \( J_B(0, 1) \) and \( J_B(0, 1) = J \setminus J_B(0, 1) \) (see Eqs. (4) and (5)). Either \( J_B(0, 1) = \emptyset \) or \( J_B(0, 1) = J \) may be possible to hold.

First, we construct a reversing vector \( x = (x_1, x_2, \ldots, x_n) \) in \( O(n) \) time such that for each player \( j \in J \),

\[ x_j = \begin{cases} 1 & \text{if } j \in J_B(0, 1), \\ 0 & \text{otherwise}. \end{cases} \] (15)

Then, we obtain the counterpart \( \overline{x} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) \) of the \( x \) also in \( O(n) \) time such that for each player \( j \in J \),

\[ \overline{x}_j = \begin{cases} 1 & \text{if } j \in J_B(0, 1), \\ 0 & \text{otherwise}. \end{cases} \] (16)

In the \( x \), the single item becomes the unique completely blue-signal-ed item, while in the counterpart \( \overline{x} \), it becomes the unique completely red-signal-ed item. No other reversing vector can make the single item either a completely blue-signal-ed or a completely red-signal-ed one.

The feasibility of a reversing vector on the budget constraint can be tested in \( O(n) \) time (see Eq. (3)). Hence, if the \( x \) and the counterpart \( \overline{x} \) are both infeasible (\( x \neq 0 \) and \( \overline{x} \neq 0 \) hold in this case), then the zero reversing vector is an optimal solution with the optimal value \( f^* = 0 \). If the \( x \) and the counterpart \( \overline{x} \) are both feasible, we can choose the better solution than the other between the \( x \) and \( \overline{x} \) as an optimal solution with respect to the objective function values \( f(x) \) and \( f(\overline{x}) \) (see Eq. (14)), which implies that in this case, \( f(x) + f(\overline{x}) = p_1 \). If the \( x \) (resp., the counterpart \( \overline{x} \)) is feasible and the counterpart \( \overline{x} \) (resp., the \( x \)) is infeasible, we can take the \( x \) (resp., the counterpart \( \overline{x} \)) as an optimal solution with the optimal value \( f^* = f(x) = (1 - \lambda)p_1 \) (resp., \( f^* = f(\overline{x}) = Ap_1 \)). Therefore, we obtain the lemma statement for the simple case in which exactly one item exists. \( \square \)
3.2. Basic Semi-solutions

In this subsection, together with Lemmas 1 and 2, we extend the discussion on atomic building blocks by Tang (2005) for the minimal switching graph problem, which is going to establish the complexity of problem CIC with \( m = 2 \) in the following subsection.

A reversing vector \( x = (x_1, x_2, \ldots, x_n) \) can be regarded as a string of \( n \) binary bits. Introducing the ordinary symbol \( * \) for representing "no care" in a bit (e.g., see Tang, 2005), we define a semi-solution \( s = [s_1, s_2, \ldots, s_n] \), where for each \( j = 1, 2, \ldots, n \),

\[
s_j = \begin{cases} 
1 & \text{if player } j \text{ is reversed}, \\
0 & \text{if player } j \text{ is not reversed}, \\
* & \text{if player } j \text{ is not cared}.
\end{cases}
\] (17)

We also define the counterpart semi-solution \( \overline{s} \) of the \( s \), where for each \( j = 1, 2, \ldots, n \),

\[
\overline{s}_j = \begin{cases} 
1 & \text{if } s_j = 0, \\
0 & \text{if } s_j = 1, \\
* & \text{if } s_j = *.
\end{cases}
\] (18)

We first consider the objective function with \( \lambda = 0 \) (see Eq. (8)). That is, for an item \( i \in I \), the players can take the profit \((1 - \lambda)p_i = p_i\) only if all signaling players to the item send blue signals. We define a blue basic semi-solution \( s^{(i)} \) for each item \( i \in I \), which summarizes all the solutions taking the profit \((1 - \lambda)p_i \) of the item as a completely blue-signaled one. For the problem instance provided in Fig. 1, there are the following six blue basic semi-solutions:

\[
s^{(1)} = [*, *, *, 1, 1, *,*,*], \\
s^{(2)} = [0, 1, *,*,* ,*,*,*], \\
s^{(3)} = [*, *, *, 0,*,*,*,*], \\
s^{(4)} = [1,*,*,*,*,0,*,*], \\
s^{(5)} = [1,*,*,*,*,0,1,*,*], \\
s^{(6)} = [*,*,0,*,*,*,*,0,0].
\]

From the \( s^{(1)} \), for example, we see that any solution \( x \) such that it satisfies \( x_4 = x_5 = 1 \) and \( c(x) \leq b \) can take the profit \((1 - \lambda)p_i\) as that of a completely blue-signaled item.

In the similar manner, we consider the objective function with \( \lambda = 1 \) (again see Eq. (8)), and we define a red basic semi-solution \( s^{(m+1)} \) for each item \( i \in I \), which summarizes all the solutions taking the profit \( \lambda p_i \) of the item as a completely red-signaled one. Notice that from Lemma 1, \( s^{(m+1)} = \overline{s}^{(i)} \) holds. For the problem instance provided in Fig. 1, there are the following six red basic semi-solutions:

\[
s^{(7)} = \overline{s}^{(1)} = [*,*,*,0,0,*,*,*], \\
s^{(8)} = \overline{s}^{(2)} = [1,0,*,*,*,*,*,*], \\
s^{(9)} = \overline{s}^{(3)} = [*,*,*,0,*,*,*,*], \\
s^{(10)} = \overline{s}^{(4)} = [0,*,*,*,*,*,*,1], \\
s^{(11)} = \overline{s}^{(5)} = [0,*,*,*,*,1,0,*,], \\
s^{(12)} = \overline{s}^{(6)} = [*,1,*,*,*,1,1,1].
\]

Also, we see from the \( s^{(7)} = \overline{s}^{(1)} \) that any solution \( x \) such that it satisfies \( x_4 = x_5 = 0 \) and \( c(x) \leq b \) can take the profit \( \lambda p_i \) as that of a completely red-signaled item. All the \( 2m \) basic semi-solutions can be obtained in \( O(mn) \) time (see also Tang, 2005).

Conversely, if a certain item \( i \in I \) can be a completely blue-signaled item (resp., a completely red-signaled item) in a solution \( x = (x_1, x_2, \ldots, x_n) \), the \( x \) contains the blue basic semi-solution \( s^{(i)} = [s^{(i)}_1, s^{(i)}_2, \ldots, s^{(i)}_n] \) (resp., the red basic semi-solution \( \overline{s}^{(i)} = [\overline{s}^{(i)}_1, \overline{s}^{(i)}_2, \ldots, \overline{s}^{(i)}_n] (= s^{(m+1)}) \), i.e., for each \( j = 1, 2, \ldots, n \), \( x_j = s^{(i)}_j \) holds if \( s^{(i)}_j \in \{0,1\} \) (resp., \( x_j = \overline{s}^{(i)}_j \) holds if \( \overline{s}^{(i)}_j \in \{0,1\} \)). We comprehensively say that a reversing vector \( x = (x_1, x_2, \ldots, x_n) \) contains a semi-solution \( s = [s_1, s_2, \ldots, s_n] \) if \( x_j = s_j \) holds for any player \( j \in J \) with \( s_j \in \{0,1\} \), and for notational convenience, we express it by \( x \supseteq s \). From a given semi-solution \( s \), we obtain a minimal \( x \supseteq s \) of reversing vector by a trivial \( O(n) \) time procedure as follows:

**Procedure** MINIMAL\_SOLUTION($s; x$)

Input: A semi-solution $s = [s_1, s_2, \ldots, s_n]$.
Output: A reversing vector $x = (x_1, x_2, \ldots, x_n)$ such that $x \succeq s$.

Step 1. For each player $j = 1, 2, \ldots, n$, let
\[
x_j = \begin{cases} 
    s_j & \text{if } s_j \in [0, 1], \\
    0 & \text{otherwise (i.e., if } s_j = *). 
\end{cases}
\]

Step 2. Terminate the computation.

In this paper, we further define the minimal cost of a semi-solution $s = [s_1, s_2, \ldots, s_n]$ by
\[
c_{\min}(s) = \sum \{ c_j \mid j \in J \text{ with } s_j = 1 \}. 
\]

(20)

For a semi-solution $s$, let $x$ temporarily be a minimal reversing vector of it obtained by procedure MINIMAL\_SOLUTION. Then it holds $c_{\min}(s) = c(x)$. Hence, if it holds $c_{\min}(s^{(i)}) \leq b$ (resp., $c_{\min}(\bar{s}^{(i)}) \leq b$) for an item $i \in I$, there exists a feasible solution $x$ with $f(x) \geq (1 - \lambda)p_i$ (resp., with $f(x) \geq \lambda p_i$).

### 3.3. Consistency of Semi-solutions

In this subsection, we introduce the following new concept for the semi-solutions. We say that two distinct semi-solutions $s = [s_1, s_2, \ldots, s_n]$ and $s' = [s'_1, s'_2, \ldots, s'_n]$ are inconsistent if there exists some player $j \in J$ who meets all the following conditions (i)–(iii):

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

(21)

Otherwise (i.e., if $s_j = s'_j$, $s_j = *$ or $s'_j = *$ holds for any player $j = 1, 2, \ldots, n$), they are consistent. The definition implies that for any item $i \in I$, the blue and red basic semi-solutions $s^{(i)}$ and $\bar{s}^{(i)}$ are inconsistent. In the problem instance provided in Fig. 1, two semi-solutions $s^{(1)}$ and $s^{(8)} = \bar{s}^{(3)}$ are consistent. For two distinct semi-solutions, their consistency can obviously be checked in $O(n)$ time by Eq. (21). For convenience, we give the $O(n)$ time checking the following function name:

**Function** CONSISTENCY($s, s'$)

Input: Two distinct semi-solutions $s$ and $s'$.
Output: If the two semi-solutions $s$ and $s'$ are consistent, then return $true$; otherwise, return $false$.

Also for convenience, we here give the $O(n)$ time budget checking of Eq. (20) the following function name:

**Function** BUDGET($s$)

Input: A semi-solution $s$.
Output: If it holds $c_{\min}(s) \leq b$, then return $true$; otherwise, return $false$.

With respect to the consistency of two distinct semi-solutions, we obtain the following lemma:

**Lemma 3.** For two distinct semi-solutions $s$ and $s'$, there is a reversing vector $x$ such that it meets $x \succeq s$ and $x \succeq s'$ both if and only if the two distinct semi-solutions $s$ and $s'$ are consistent.

**Proof.** Suppose that the $s$ and $s'$ are consistent. Then, we obtain a semi-solution $\tilde{s}$ from the two semi-solutions $s$ and $s'$ such that for each player $j = 1, 2, \ldots, n$, it holds
\[
\tilde{s}_j = \begin{cases} 
    s_j (= s'_j) & \text{if } s_j = s'_j \in [0, 1], \\
    s_j & \text{if } s_j \in [0, 1] \text{ and } s'_j = *, \\
    s'_j & \text{if } s_j = * \text{ and } s'_j \in [0, 1], \\
    * & \text{if } s_j = s'_j = *.
\end{cases}
\]

(22)
since no player \( j \in J \) meets Eq. (21). For the semi-solution \( \hat{s} \), we further call procedure \( \text{MINIMAL\_SOLUTION}(\hat{s}; x) \).

From Eq. (22), the resulting \( x \) of reversing vector clearly satisfies \( x_j = s_j \) for any \( j \in J \) with \( s_j \in [0, 1] \) and \( x_j = s_j' \) for any \( j \in J \) with \( s_j' \in [0, 1] \). Hence, we have \( x \equiv s \) and \( x \equiv s' \).

On the other hand, suppose that there is a reversing vector \( x \) such that it meets \( x \equiv s \) and \( x \equiv s' \) both. Then, either \( s_j = * \) or \( s_j' = * \) holds for a player \( j \in J \) with \( s_j \neq s_j' \), since \( x_j = s_j = s_j' \) holds for a player \( j \in J \) such that \( s_j \in [0, 1] \) and \( s_j' \in [0, 1] \). This implies that there is no player \( j \in J \) satisfying Eq. (21), and hence the two semi-solutions \( s \) and \( s' \) are consistent.

As seen in the proof of Lemma 3, when two distinct semi-solutions \( s \) and \( s' \) are consistent, we can make a compound semi-solution \( \hat{s} \) of them by Eq. (22). Henceforth, we use the notation \( \oplus \) to denote the compounding operator for two distinct semi-solutions. Note that for two distinct semi-solutions \( s \) and \( s' \) which are consistent, it holds \( s \oplus s' = s' \oplus s \).

Moreover, if any two distinct semi-solutions among \( s \), \( s' \) and \( s'' \) are consistent, it then holds \( (s \oplus s') \oplus s'' = s \oplus (s' \oplus s'') \).

We provide the following procedure:

**Procedure COMPOUND**\((s, s'; \hat{s})\)

Input: Two distinct semi-solutions \( s = [s_1, s_2, \ldots, s_n] \) with \( c_{\min}(s) \leq b \) and \( s' = [s_1', s_2', \ldots, s_n'] \).

Output: A semi-solution \( \hat{s} = [\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_n] \) (such that either \( \hat{s} = s \oplus s' \) or \( \hat{s} = s \)).

Step 1. Call function \( \text{CONSISTENCY}(s, s') \). If \( \text{CONSISTENCY}(s, s') \) is false, then go to Step 4.

Step 2. Let \( \hat{s} = s \oplus s' \) be a compound semi-solution by Eq. (22).

Step 3. Call function \( \text{BUDGET}(\hat{s}) \). If \( \text{BUDGET}(\hat{s}) \) is true, then go to Step 5.

Step 4. Let \( \hat{s} = s \) (when either \( \text{CONSISTENCY}(s, s') \) or \( \text{BUDGET}(s \oplus s') \) is false).

Step 5. Terminate the computation.

Each of the first four steps in procedure COMPOUND requires \( O(n) \) time, and hence the procedure runs in \( O(n) \times O(1) = O(n) \) time.

Consider again two semi-solutions \( s'(1) \) and \( s'(2) = \vec{s}_{(2)} \) in the problem instance of Fig. 1, which are consistent. Procedure COMPOUND makes the compound semi-solution \( \hat{s} = s'(1) \oplus \vec{s}_{(2)} = [1, 0, *, 1, *, 1, *, *, *, *] \).

From the compound semi-solution \( \hat{s} = s'(1) \oplus \vec{s}_{(2)} \), we see that any solution \( x \) such that it satisfies \( x_1 = x_4 = x_5 = 1 \), \( x_2 = 0 \) and \( c(x) \leq b \) can take the total weighted profit \( f(x) \geq (1 - \lambda)p_1 + \lambda p_2 \).

For notational convenience, we express the semi-solution \( s = [s_1, s_2, \ldots, s_n] \) with \( s_j = * \) for all \( j \in J \) by \( s = s' \).

Notice that a minimal solution of the semi-solution \( s = s' \) is the zero reversing vector \( x = 0 \). From Lemma 2, we see that in an optimal solution, each item is in one of three possible states, i.e., completely blue-signaled, completely red-signaled, or neither. This implies that it is sufficient for an exact algorithm to enumerate \( O(m^3) \) solutions. It is obvious that it is also sufficient for an exact algorithm to enumerate \( O(n^3) \) solutions from the viewpoint of the set of players. Hence, there generally exists an \( O(mn \min(3^n, 2^n)) \) time enumerative exact algorithm. For an instance of problem CIC with \( m = 2 \), it suffices to examine the consistency of each pair of two distinct semi-solutions \( s \in \{s', s'(1), \vec{s}_{(1)}\} \) and \( s' \in \{s', s'(2), \vec{s}_{(2)}\} \), and check the feasibility on the budget constraint of the compound semi-solution \( s \oplus s' \).

Therefore, we obtain the following theorem:

**Theorem 2.** For an instance of problem CIC with \( m = 2 \), an optimal solution can be obtained in \( O(n) \) time.

For the case of \( m = 2 \), an exact algorithm which enumerates at most \( 3 \times 3 = 9 \) possible pairs of two distinct semi-solutions \( s \in \{s', s'(1), \vec{s}_{(1)}\} \) and \( s' \in \{s', s'(2), \vec{s}_{(2)}\} \) is described as algorithm ENUMERATE (see Algorithm 1). The time complexity is obviously \( O(n) \), since the algorithm calls procedures COMPOUND and MINIMAL\_SOLUTION in the internal for-loop at most six times, i.e., \( O(n) \times O(1) = O(n) \).

When \( s = s' \) and \( s' = s' \), procedure COMPOUND returns \( \hat{s} = s' \), and hence procedure MINIMAL\_SOLUTION returns the zero reversing vector \( x = 0 \). The initial setting of an optimal solution in the algorithm allows us to omit the checking for the pair of \( s = s' \) and \( s' = s' \), and to make the algorithm return a feasible solution for an instance of problem CIC with \( m = 2 \).

From Lemma 3, we may design a heuristic algorithm from the viewpoint of a constrained maximum weighted clique problem (e.g., see Skiena, 2008). However, as the first step of ours for problem CIC, by calling the compounding procedure repeatedly, we design greedy heuristics in the following section.
Algorithm 1 ENUMERATE

Input: An instance of problem CIC with \( m = 2 \).
Output: An optimal solution \( x' = (x'_1, x'_2, \ldots, x'_n) \).
1: \( x' := 0; f' := f(0); \) /* Initialization of an optimal solution */
2: for each \( s \in \{x', x''(1), x''(2)\} \) in this order do
3:   if (\( \text{BUDGET}(s) \) is true) then
4:     for each \( s' \in \{x''(2), x''(3)\} \) in this order do
5:       Call procedure COMPOUND(s, s'; \( \delta \));
6:     Call procedure MINIMAL_SOLUTION(\( \delta; x \));
7:     if (\( f(x) > f' \)) then
8:       \( x' := x; f' := f(x') \); end if
9:   end if
10: end for
11: end if
12: end for /* End of the algorithm */

4. Greedy Heuristic Algorithms

For simplicity, we treat blue basic semi-solutions and red basic semi-solutions individually in the first greedy algorithm, GREEDY-1A (see Algorithm 2). We assume that basic semi-solutions \( s^{(i)} \) and \( x^{(i)} \) for each item \( i \in I \) have been obtained from the given directed bipartite structure by an \( O(mn) \) time preprocessing (see Section 3.2).

Algorithm 2 GREEDY-1A

Input: An instance of problem CIC, i.e., a set \( I \) of \( m \) items, a set \( J \) of \( n \) players, a positive profit \( p_i \) for each item \( i \in I \), a positive cost \( c_j \) for each player \( j \in J \), a positive budget \( b \), basic semi-solutions \( s^{(i)} \) and \( x^{(i)} \) for each item \( i \in I \).
Output: A heuristic solution \( x' = (x'_1, x'_2, \ldots, x'_n) \).
1: Renumber the \( m \) items so that \( p_1 \geq p_2 \geq \cdots \geq p_m \);
2: \( x' := 0; f' := f(0); \) /* Initialization of the heuristic solution */
3: for \( k = 1 \) to 2 do
4:   /* \( k = 1 \) indicates the blue only part, while \( k = 2 \) does the red only part. */
5:   \( s := s'_k \) (i.e., \( s_j := * \) for all \( j = 1, 2, \ldots, n \));
6:   for \( i = 1 \) to \( m \) do
7:     if (\( k = 1 \)) then
8:       \( s'_k := 0; f'_k := f(0); \) /* Compounding the same color basic semi-solutions in a greedy manner */
9:     else (i.e., \( k = 2 \))
10: \( s'_k := \pi^{(i)} \);
11: end if
12: Call procedure COMPOUND(s, s'; \( \delta \));
13: end for
14: for \( i = 1 \) to \( n \) do
15: Call procedure MINIMAL_SOLUTION(s; x);
16: if (\( f(x) > f' \)) then
17: \( x' := x; f' := f(x') \);
18: end if
19: Construct the counterpart \( x' \) of the \( x \); /* See Lemma 1 */
20: if (\( c(x) \leq b \) and \( f(x) > f' \)) then
21: \( x' := \pi; f' := f(x') \);
22: end if
23: end for /* End of the algorithm */

Next, we improve the blue only part (resp., the red only part) in algorithm GREEDY-1A into the blue-first-red-second (resp., the red-first-blue-second) part in algorithm GREEDY-2A (see Algorithm 3). In both algorithms GREEDY-1A and GREEDY-2A, the \( m \) items are initially sorted in the non-increasing order of their profits. In order to examine the effect of the initial sorting of the items, we are going to demonstrate the empirical performance of the greedy algorithms without the initial sorting of the items, which we call GREEDY-1B and GREEDY-2B, respectively. Further, in the initial sorting of the items, we may be able to use a modified profit, e.g.,

\[
p'_i = \frac{p_i}{d_i} \times \frac{\alpha}{(1 - \alpha)c_{\text{min}}(s^{(i)}) + \alpha c_{\text{min}}(x^{(i)}) + 1},
\]

where \( d_i \) denotes the degree of item \( i \in I \) in the reversing graph \( G(0) \) and \( \alpha \geq 1 \) is a scaling constant for actual computing, instead of the raw profit \( p_i \). By this, we intend to give a higher priority in the greedy manner to an item with a smaller
degree and a smaller minimal cost of the corresponding basic semi-solution. We call the better choices between the greedy algorithms with the raw and modified profits GREEDY-1C and GREEDY-2C, respectively.

**Algorithm 3 GREEDY-2A**

**Input:** An instance of problem CIC, i.e., a set $I$ of $m$ items, a set $J$ of $n$ players, a positive profit $p_i$ for each item $i \in I$, a positive cost $c_j$ for each player $j \in J$, a positive budget $b$, basic semi-solutions $s^i$ and $\pi^i$ for each item $i \in I$.

**Output:** A heuristic solution $x' = (x'_1, x'_2, \ldots, x'_n)$.

1. Renumber the $m$ items so that $p_1 \geq p_2 \geq \cdots \geq p_m$.
2. $x' := (0); f' := (0); /* Initialization of the heuristic solution */$
3. for $k = 1$ to 2 do
   4. /* $k = 1$ indicates the blue-first-red-second part, while $k = 2$ does the red-first-blue-second part */
   5. $s := s'$ (i.e., $s_j := t$ for all $j = 1, 2, \ldots, n$);
   6. for $i = 1$ to $m$ do
      7. /* This is the same for-loop as that in algorithm GREEDY-1A */
      8. if $(k = 1)$ then
         9. $x'_i := s'^i; /* Blue first */$
      10. else (i.e., $k = 2$)
          11. $x'_i := s'^i; /* Red first */$
      12. end if
   13. Call procedure COMPOUND($s, s'; \delta$); /* Compounding the first color basic semi-solutions in a greedy manner */
   14. $s := \delta$;
   15. end for
16. for $i = 1$ to $m$ do
   17. /* This is the additional for-loop */
   18. if $(k = 1)$ then
   19. $x'_i := s'^i; /* Red second */$
   20. else (i.e., $k = 2$)
   21. $s'_i := s'^i; /* Blue second */$
   22. end if
   23. Call procedure COMPOUND($s, s'; \delta$); /* Additional compounding of the second color basic semi-solutions */
   24. /* If the current $s$ has contained a basic semi-solution $s'^i$, then the counterpart $\pi'^i$ is automatically detected to be inconsistent with the $s$ by the procedure. */
   25. $s := \delta$;
   26. end for
27. Call procedure MINIMAL-SOLUTION($s, x$);
28. if $(f(x) > f')$ then
29. $x' := x; f' := f(x)$;
30. end if
31. Construct the counterpart $\pi$ of the $x$; /* See Lemma 1 */
32. if $(c(\pi) \leq b$ and $f(\pi) > f')$ then
33. $x' := \pi; f' := f(x)$;
34. end if
35. end for
36. end for /* End of the algorithm */

The initial sorting of the $m$ items requires $O(m \log m)$ time. Since the time complexity of procedure COMPOUND is $O(n)$, it takes $O(mn)$ time to perform a for-loop of $i = 1, 2, \ldots, m$ in algorithms GREEDY-1A and GREEDY-2A. Hence, both of the proposed greedy algorithms run in $O(m \log m + mn)$ time.

5. Numerical Results

The program is written in C language, and is run on a laptop personal computer with Windows 10 Pro (64bit), Intel Core i7 6500U CPU (2.50 GHz) and 8GB memory. Before examining the proposed greedy heuristic algorithms on randomly generated instances of problem CIC, we apply them to an instance of the minimal switching graph problem with $m = n = 48$, which has been produced by Tang (2005) as a benchmark. From the viewpoint of problem CIC, the instance is an enough budget one, and the coefficient of the linear combination in the objective function is set to be $\lambda = 0.5$. The result is shown in Table 1. We observe that both greedy heuristics GREEDY-1A and GREEDY-2A find an optimal solution of the instance with short execution times less than 1 [msec]. The genetic algorithm presented by Tang (2005) has been reported to find an optimal solution of the instance with execution times 5.69–22.38 [sec]. However, Tang (2005) did not give the detail of computational environment, and it would be hard for us to make the direct comparison between the execution times of the genetic algorithm and proposed greedy heuristics.

Instances of problem CIC to be tested are randomly generated as follows:

- The number of items: $m \in \{20, 50, 80\}$. 

[DOI: 10.1299/jamdsm.2017jamdsm0025]
The number of players: \( n = m \).

The profits of items: \( p_i = 1 \) for all \( i \in I \), or uniformly random integers \( p_i \in \{1, 2, \ldots, 10\} \).

The costs of players: \( c_j = 1 \) for all \( j \in J \), or uniformly random integers \( c_j \in \{1, 2, \ldots, 10\} \).

The budget: \( b \in \left\{ \sum_{j \in J} c_j, 0.75 \times \sum_{j \in J} c_j, 0.5 \times \sum_{j \in J} c_j, 0.25 \times \sum_{j \in J} c_j \right\} \).

The number of signaling players of each item \( i \in I \) (i.e., the degree of the item): \( d_i = |\{ j \in J \mid i \in B_j \cup R_j \}| \in \{2, 3, 4\} \) with probabilities 0.35, 0.35 and 0.30, respectively. (For example, at most four wires may be associated with a typical via of the circuit design, which are situated at the via in a perpendicular manner.)

The total number of arcs between the sets \( I \) and \( J \): \(|A_p(0) \cup A_R(0)| = \min \left\{ \sum_{i \in I} d_i, mn \right\} \). From the above setting of degrees \( d_i \), the sum \( \sum_{i \in I} d_i \) always takes the minimum in the arc generation of test instances. The direction of each arc is chosen between two alternatives with the same probability. The arcs are generated so that Eq. (1) is also satisfied.

The coefficient of the linear combination in the objective function: \( \lambda \in \{0.00, 0.25, 0.50, 0.75, 1.00\} \).

In all numerical results to be shown in the following tables, each of the data indicates the mean value in a set of one hundred test instances.

### Table 1

<table>
<thead>
<tr>
<th>Coefficient in ( f )</th>
<th>GREEDY-1A</th>
<th>GREEDY-2A</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>( f(x) )</td>
<td>CPU Time [msec]</td>
</tr>
<tr>
<td>0.5</td>
<td>24.0</td>
<td>0.10</td>
</tr>
</tbody>
</table>

\( m = n = 48, p_i = 1 \) for all \( i \in I \), \( c_j = 1 \) for all \( j \in J \), \( b = n \)

### Table 2

<table>
<thead>
<tr>
<th>Coefficient in ( f )</th>
<th>Budget</th>
<th>( p_i = 1 ) and ( c_j = 1 )</th>
<th>( p_i, c_j \in {1, 2, \ldots, 10} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>( b / \sum c_j )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1A</td>
<td>2A</td>
<td>( f(0) )</td>
<td>( f^* )</td>
</tr>
<tr>
<td>0.00</td>
<td>6.2</td>
<td>6.2</td>
<td>3.1</td>
</tr>
<tr>
<td>0.25</td>
<td>5.4</td>
<td>5.4</td>
<td>3.0</td>
</tr>
<tr>
<td>0.50</td>
<td>5.1</td>
<td>5.1</td>
<td>3.0</td>
</tr>
<tr>
<td>0.75</td>
<td>5.7</td>
<td>5.7</td>
<td>3.0</td>
</tr>
<tr>
<td>1.00</td>
<td>6.6</td>
<td>6.6</td>
<td>3.0</td>
</tr>
</tbody>
</table>

| 0.00 | 8.4 | 8.4 | 3.1 | 10.1 | 55.2 | 55.3 | 16.8 | 60.4 |
| 0.25 | 5.6 | 5.6 | 3.0 | 7.3 | 34.8 | 36.3 | 16.6 | 42.5 |
| 0.50 | 6.9 | 7.1 | 3.0 | 8.5 | 44.5 | 45.3 | 16.4 | 49.5 |
| 0.75 | 8.4 | 8.4 | 3.0 | 10.1 | 55.6 | 55.8 | 16.3 | 60.6 |
| 1.00 | 8.6 | 8.6 | 3.1 | 10.2 | 56.7 | 56.8 | 16.8 | 61.3 |

| 0.00 | 7.1 | 7.3 | 3.0 | 8.5 | 45.7 | 46.4 | 16.7 | 50.2 |
| 0.25 | 5.7 | 5.9 | 3.0 | 7.3 | 35.0 | 36.1 | 16.6 | 42.5 |
| 0.50 | 7.1 | 7.3 | 3.0 | 8.5 | 45.7 | 46.4 | 16.4 | 50.2 |
| 1.00 | 8.6 | 8.6 | 3.0 | 10.2 | 56.6 | 56.8 | 16.3 | 61.3 |

Numerical results of the proposed greedy heuristics on randomly generated test instances with \( m = n = 20 \) are shown in Table 2. The optimal value \( f^* \) is computed by an enumeration of 0-1 vectors \( x \) in Gray code representation (e.g., see Skiena, 2008). Both greedy heuristics GREEDY-1A and GREEDY-2A improve no less than 1.7 (\( \approx 5.1/3.0 \)) times the initial objective function value \( f(0) \), while they indicate the error corresponding to the weighted profits of two or three
Table 3  Objective function values by the proposed greedy heuristics on instances with unit profits and unit costs, i.e., $p_i = 1$ for all items $i \in I$ and $c_j = 1$ for all players $j \in J$

<table>
<thead>
<tr>
<th>Coefficient in $f$</th>
<th>Budget $b/\sum c_j$</th>
<th>$m = n = 20$</th>
<th>$m = n = 50$</th>
<th>$m = n = 80$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1A 2A $f(0)$</td>
<td>1A 2A $f(0)$</td>
<td>1A 2A $f(0)$</td>
</tr>
<tr>
<td>0.00</td>
<td>0.25</td>
<td>6.2 6.2 3.1</td>
<td>15.3 15.3 7.5</td>
<td>24.6 24.6 11.9</td>
</tr>
<tr>
<td>0.50 0.25</td>
<td></td>
<td>5.1 5.1 3.0</td>
<td>13.0 13.0 7.3</td>
<td>21.0 21.0 11.9</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>5.7 5.7 3.0</td>
<td>12.6 12.7 7.0</td>
<td>20.9 20.9 11.9</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>6.6 6.6 3.0</td>
<td>14.6 14.6 6.9</td>
<td>24.5 24.5 11.8</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>8.4 8.4 3.1</td>
<td>21.6 21.6 7.5</td>
<td>34.5 34.5 11.9</td>
</tr>
<tr>
<td>0.25</td>
<td></td>
<td>6.9 7.0 3.0</td>
<td>17.7 18.2 7.3</td>
<td>28.3 29.0 11.9</td>
</tr>
<tr>
<td>0.50 0.50</td>
<td></td>
<td>5.6 5.9 3.0</td>
<td>14.1 15.0 7.2</td>
<td>22.6 23.9 11.8</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>6.9 7.1 3.0</td>
<td>17.6 18.1 7.0</td>
<td>28.2 29.0 11.8</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>8.4 8.4 3.0</td>
<td>21.5 21.5 6.9</td>
<td>34.5 34.5 11.8</td>
</tr>
</tbody>
</table>

items for each test instance. From the observation, we would like to design a further improved heuristic algorithm for future research, although the proposed greedy heuristics find an optimal solution to the benchmark instance in Table 1.

Performance of the proposed greedy heuristics on instances with unit profits and unit costs is summarized in Table 3. Although there is no significant difference between GREEDY-1A and GREEDY-2A, the latter heuristic takes better objective function values than the former. It is also natural that both greedy heuristics almost take smaller objective function values for instances with a smaller budget than those for instances with a larger budget. For enough budget instances, the proposed greedy heuristics make around 40–60% of the items completely blue-signaled or completely red-signaled ones. We have the similar observations in Table 4, which shows the performance of proposed greedy heuristics on in-

Table 4  Objective function values by the proposed greedy heuristics on instances with $p_i \in \{1, 2, \ldots, 10\}$ for each item $i \in I$ and $c_j \in \{1, 2, \ldots, 10\}$ for each player $j \in J$

<table>
<thead>
<tr>
<th>Coefficient in $f$</th>
<th>Budget $b/\sum c_j$</th>
<th>$m = n = 20$</th>
<th>$m = n = 50$</th>
<th>$m = n = 80$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1A 2A $f(0)$</td>
<td>1A 2A $f(0)$</td>
<td>1A 2A $f(0)$</td>
</tr>
<tr>
<td>0.00</td>
<td>0.25</td>
<td>42.7 42.8 16.8</td>
<td>118.0 118.0 40.5</td>
<td>183.9 183.9 65.4</td>
</tr>
<tr>
<td>0.25</td>
<td></td>
<td>35.7 35.9 16.7</td>
<td>95.8 96.3 39.7</td>
<td>150.0 150.3 65.4</td>
</tr>
<tr>
<td>0.50 0.25</td>
<td></td>
<td>31.9 32.6 16.6</td>
<td>79.5 80.4 38.9</td>
<td>123.0 123.8 65.4</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>37.4 37.8 16.4</td>
<td>94.1 94.3 38.0</td>
<td>147.2 147.6 65.4</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>45.1 45.2 16.3</td>
<td>114.8 114.8 37.2</td>
<td>179.4 179.4 65.4</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>55.2 55.3 16.8</td>
<td>145.3 145.3 40.5</td>
<td>225.7 225.7 65.4</td>
</tr>
<tr>
<td>0.25</td>
<td></td>
<td>44.2 45.0 16.7</td>
<td>115.3 117.9 39.7</td>
<td>180.0 184.2 65.4</td>
</tr>
<tr>
<td>0.50 0.50</td>
<td></td>
<td>34.8 36.3 16.6</td>
<td>88.6 92.4 38.9</td>
<td>137.6 144.9 65.4</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>44.5 45.3 16.4</td>
<td>115.5 117.7 38.0</td>
<td>180.5 184.2 65.4</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>55.6 55.8 16.3</td>
<td>144.7 144.7 37.2</td>
<td>225.6 225.6 65.4</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>56.7 56.8 16.8</td>
<td>145.9 145.9 40.5</td>
<td>226.4 226.4 65.4</td>
</tr>
<tr>
<td>0.25</td>
<td></td>
<td>45.7 46.4 16.7</td>
<td>117.4 119.1 39.7</td>
<td>182.1 185.6 65.4</td>
</tr>
<tr>
<td>0.50 0.75</td>
<td></td>
<td>35.0 36.1 16.6</td>
<td>88.8 92.3 38.9</td>
<td>137.9 144.9 65.4</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>45.7 46.4 16.4</td>
<td>117.3 119.1 38.0</td>
<td>182.1 185.6 65.4</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>56.6 56.8 16.3</td>
<td>145.9 145.9 37.2</td>
<td>226.4 226.4 65.4</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>56.7 56.8 16.8</td>
<td>145.9 145.9 40.5</td>
<td>226.4 226.4 65.4</td>
</tr>
<tr>
<td>0.25</td>
<td></td>
<td>45.8 46.4 16.7</td>
<td>117.4 119.1 39.7</td>
<td>182.1 185.6 65.4</td>
</tr>
<tr>
<td>0.50 1.00</td>
<td></td>
<td>35.0 36.1 16.6</td>
<td>88.8 92.3 38.9</td>
<td>137.9 144.9 65.4</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>45.8 46.4 16.4</td>
<td>117.4 119.1 38.0</td>
<td>182.1 185.6 65.4</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>56.7 56.8 16.3</td>
<td>145.9 145.9 37.2</td>
<td>226.4 226.4 65.4</td>
</tr>
</tbody>
</table>
Table 5 Objective function values by the greedy heuristics without the initial sorting on instances with $p_i \in \{1, \ldots, 10\}$ for each item $i \in I$ and $c_j \in \{1, \ldots, 10\}$ for each player $j \in J$.

<table>
<thead>
<tr>
<th>Coefficient in $f$</th>
<th>Budget</th>
<th>$m = n = 20$</th>
<th>$m = n = 50$</th>
<th>$m = n = 80$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>$b/ \Sigma c_i$</td>
<td>$f(0)$</td>
<td>$f(0)$</td>
<td>$f(0)$</td>
</tr>
<tr>
<td>0.00</td>
<td>0.25</td>
<td>35.1</td>
<td>35.4</td>
<td>16.8</td>
</tr>
<tr>
<td>0.25</td>
<td>0.50</td>
<td>30.4</td>
<td>30.9</td>
<td>16.7</td>
</tr>
<tr>
<td>0.50</td>
<td>0.75</td>
<td>28.6</td>
<td>29.2</td>
<td>16.6</td>
</tr>
<tr>
<td>0.75</td>
<td>1.00</td>
<td>31.8</td>
<td>32.2</td>
<td>16.4</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>36.9</td>
<td>37.0</td>
<td>16.3</td>
</tr>
<tr>
<td>0.00</td>
<td>0.25</td>
<td>44.7</td>
<td>44.9</td>
<td>16.8</td>
</tr>
<tr>
<td>0.25</td>
<td>0.50</td>
<td>36.7</td>
<td>37.8</td>
<td>16.7</td>
</tr>
<tr>
<td>0.50</td>
<td>0.75</td>
<td>31.2</td>
<td>32.8</td>
<td>16.6</td>
</tr>
<tr>
<td>0.75</td>
<td>1.00</td>
<td>37.3</td>
<td>38.5</td>
<td>16.4</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>45.1</td>
<td>45.5</td>
<td>16.3</td>
</tr>
</tbody>
</table>

In order to examine the effect of the initial sorting of the items in non-increasing order of their profits, the performance of the greedy heuristics without the initial sorting on instances with different profits and different costs is shown in Table 5. That is, we compare GREEDY-1A (resp., GREEDY-2A) with GREEDY-1B (resp., with GREEDY-2B). We observe that the initial sorting of the items is clearly effective on the performance of the proposed heuristics, comparing with the results indicated in Table 4. In addition, the comparison between the raw profits and the modified profits (see Eq. (23)) with the scaling constant $\alpha = 1$ (i.e., non-scaled) in the initial sorting of the items is provided in Table 6. The initial sorting of the

Table 6 Objective function values by the greedy heuristic with the initial sorting of the modified profits ($\alpha = 1$) on instances with $m = n = 20$.

<table>
<thead>
<tr>
<th>Coefficient in $f$</th>
<th>Budget</th>
<th>$p_i = 1$ and $c_j = 1$</th>
<th>$p_i, c_j \in {1, \ldots, 10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>$b/ \Sigma c_i$</td>
<td>$2A$</td>
<td>$2C$</td>
</tr>
<tr>
<td>0.00</td>
<td>0.25</td>
<td>6.2</td>
<td>7.7</td>
</tr>
<tr>
<td>0.25</td>
<td>0.50</td>
<td>5.4</td>
<td>6.1</td>
</tr>
<tr>
<td>0.50</td>
<td>0.75</td>
<td>5.1</td>
<td>5.1</td>
</tr>
<tr>
<td>0.75</td>
<td>1.00</td>
<td>5.7</td>
<td>6.1</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>6.6</td>
<td>7.8</td>
</tr>
<tr>
<td>0.00</td>
<td>0.25</td>
<td>8.4</td>
<td>8.8</td>
</tr>
<tr>
<td>0.25</td>
<td>0.50</td>
<td>7.0</td>
<td>7.4</td>
</tr>
<tr>
<td>0.50</td>
<td>0.75</td>
<td>5.9</td>
<td>6.0</td>
</tr>
<tr>
<td>0.75</td>
<td>1.00</td>
<td>7.1</td>
<td>7.5</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>8.4</td>
<td>8.9</td>
</tr>
<tr>
<td>0.00</td>
<td>0.25</td>
<td>8.6</td>
<td>8.9</td>
</tr>
<tr>
<td>0.25</td>
<td>0.50</td>
<td>7.3</td>
<td>7.5</td>
</tr>
<tr>
<td>0.50</td>
<td>0.75</td>
<td>5.9</td>
<td>6.0</td>
</tr>
<tr>
<td>0.75</td>
<td>1.00</td>
<td>7.3</td>
<td>7.6</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>8.6</td>
<td>8.9</td>
</tr>
</tbody>
</table>
Table 7 Execution times of the proposed greedy heuristics on instances with \( p_i \in \{1, 2, \ldots, 10\} \) for each item \( i \in I \) and \( c_j \in \{1, 2, \ldots, 10\} \) for each player \( j \in J \), \( \times 10^{-2} \) [msec]

<table>
<thead>
<tr>
<th>Coefficient in ( f )</th>
<th>Budget ( b/\sum c_j )</th>
<th>( m = n = 20 )</th>
<th>( m = n = 50 )</th>
<th>( m = n = 80 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td></td>
<td>1A</td>
<td>2A</td>
<td>1A</td>
</tr>
<tr>
<td>0.00</td>
<td>1.9</td>
<td>9.2</td>
<td>13.2</td>
<td>19.7</td>
</tr>
<tr>
<td>0.25</td>
<td>2.0</td>
<td>8.9</td>
<td>13.0</td>
<td>18.9</td>
</tr>
<tr>
<td>0.50</td>
<td>2.2</td>
<td>9.0</td>
<td>13.2</td>
<td>19.7</td>
</tr>
<tr>
<td>0.75</td>
<td>1.9</td>
<td>8.7</td>
<td>13.1</td>
<td>19.0</td>
</tr>
<tr>
<td>1.00</td>
<td>2.0</td>
<td>8.6</td>
<td>12.9</td>
<td>19.2</td>
</tr>
<tr>
<td>0.00</td>
<td>1.9</td>
<td>8.2</td>
<td>11.9</td>
<td>17.9</td>
</tr>
<tr>
<td>0.25</td>
<td>1.7</td>
<td>8.3</td>
<td>11.7</td>
<td>17.9</td>
</tr>
<tr>
<td>0.50</td>
<td>1.9</td>
<td>7.9</td>
<td>11.6</td>
<td>17.8</td>
</tr>
<tr>
<td>0.75</td>
<td>1.9</td>
<td>8.1</td>
<td>11.7</td>
<td>17.7</td>
</tr>
<tr>
<td>1.00</td>
<td>1.6</td>
<td>8.1</td>
<td>11.7</td>
<td>17.7</td>
</tr>
<tr>
<td>0.00</td>
<td>2.0</td>
<td>9.9</td>
<td>13.3</td>
<td>21.7</td>
</tr>
<tr>
<td>0.25</td>
<td>2.0</td>
<td>9.8</td>
<td>12.9</td>
<td>21.1</td>
</tr>
<tr>
<td>0.50</td>
<td>1.9</td>
<td>9.3</td>
<td>13.0</td>
<td>21.3</td>
</tr>
<tr>
<td>0.75</td>
<td>2.2</td>
<td>10.0</td>
<td>12.9</td>
<td>21.5</td>
</tr>
<tr>
<td>1.00</td>
<td>2.0</td>
<td>9.9</td>
<td>12.9</td>
<td>21.9</td>
</tr>
<tr>
<td>0.00</td>
<td>2.2</td>
<td>9.9</td>
<td>13.3</td>
<td>21.6</td>
</tr>
<tr>
<td>0.25</td>
<td>2.0</td>
<td>9.9</td>
<td>13.2</td>
<td>21.6</td>
</tr>
<tr>
<td>0.50</td>
<td>2.2</td>
<td>10.0</td>
<td>13.0</td>
<td>21.5</td>
</tr>
<tr>
<td>0.75</td>
<td>2.2</td>
<td>9.7</td>
<td>13.1</td>
<td>22.0</td>
</tr>
<tr>
<td>1.00</td>
<td>2.0</td>
<td>9.9</td>
<td>13.1</td>
<td>22.1</td>
</tr>
</tbody>
</table>

items with the modified profits in GREEDY-2A, i.e., algorithm GREEDY-2C, illustrates a small improvement on the test instances.

The execution times of greedy heuristics GREEDY-1A and GREEDY-2A are shown in Table 7. As evaluated in the previous section, the time complexity of each greedy heuristic is \( O(m \log m + mn) \). We observe that the actual execution time almost follows the theoretical evaluation.

6. Concluding Remarks

In this paper, we considered a variant of 0-1 knapsack problems in graph structures, which we call problem CIC (cooperative item collecting). Given a directed bipartite structure with a set of items and a set of players, the problem asks to find an arc reversing strategy of the players which collects items in their cooperation with a budget constraint so that the total weighted profit of the collected items is maximized. We first addressed some mathematical properties of the problem such as counterpart vectors, basic semi-solutions and the consistency of distinct semi-solutions. Then, utilizing the compounding procedure of distinct semi-solutions with budget checking, we proposed two greedy heuristic algorithms, both of which run in polynomial time. We also conducted numerical experiments on randomly generated instances to demonstrate the empirical performance of the greedy heuristic algorithms. The results showed that the proposed greedy heuristics ran within 1 [msec] for the test instances with up to \( m = 80 \) and \( n = 80 \) on a laptop personal computer, where \( m \) and \( n \) denote the number of profitable items and the number of players, respectively, and returns heuristic solutions getting around 40–60 [%] of the \( m \) items. In addition, the proposed greedy heuristic algorithms found an optimal solution of the test instance of the minimal switching graph problem as a constrained via minimization model provided by Tang (2005) in short execution times.

It is well known that efficient enumerative algorithms based on the branch-and-bound technique can compute an optimal solution for a large size instance of the traditional 0-1 knapsack problem (e.g., see Ibaraki, 1989). For future research, it would be significant to design an efficient exact algorithm for problem CIC as well. It would also be important to analyze the approximability/inapproximability of problem CIC (e.g., see Vazirani, 2001). Of course, it would be interesting to apply a meta-heuristic technique to the problem for further improvement.

Appendix: A Proof of Theorem 1

In this extra section, we show that problem CIC with a non-enough budget is NP-hard. The NP-hardness is proved by a polynomial time reduction from the following NP-complete problem (e.g., see Garey and Johnson, 1979):
KNAPSACK

Instance: A set \( I = \{ i \mid i = 1, 2, \ldots, m \} \) of \( m \) items, a positive integer weight \( w_i \) and a positive integer profit \( p_i \) for each item \( i \in I \), a positive integer capacity \( b \) of the single knapsack, and a positive integer \( K \).

Question: Is there a subset \( I' \subseteq I \) of items such that it meets \( \sum_{i \in I'} w_i \leq b \) and \( \sum_{i \in I'} p_i \geq K \)?

Without loss of generality, it is assumed that \( \max_{i \in I} w_i \leq b < \sum_{i \in I} w_i \), and that \( K > \max_{i \in I} \{ p_i \} \). Note that an instance of problem KNAPSACK with \( b = \sum_{i \in I} w_i \) can be solved in polynomial time.

In the following, we transform an instance of problem KNAPSACK to a certain instance of problem CIC with \( \lambda = 0.0 \) (i.e., a completely blue-signaled item is desired). Then, we show that the transformed instance of problem CIC has a feasible solution \( x \) with the total weighted profit \( f(x) = \sum_{i \in I'} w_i x_i \geq K \) (see Eq. (8)) if and only if the original instance of problem KNAPSACK has a solution \( I' \) such that \( \sum_{i \in I'} w_i \leq b \) and \( \sum_{i \in I'} p_i \geq K \).

We use the item set \( I \) of problem KNAPSACK as an item set in the transformed instance of problem CIC. Let \( J = J_1 \cup J_2 \) denote a set of players, where \( J_1 = \{ j \mid j = 1, 2, \ldots, m \} \) and \( J_2 = \{ m + j \mid j = 1, 2, \ldots, m \} \), and hence the number of players is set by \( n = 2m \). We also use the profit \( p_i \) for each item \( i \in I \) of problem KNAPSACK in the transformed instance of problem CIC. For each player \( j \in J_1 \), the cost is set by \( c_j := w_j \), while for each player \( j \in J_2 \), the cost is set by \( c_j := b \). Further, the knapsack capacity \( b \) is used also as the budget imposed on the \( n = 2m \) players. The directed bipartite graph with the set of \( m \) items and the set of \( n = 2m \) players is constructed in Fig. 2. That is, the initial arc sets are defined by \( A_B(0) := \{ (i, j) \mid i \in I, j \in J_2, i = m + j \} \) and \( A_B(0) := \{ (j, i) \mid j \in J_1, i \in I, j = m + i \} \). It is obvious that such an instance of problem CIC is transformed from any instance of problem KNAPSACK in polynomial time.

Suppose that the original instance of problem KNAPSACK has a solution \( I' \). Then, consider the following reversing vector \( x = (x_1, x_2, \ldots, x_{2m}) \):

\[
x_j = \begin{cases} 1 & \text{if } j \in I' \subseteq \{1, 2, \ldots, m\} = J_1, \\ 0 & \text{otherwise}. \end{cases}
\]

(24)

Note that in the \( x \), no player \( j \in J_2 \) is reversed, and the reversing cost meets

\[
c(x) = \sum_{j=1}^{2m} c_j x_j = \sum_{j=1}^{m} c_j x_j = \sum_{j \in J_1} c_j = \sum_{i \in I'} w_i \leq b,
\]

and hence the \( x \) is a feasible solution. Initially, we see \( I_B(0) = \emptyset \) (i.e., there is no completely blue-signaled item), while an item \( i \in I' \) becomes a completely blue-signed one since the player \( j \in I' \) is reversed in the \( x \), and \( I_B(x) = I' \subseteq I \). Then, we have

\[
f(x) = f_B(x) = \sum_{i \in I'} p_i \geq K.
\]

(25)

Hence, if the answer to the original instance of problem KNAPSACK is yes, the transformed instance of problem CIC has a reversing vector \( x \) such that \( c(x) \leq b \) and \( f(x) \geq K \).

Fig. 2 Directed bipartite structure of an instance of problem CIC transformed from problem KNAPSACK
Suppose that for the transformed instance of problem CIC, there exists a reversing vector \( x = (x_1, x_2, \ldots, x_n) \) such that \( c(x) \leq b \) and \( f(x) = f_b(x) \geq K \). If \( x_j = 1 \) holds for some \( j \in J_2 \) in the \( x \), then \( x_k = 0 \) must hold for any \( k \in J \setminus \{j\} \) since \( c_j = b \). However, in such a reversing vector \( x \), no item \( i \in I \) can be a completely blue-signaled item (even the item \( j - m \in I \) is a completely red-signaled one), and \( f(x) = 0 < K \), which contradicts the assumption. Hence, the \( x \) reverses players only in \( J_1 \). Let \( J' \subseteq J_1 = \{1, 2, \ldots, m\} \) denote the set of reversed players in the \( x \). Then, it satisfies \( c(x) = \sum_{j \in J'} c_j = \sum_{i \in J'} w_i \leq b \) and \( f(x) = \sum_{i \in J'} p_i \geq K \), which implies that the original instance of problem KNAPSACK has a yes solution \( I' = J' \).

\[\square\]

**Acknowledgment**

The authors are grateful to two anonymous referees for their comments which were truly helpful to improve the first version of this paper. This research was partially supported by JSPS KAKENHI Grant JP16K01241.

**References**


