Flaws of Termination and Optimality in ADOPT-based Algorithms (Appendices)

Koji Noshiro and **Koji Hasebe**
University of Tsukuba
noshiro@mas.cs.tsukuba.ac.jp, hasebe@cs.tsukuba.ac.jp

A Overview of ADOPT

ADOPT [Modi *et al.*, 2005] is known as an asynchronous complete algorithm to solve DCOPs. Since the messages in ADOPT are exchanged based on a DFS pseudo-tree, ADOPT requires constructing it in the preprocessing.

The message framework of ADOPT consists of four messages, namely VALUE, COST, THRESHOLD, and TERMINATE. A VALUE message reports the current value d_i of agent x_i to its children and pseudo-children x_c. When agent x_i receives a VALUE message from its parent or pseudo-parent x_p, x_i records the value in the message to the current context CX_i. A context of agent x_i is the assignments of variables higher than x_i in the DFS pseudo-tree.

A COST message is sent from agent x_i to its parent x_p and reports the current context CX_i and the lower and upper bounds of the cost for the subtree rooted at x_i, denoted as LB_i and UB_i, respectively. For the definition of LB_i and UB_i, we define local cost $\delta_i(d, CX)$ for value d and context CX of x_i. This is the sum of cost functions between x_i and its higher neighbors (i.e., its parent and pseudo-parents), given value d and context CX. Local cost $\delta_i(d, CX)$ is calculated as follows:

$$\delta_i(d, CX) := \sum_{(x_j, d_j) \in CX} f_{i,j}(d, d_j).$$

$LB_i(d)$ and $UB_i(d)$ are the lower and upper bounds for value d of agent x_i, respectively, and are defined as follows:

$$LB_i(d) := \delta_i(d, CX_i) + \sum_{x_c \in C(x_i)} lb_i(d, x_c) \tag{3}$$

$$UB_i(d) := \delta_i(d, CX_i) + \sum_{x_c \in C(x_i)} ub_i(d, x_c), \tag{4}$$

where $C(x_i)$ is the children of x_i and $lb_i(d, x_c)$ and $ub_i(d, x_c)$ are the lower and upper bounds received from a child x_c through a COST message whose context contains assignment (x_i, d). In initialization, lower bounds $lb_i(d, x_c)$ and upper bounds $ub_i(d, x_c)$ are set as $lb_i(d, x_c) := 0$ and $ub_i(d, x_c) := \infty$. LB_i and UB_i are the lower and upper bounds, respectively, and are defined as follows:

$$LB_i := \min_{d \in D_i} LB_i(d) \tag{5}$$

$$UB_i := \min_{d \in D_i} UB_i(d). \tag{6}$$

When agent x_i receives a COST message from a child x_c, and the context in the message contains assignment (x_i, d) and is compatible with the current context CX_i, x_i stores the received context, lower bound, and upper bound in $cx_i(d, x_c)$, $lb_i(d, x_c)$, and $ub_i(d, x_c)$, respectively. Here, two contexts are compatible if their assignments of any variables do not take different values. Additionally, x_i also stores the assignment (x_p, d_p) contained by the context in a COST message if x_p is not the neighbor of x_i.

If the current context CX_i becomes incompatible with context $cx_i(d, x_c)$ for value d and a child x_c by receiving VALUE or COST messages, x_i reinitializes the context $cx_i(d, x_c)$ to be empty, the lower bound $lb_i(d, x_c)$ to 0, and the upper bound $ub_i(d, x_c)$ to ∞. This reinitialization means that x_i stores lower and upper bounds only for the current context CX_i, which leads to efficient space complexity. However, reinitialization can cause an agent to recompute the lower and upper bounds for a previous context from scratch. Additionally, an agent may change the variable value multiple times due to the best-first search in ADOPT.

In order to avoid changing the value many times, ADOPT introduces a threshold that stores cost information received from children. Agent x_i maintains threshold TH_i and allocates it to the children. $th_i(d, x_c)$ represents the threshold allocated to a child x_c when x_i takes value d. The thresholds TH_i and $th_i(d, x_c)$ of x_i are updated to maintain the following three invariants:

Definition 2 (ThresholdInvariant)**.**

$$LB_i \leq TH_i \leq UB_i.$$

Definition 3 (AllocationInvariant)**.**

$$TH_i = \delta_i(d_i, CX_i) + \sum_{x_c \in C(x_i)} th_i(d_i, x_c),$$

where $d_i \in D$ is the current value of x_i.

Definition 4 (ChildThresholdInvariant)**.** $\forall d \in D_i, \forall x_c \in C(x_i)$,

$$lb_i(d, x_c) \leq th_i(d, x_c) \leq ub_i(d, x_c).$$

TH_i and $th_i(d, x_c)$ are initialized to 0. Furthermore, $th_i(d, x_c)$ is reinitialized as well as $lb_i(d, x_c)$ and $ub_i(d, x_c)$ when the current context CX_i becomes incompatible with context $cx_i(d, x_c)$. A THRESHOLD message is sent from

agent x_i to its children x_c and reports the current context CX_i and the threshold $th_i(d_i, x_c)$ allocated to x_c for the current value d_i of x_i. When agent x_i receives a THRESHOLD message from its parent x_p, x_i stores the received threshold in TH_i if the context in the message is compatible with the current context CX_i.

The threshold TH_i is used for the conditions for changing the variable value and termination. If the lower bound for the current value d_i, represented by $LB_i(d_i)$, becomes greater than the threshold TH_i, agent x_i changes the value to $d \in D_i$ that minimizes the lower bound $LB_i(d)$. The termination condition is that the threshold TH_i is equal to the upper bound UB_i. If the root agent x_r in the DFS pseudotree satisfies this condition, x_r changes the value to d that minimizes $UB_i(d)$ and sends TERMINATE messages to its children, and then x_r terminates. If non-root agent x_i receives a TERMINATE message from its parent x_p and satisfies the termination condition, x_i also changes the value, sends TERMINATE messages, and executes termination.

The original study of ADOPT provides three properties (as theorems in the paper) in terms of its termination and optimality. Before showing the properties, we define the optimal costs for the subtree rooted at agent x_i given a context. These costs are named gamma costs in [Yeoh *et al.*, 2010], which correspond to $OPT(x_i, context)$ in the original study of ADOPT. The gamma costs $\gamma_i(CX)$ and $\gamma_i(d, CX)$ are defined recursively as follows:

$$\gamma_i(d, CX) := \delta_i(d, CX) + \sum_{x_c \in C(x_i)} \gamma_i(CX \cup \{(x_i, d)\})$$

$$\gamma_i(CX) := \min_{d \in D_i} \gamma_i(d, CX),$$

where d represents a value of x_i and CX denotes a context of x_i. Three properties shown below are given in the study of ADOPT. Property 1 shows the relations between the bounds and the gamma cost; Property 2 implies guaranteeing the termination; and Property 3 implies guaranteeing the optimality.

Property 1. $\forall x_i \in X, LB_i \leq \gamma_i(CX_i) \leq UB_i$.

Property 2. $\forall x_i \in X$, *if the current context CX_i is fixed, then $TH_i = UB_i$ will eventually occur.*

Property 3. $\forall x_i \in X$, *x_i's final threshold value TH_i is equal to $\gamma_i(CX_i)$.*

B Traces of Counterexamples

In this section, we show the detailed traces of the counterexamples in Section 3. The traces are composed of figures that show sequences of agent actions and tables that show states of agents. In a figure, shaded agents (i.e., nodes) receive messages, process them, and send new messages. Received messages are represented by arrows whose heads are painted in black, while sent messages are represented by arrows whose heads are not painted. We omit some messages, e.g., most THRESHOLD messages, that are not crucial. Additionally, tables show the states of agents when they complete processing the received messages, and some trivial states, e.g., the current context of a root agent, are omitted.

B.1 Counterexample to Termination

Figures 7–33 and Tables 2–12 show the trace of the counterexample to termination, described in 3.1.

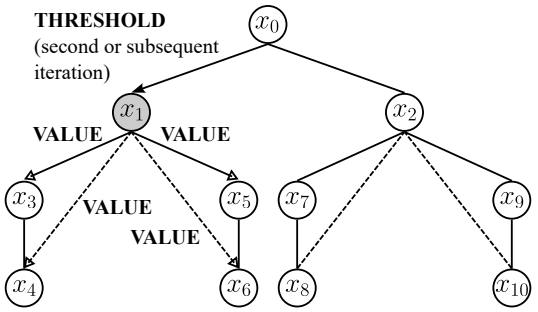

Figure 7: Step 1 (1). x_1 sends VALUE messages to its children and pseudo-children x_3, x_4, x_5, and x_6.

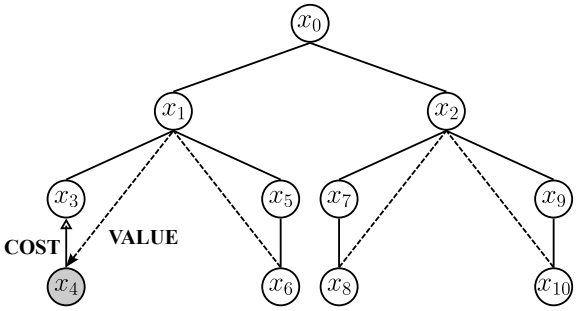

Figure 8: Step 1 (2). x_4 receives the VALUE message and computes the lower bound using the updated context $CX_4 \ni (x_1, 0)$, and thus $LB_4 = 5$. Then x_4 reports the lower bound to x_3 by sending a COST message.

Variables	Step 0	Step 1 (1)	Step 1 (2)
x_0			
d_0	0	0	0
LB_0	0	0	0
$lb_0(0, x_1)$	0	0	0
$lb_0(0, x_2)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(0, x_2)$	∞	∞	∞
TH_0	0	0	0
$th_0(0, x_1)$	0	0	0
$th_0(0, x_2)$	0	0	0
x_1			
d_1	0	0	0
LB_1	0	0	0
$lb_1(0, x_3)$	0	0	0
$lb_1(0, x_5)$	0	0	0
$lb_1(1, x_3)$	0	0	0
$lb_1(1, x_5)$	0	0	0
UB_1	∞	∞	∞
$ub_1(0, x_3)$	∞	∞	∞
$ub_1(0, x_5)$	∞	∞	∞
$ub_1(1, x_3)$	∞	∞	∞
$ub_1(1, x_5)$	∞	∞	∞
TH_1	0	0	0
x_3			
d_3	0	0	0
CX_3	{}	{}	{}
$cx_3(0, x_4)$	{}	{}	{}
LB_3	0	0	0
$lb_3(0, x_4)$	0	0	0
UB_3	∞	∞	∞
$ub_3(0, x_4)$	∞	∞	∞
x_4			
d_4	0	0	0
CX_4	{}	{}	$\boldsymbol{\{(x_1, 0), (x_3, 0)\}}$
LB_4	0	0	**5**
UB_4	∞	∞	**5**
x_5			
d_5	0	0	0
CX_5	{}	{}	{}
$cx_5(0, x_6)$	{}	{}	{}
LB_5	0	0	0
$lb_5(0, x_6)$	0	0	0
UB_5	∞	∞	∞
$ub_5(0, x_6)$	∞	∞	∞
x_6			
d_6	0	0	0
CX_6	{}	{}	{}
LB_6	0	0	0
UB_6	∞	∞	∞

Table 2: States of agents in Steps 0–1 (2).

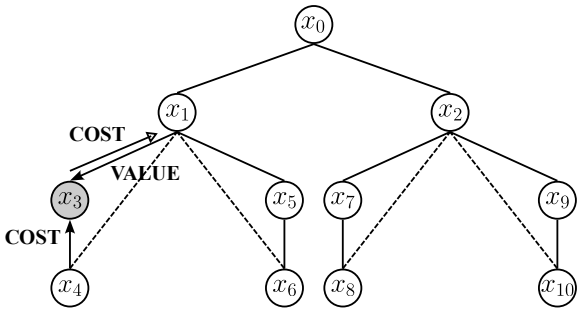

Figure 9: Step 1 (3). x_3 receives the VALUE message from x_1 and the COST message from x_4 and computes the lower bound as $LB_3 = 5$. Then x_3 sends a COST message with $CX_3 \ni (x_1, 0)$ and $LB_3 = 5$ to x_1.

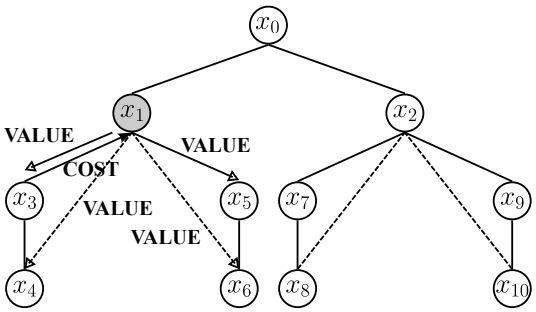

Figure 10: Step 1 (4). After x_1 receives the COST message, x_1 updates $LB_1(0)$ to 5. Since $LB_1(0) = 5 > TH_1 = 0$, x_1 changes its value d_1 to 1 and sends VALUE messages to its children and pseudo-children.

Variables	Step 1 (3)	Step 1 (4)
x_0		
d_0	0	0
LB_0	0	0
$lb_0(0, x_1)$	0	0
$lb_0(0, x_2)$	0	0
UB_0	∞	∞
$ub_0(0, x_1)$	∞	∞
$ub_0(0, x_2)$	∞	∞
TH_0	0	0
$th_0(0, x_1)$	0	0
$th_0(0, x_2)$	0	0
x_1		
d_1	0	**1**
LB_1	0	0
$lb_1(0, x_3)$	0	**5**
$lb_1(0, x_5)$	0	0
$lb_1(1, x_3)$	0	0
$lb_1(1, x_5)$	0	0
UB_1	∞	∞
$ub_1(0, x_3)$	∞	**5**
$ub_1(0, x_5)$	∞	∞
$ub_1(1, x_3)$	∞	∞
$ub_1(1, x_5)$	∞	∞
TH_1	0	0
x_3		
d_3	0	0
CX_3	$\boldsymbol{\{(x_1, 0)\}}$	$\{(x_1, 0)\}$
$cx_3(0, x_4)$	$\boldsymbol{\{(x_1, 0)\}}$	$\{(x_1, 0)\}$
LB_3	**5**	5
$lb_3(0, x_4)$	**5**	5
UB_3	**5**	5
$ub_3(0, x_4)$	**5**	5
x_4		
d_4	0	0
CX_4	$\{(x_1, 0),$ $(x_3, 0)\}$	$\{(x_1, 0),$ $(x_3, 0)\}$
LB_4	5	5
UB_4	5	5
x_5		
d_5	0	0
CX_5	$\{\}$	$\{\}$
$cx_5(0, x_6)$	$\{\}$	$\{\}$
LB_5	0	0
$lb_5(0, x_6)$	0	0
UB_5	∞	∞
$ub_5(0, x_6)$	∞	∞
x_6		
d_6	0	0
CX_6	$\{\}$	$\{\}$
LB_6	0	0
UB_6	∞	∞

Table 3: States of agents in Steps 1 (3)–1 (4).

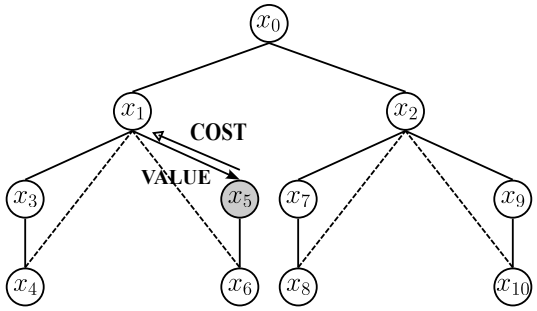

Figure 11: Step 2 (1). x_5 receives the VALUE messages from x_1 and sends a COST message to x_1, but this COST message does not affect the bounds of x_1 because $LB_5 = 0$ and $UB_5 = \infty$, which are the initial bounds.

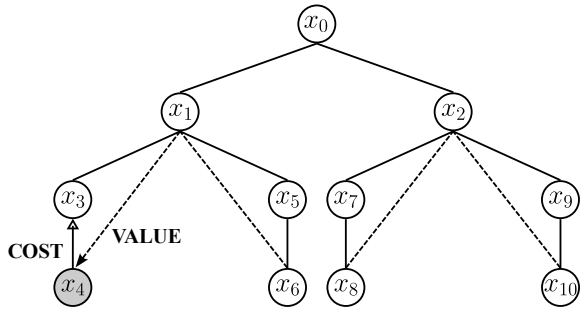

Figure 12: Step 2 (2). Similar to the procedure in Step 1, x_4 receives and sends messages.

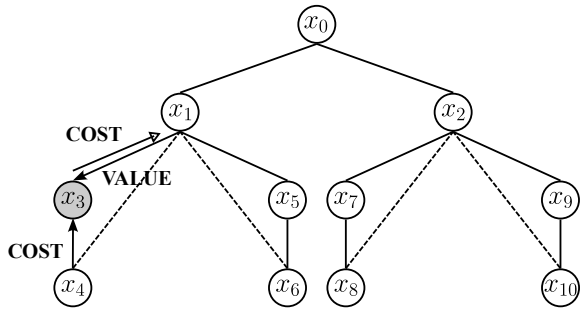

Figure 13: Step 2 (3). Similar to the procedure in Step 1, x_3 receives and sends messages.

Variables	Step 2 (1)	Step 2 (2)	Step 2 (3)
x_0			
d_0	0	0	0
LB_0	0	0	0
$lb_0(0, x_1)$	0	0	0
$lb_0(0, x_2)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(0, x_2)$	∞	∞	∞
TH_0	0	0	0
$th_0(0, x_1)$	0	0	0
$th_0(0, x_2)$	0	0	0
x_1			
d_1	1	1	1
LB_1	0	0	0
$lb_1(0, x_3)$	5	5	5
$lb_1(0, x_5)$	0	0	0
$lb_1(1, x_3)$	0	0	0
$lb_1(1, x_5)$	0	0	0
UB_1	∞	∞	∞
$ub_1(0, x_3)$	5	5	5
$ub_1(0, x_5)$	∞	∞	∞
$ub_1(1, x_3)$	∞	∞	∞
$ub_1(1, x_5)$	∞	∞	∞
TH_1	0	0	0
x_3			
d_3	0	0	0
CX_3	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$	$\mathbf{\{(x_1, 1)\}}$
$cx_3(0, x_4)$	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$	$\mathbf{\{(x_1, 1)\}}$
UB_3	5	5	**6**
$lb_3(0, x_4)$	5	5	**6**
UB_3	5	5	**6**
$ub_3(0, x_4)$	5	5	**6**
x_4			
d_4	0	0	0
CX_4	$\{(x_1, 0),$ $(x_3, 0)\}$	$\mathbf{\{(x_1, 1),}$ $\mathbf{(x_3, 0)\}}$	$\{(x_1, 1),$ $(x_3, 0)\}$
LB_4	5	**6**	6
UB_4	5	**6**	6
x_5			
d_5	0	0	0
CX_5	$\{\}$	$\{\}$	$\{\}$
$cx_5(0, x_6)$	$\{\}$	$\{\}$	$\{\}$
LB_5	0	0	0
$lb_5(0, x_6)$	0	0	0
UB_5	∞	∞	∞
$ub_5(0, x_6)$	∞	∞	∞
x_6			
d_6	0	0	0
CX_6	$\{\}$	$\{\}$	$\{\}$
LB_6	0	0	0
UB_6	∞	∞	∞

Table 4: States of agents in Steps 2 (1)–2 (3).

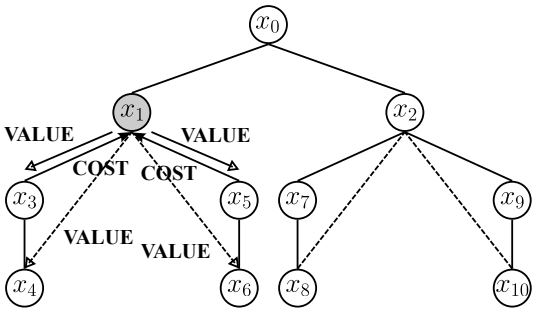

Figure 14: Step 2 (4). Similar to the procedure in Step 1, x_1 receives and sends messages, and then x_1 updates the lower bound as $LB_1(1) = 6$. Additionally, the threshold of x_1 increases as $TH_1 = LB_1 = \min\{LB_1(0), LB_1(1)\} = 5$ because of ThresholdInvariant. Since $LB_1(1) = 6 > TH_1 = 5$, x_1 changes its value d_1 back to 0.

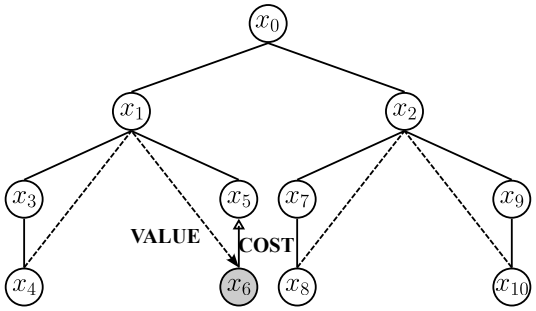

Figure 15: Step 3 (1). Similar cost calculations and value changes are performed in x_6.

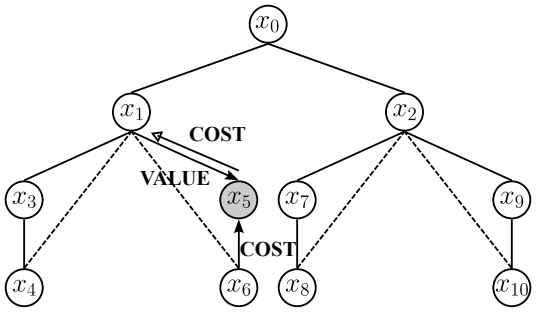

Figure 16: Step 3 (2). Similar cost calculations and value changes are performed in x_5.

Variables	Step 2 (4)	Step 3 (1)	Step 3 (2)
x_0			
d_0	0	0	0
LB_0	0	0	0
$lb_0(0, x_1)$	0	0	0
$lb_0(0, x_2)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(0, x_2)$	∞	∞	∞
TH_0	0	0	0
$th_0(0, x_1)$	0	0	0
$th_0(0, x_2)$	0	0	0
x_1			
d_1	**0**	0	0
LB_1	**5**	5	5
$lb_1(0, x_3)$	5	5	5
$lb_1(0, x_5)$	0	0	0
$lb_1(1, x_3)$	**6**	6	6
$lb_1(1, x_5)$	0	0	0
UB_1	∞	∞	∞
$ub_1(0, x_3)$	5	5	5
$ub_1(0, x_5)$	∞	∞	∞
$ub_1(1, x_3)$	**6**	6	6
$ub_1(1, x_5)$	∞	∞	∞
TH_1	**5**	5	5
x_3			
d_3	0	0	0
CX_3	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$
$cx_3(0, x_4)$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$
UB_3	6	6	6
$lb_3(0, x_4)$	6	6	6
UB_3	6	6	6
$ub_3(0, x_4)$	6	6	6
x_4			
d_4	0	0	0
CX_4	$\{(x_1, 1), (x_3, 0)\}$	$\{(x_1, 1), (x_3, 0)\}$	$\{(x_1, 1), (x_3, 0)\}$
LB_4	6	6	6
UB_4	6	6	6
x_5			
d_5	0	0	0
CX_5	$\{\}$	$\{\}$	$\boldsymbol{\{(x_1, 0)\}}$
$cx_5(0, x_6)$	$\{\}$	$\{\}$	$\boldsymbol{\{(x_1, 0)\}}$
LB_5	0	0	**2**
$lb_5(0, x_6)$	0	0	**2**
UB_5	∞	∞	**2**
$ub_5(0, x_6)$	∞	∞	**2**
x_6			
d_6	0	0	0
CX_6	$\{\}$	$\boldsymbol{\{(x_1, 0), (x_5, 0)\}}$	$\{(x_1, 0), (x_5, 0)\}$
LB_6	0	**2**	2
UB_6	∞	**2**	2

Table 5: States of agents in Steps 2 (4)–3 (2).

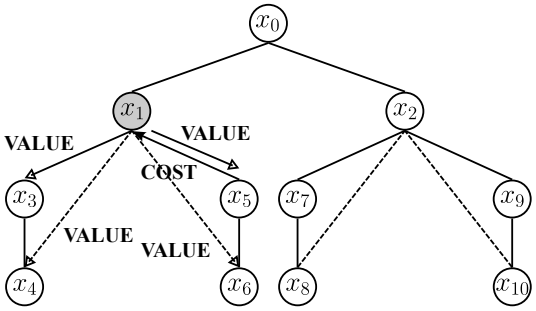

Figure 17: Step 3 (3). Similar cost calculations and value changes are performed in x_1.

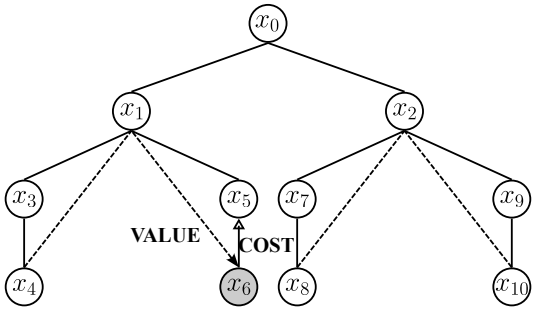

Figure 18: Step 3 (4). Similar cost calculations and value changes are performed in x_6.

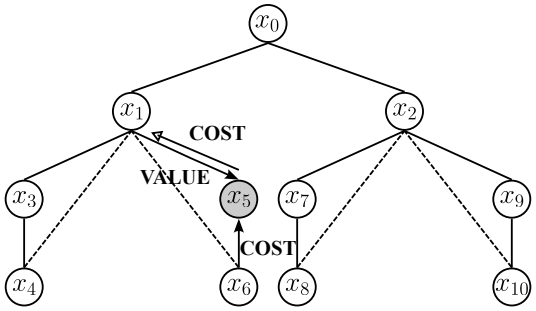

Figure 19: Step 3 (5). Similar cost calculations and value changes are performed in x_5.

Variables	Step 3 (3)	Step 3 (4)	Step 3 (5)
x_0			
d_0	0	0	0
LB_0	0	0	0
$lb_0(0, x_1)$	0	0	0
$lb_0(0, x_2)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(0, x_2)$	∞	∞	∞
TH_0	0	0	0
$th_0(0, x_1)$	0	0	0
$th_0(0, x_2)$	0	0	0
x_1			
d_1	**1**	1	1
LB_1	**6**	6	6
$lb_1(0, x_3)$	5	5	5
$lb_1(0, x_5)$	**2**	2	2
$lb_1(1, x_3)$	6	6	6
$lb_1(1, x_5)$	0	0	0
UB_1	**7**	7	7
$ub_1(0, x_3)$	5	5	5
$ub_1(0, x_5)$	**2**	2	2
$ub_1(1, x_3)$	6	6	6
$ub_1(1, x_5)$	∞	∞	∞
TH_1	**6**	6	6
x_3			
d_3	0	0	0
CX_3	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$
$cx_3(0, x_4)$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$
LB_3	6	6	6
$lb_3(0, x_4)$	6	6	6
UB_3	6	6	6
$ub_3(0, x_4)$	6	6	6
x_4			
d_4	0	0	0
CX_4	$\{(x_1, 1),$ $(x_3, 0)\}$	$\{(x_1, 1),$ $(x_3, 0)\}$	$\{(x_1, 1),$ $(x_3, 0)\}$
LB_4	6	6	6
UB_4	6	6	6
x_5			
d_5	0	0	0
CX_5	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$	$\boldsymbol{\{(x_1, 1)\}}$
$cx_5(0, x_6)$	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$	$\boldsymbol{\{(x_1, 1)\}}$
LB_5	2	2	**3**
$lb_5(0, x_6)$	2	2	**3**
UB_5	2	2	**3**
$ub_5(0, x_6)$	2	2	**3**
x_6			
d_6	0	0	0
CX_6	$\{(x_1, 0),$ $(x_5, 0)\}$	$\boldsymbol{\{(x_1, 1),}$ $\boldsymbol{(x_5, 0)\}}$	$\{(x_1, 1),$ $(x_5, 0)\}$
LB_6	2	**3**	3
UB_6	2	**3**	3

Table 6: States of agents in Steps 3 (3)–3 (5).

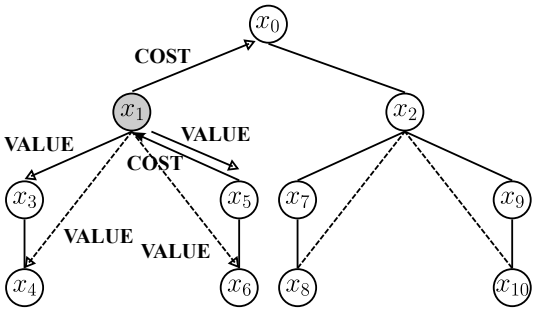

Figure 20: Step 3 (6). Similar cost calculations and value changes are performed in x_1. After the cost calculation, x_1 sends VALUE messages with the current value $d_1 = 0$ to the lower neighbors and a COST message with $LB_1 = 7$ to x_0.

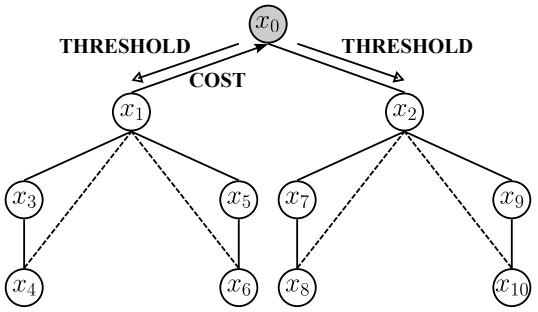

Figure 21: Step 3 (7). x_0 receives the COST message and then increases LB_0, TH_0, and $th_0(0, x_1)$ to 7.

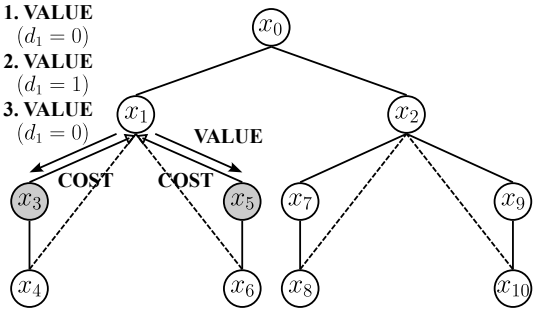

Figure 22: Step 4. x_3 receives the three VALUE messages from x_1, in which the values are $d_1 = 0, d_1 = 1$, and $d_1 = 0$, in the order of sending. When x_3 processes the first VALUE message, the lower bound of x_3 is reinitialized as $LB_3 = LB_3(0) = lb_3(0, x_4) = 0$ since context $cx_3(0, x_4) \ni (x_1, 1)$, received from x_4 through the COST message, is incompatible with the updated context $CX_3 \ni (x_1, 0)$. After x_3 processes the remaining messages, x_3 sends two types of COST messages to x_1, i.e., the message with $LB_3 = 0$ and $CX_3 = \{(x_1, 0)\}$ and the message with $LB_3 = 0$ and $CX_3 = \{(x_1, 1)\}$. Similarly, x_5 receives the VALUE message with $d_1 = 0$ from x_1 and reinitializes the bounds. Additionally, x_5 sends a COST message with $LB_5 = 0$ and $CX_5 = \{(x_1, 0)\}$ to x_1.

Variables	Step 3 (6)	Step 3 (7)	Step 4
x_0			
d_0	0	0	0
LB_0	0	**7**	7
$lb_0(0, x_1)$	0	**7**	7
$lb_0(0, x_2)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	**7**	7
$ub_0(0, x_2)$	∞	∞	∞
TH_0	0	**7**	7
$th_0(0, x_1)$	0	**7**	7
$th_0(0, x_2)$	0	0	0
x_1			
d_1	**0**	0	0
LB_1	**7**	7	7
$lb_1(0, x_3)$	5	5	5
$lb_1(0, x_5)$	2	2	2
$lb_1(1, x_3)$	6	6	6
$lb_1(1, x_5)$	**3**	3	3
UB_1	7	7	7
$ub_1(0, x_3)$	5	5	5
$ub_1(0, x_5)$	2	2	2
$ub_1(1, x_3)$	6	6	6
$ub_1(1, x_5)$	**3**	3	3
TH_1	**7**	7	7
x_3			
d_3	0	0	0
CX_3	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\boldsymbol{\{(x_1, 0)\}}$
$cx_3(0, x_4)$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\boldsymbol{\{\}}$
LB_3	6	6	**0**
$lb_3(0, x_4)$	6	6	**0**
UB_3	6	6	∞
$ub_3(0, x_4)$	6	6	∞
x_4			
d_4	0	0	0
CX_4	$\{(x_1, 1), (x_3, 0)\}$	$\{(x_1, 1), (x_3, 0)\}$	$\{(x_1, 1), (x_3, 0)\}$
LB_4	6	6	6
UB_4	6	6	6
x_5			
d_5	0	0	0
CX_5	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\boldsymbol{\{(x_1, 0)\}}$
$cx_5(0, x_6)$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\boldsymbol{\{\}}$
LB_5	3	3	**0**
$lb_5(0, x_6)$	3	3	**0**
UB_5	3	3	∞
$ub_5(0, x_6)$	3	3	∞
x_6			
d_6	0	0	0
CX_6	$\{(x_1, 1), (x_5, 0)\}$	$\{(x_1, 1), (x_5, 0)\}$	$\{(x_1, 1), (x_5, 0)\}$
LB_6	3	3	3
UB_6	3	3	3

Table 7: States of agents in Steps 3 (6)–4.

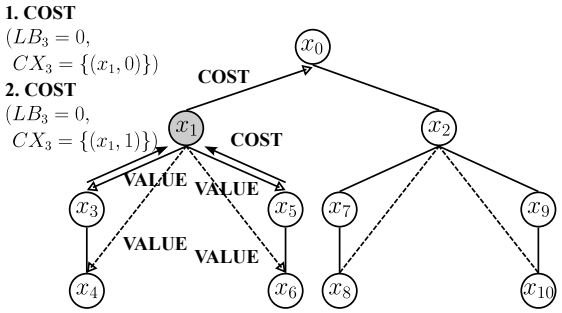

Figure 23: Step 5 (1). x_1 receives the COST messages from x_3 and x_5 and updates the lower bounds as $LB_1(0) = lb_1(0, x_3) + lb_1(0, x_5) = 0$, $LB_1(1) = lb_1(1, x_3) + lb_1(1, x_5) = 3$. Thus, x_1 obtains the lower bound as $LB_1 = 0$ and keeps the value $d_1 = 0$. Additionally, x_1 sends a COST message with $LB_1 = 0$ to x_0.

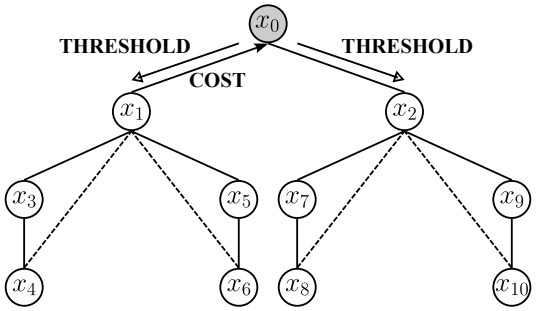

Figure 24: Step 5 (2). x_0 receives the COST message from x_1 and updates the lower bound as $LB_0 = LB_0(0) = lb_0(0, x_1) + lb_0(0, x_2) = 0$. Although LB_0 decreases, the thresholds for the children are kept as $th_0(0, x_1) = 7$ and $th_0(0, x_2) = 0$.

Variables	Step 5 (1)	Step 5 (2)
x_0		
d_0	0	0
LB_0	7	**0**
$lb_0(0, x_1)$	7	**0**
$lb_0(0, x_2)$	0	0
UB_0	∞	∞
$ub_0(0, x_1)$	7	∞
$ub_0(0, x_2)$	∞	∞
TH_0	7	7
$th_0(0, x_1)$	7	7
$th_0(0, x_2)$	0	0
x_1		
d_1	0	0
LB_1	**0**	0
$lb_1(0, x_3)$	**0**	0
$lb_1(0, x_5)$	**0**	0
$lb_1(1, x_3)$	**0**	0
$lb_1(1, x_5)$	3	3
UB_1	∞	∞
$ub_1(0, x_3)$	∞	∞
$ub_1(0, x_5)$	∞	∞
$ub_1(1, x_3)$	∞	∞
$ub_1(1, x_5)$	3	3
TH_1	7	7
x_3		
d_3	0	0
CX_3	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$
$cx_3(0, x_4)$	$\{\}$	$\{\}$
LB_3	0	0
$lb_3(0, x_4)$	0	0
UB_3	∞	∞
$ub_3(0, x_4)$	∞	∞
x_4		
d_4	0	0
CX_4	$\{(x_1, 1),$ $(x_3, 0)\}$	$\{(x_1, 1),$ $(x_3, 0)\}$
LB_4	6	6
UB_4	6	6
x_5		
d_5	0	0
CX_5	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$
$cx_5(0, x_6)$	$\{\}$	$\{\}$
LB_5	0	0
$lb_5(0, x_6)$	0	0
UB_5	∞	∞
$ub_5(0, x_6)$	∞	∞
x_6		
d_6	0	0
CX_6	$\{(x_1, 1),$ $(x_5, 0)\}$	$\{(x_1, 1),$ $(x_5, 0)\}$
LB_6	3	3
UB_6	3	3

Table 8: States of agents in Steps 5 (1)–5 (2).

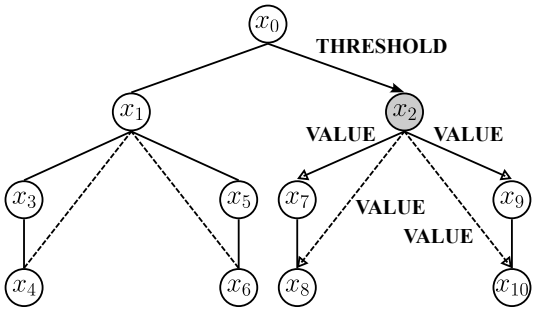

Figure 25: Step 6 (1). The same process is performed in the subtree rooted at x_2.

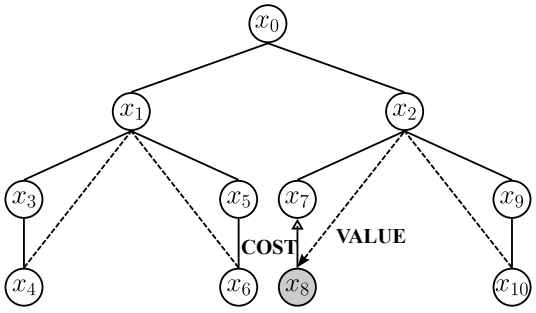

Figure 26: Step 6 (2). The same process is performed in the subtree rooted at x_2.

Variables	Step 6 (1)	Step 6 (2)
x_0		
d_0	0	0
LB_0	0	0
$lb_0(0, x_1)$	0	0
$lb_0(0, x_2)$	0	0
UB_0	∞	∞
$ub_0(0, x_1)$	∞	∞
$ub_0(0, x_2)$	∞	∞
TH_0	7	7
$th_0(0, x_1)$	7	7
$th_0(0, x_2)$	0	0
x_2		
d_2	0	0
LB_2	0	0
$lb_2(0, x_7)$	0	0
$lb_2(0, x_9)$	0	0
$lb_2(1, x_7)$	0	0
$lb_2(1, x_9)$	0	0
UB_2	∞	∞
$ub_2(0, x_7)$	∞	∞
$ub_2(0, x_9)$	∞	∞
$ub_2(1, x_7)$	∞	∞
$ub_2(1, x_9)$	∞	∞
TH_2	0	0
x_7		
d_7	0	0
CX_7	{}	{}
$cx_7(0, x_8)$	{}	{}
LB_7	0	0
$lb_7(0, x_8)$	0	0
UB_7	∞	∞
$ub_7(0, x_8)$	∞	∞
x_8		
d_8	0	0
CX_8	{}	$\boldsymbol{\{(x_1, 0), (x_7, 0)\}}$
LB_8	0	**5**
UB_8	∞	**5**
x_9		
d_9	0	0
CX_9	{}	{}
$cx_9(0, x_{10})$	{}	{}
LB_9	0	0
$lb_9(0, x_{10})$	0	0
UB_9	∞	∞
$ub_9(0, x_{10})$	∞	∞
x_{10}		
d_{10}	0	0
CX_{10}	{}	{}
LB_{10}	0	0
UB_{10}	∞	∞

Table 9: States of agents in Steps 6 (1)–6 (2).

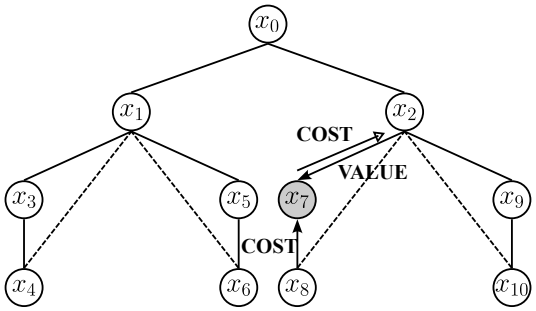

Figure 27: Step 6 (3). The same process is performed in the subtree rooted at x_2.

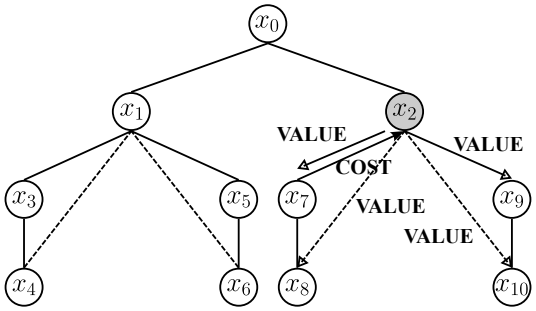

Figure 28: Step 6 (4). The same process is performed in the subtree rooted at x_2.

Variables	Step 6 (3)	Step 6 (4)
x_0		
d_0	0	0
LB_0	0	0
$lb_0(0, x_1)$	0	0
$lb_0(0, x_2)$	0	0
UB_0	∞	∞
$ub_0(0, x_1)$	∞	∞
$ub_0(0, x_2)$	∞	∞
TH_0	7	7
$th_0(0, x_1)$	7	7
$th_0(0, x_2)$	0	0
x_2		
d_2	0	**1**
LB_2	0	0
$lb_2(0, x_7)$	0	**5**
$lb_2(0, x_9)$	0	0
$lb_2(1, x_7)$	0	0
$lb_2(1, x_9)$	0	0
UB_2	∞	∞
$ub_2(0, x_7)$	∞	**5**
$ub_2(0, x_9)$	∞	∞
$ub_2(1, x_7)$	∞	∞
$ub_2(1, x_9)$	∞	∞
TH_2	0	0
x_7		
d_7	0	0
CX_7	$\boldsymbol{\{(x_1, 0)\}}$	$\{(x_1, 0)\}$
$cx_7(0, x_8)$	$\boldsymbol{\{(x_1, 0)\}}$	$\{(x_1, 0)\}$
LB_7	**5**	5
$lb_7(0, x_8)$	**5**	5
UB_7	**5**	5
$ub_7(0, x_8)$	**5**	5
x_8		
d_8	0	0
CX_8	$\{(x_1, 0),$ $(x_7, 0)\}$	$\{(x_1, 0),$ $(x_7, 0)\}$
LB_8	5	5
UB_8	5	5
x_9		
d_9	0	0
CX_9	$\{\}$	$\{\}$
$cx_9(0, x_{10})$	$\{\}$	$\{\}$
LB_9	0	0
$lb_9(0, x_{10})$	0	0
UB_9	∞	∞
$ub_9(0, x_{10})$	∞	∞
x_{10}		
d_{10}	0	0
CX_{10}	$\{\}$	$\{\}$
LB_{10}	0	0
UB_{10}	∞	∞

Table 10: States of agents in Steps 6 (3)–6 (4).

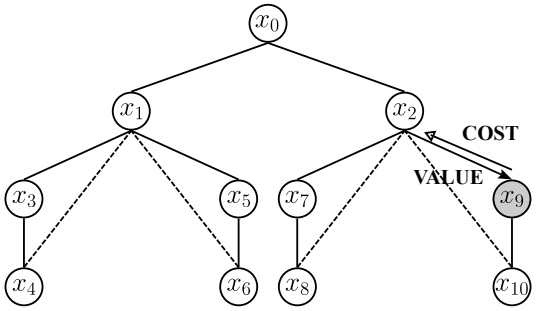

Figure 29: Step 6 (5). The same process is performed in the subtree rooted at x_2.

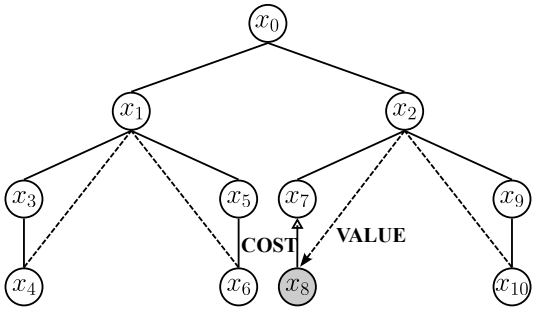

Figure 30: Step 6 (6). The same process is performed in the subtree rooted at x_2.

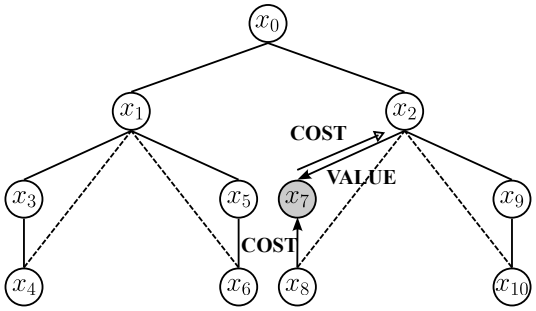

Figure 31: Step 6 (7). The same process is performed in the subtree rooted at x_2.

Variables	Step 6 (5)	Step 6 (6)	Step 6 (7)
x_0			
d_0	0	0	0
LB_0	0	0	0
$lb_0(0, x_1)$	0	0	0
$lb_0(0, x_2)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(0, x_2)$	∞	∞	∞
TH_0	7	7	7
$th_0(0, x_1)$	7	7	7
$th_0(0, x_2)$	0	0	0
x_2			
d_2	1	1	1
LB_2	0	0	0
$lb_2(0, x_7)$	5	5	5
$lb_2(0, x_9)$	0	0	0
$lb_2(1, x_7)$	0	0	0
$lb_2(1, x_9)$	0	0	0
UB_2	∞	∞	∞
$ub_2(0, x_7)$	5	5	5
$ub_2(0, x_9)$	∞	∞	∞
$ub_2(1, x_7)$	∞	∞	∞
$ub_2(1, x_9)$	∞	∞	∞
TH_2	0	0	0
x_7			
d_7	0	0	0
CX_7	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$	$\boldsymbol{\{(x_1, 1)\}}$
$cx_7(0, x_8)$	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$	$\boldsymbol{\{(x_1, 1)\}}$
LB_7	5	5	**6**
$lb_7(0, x_8)$	5	5	**6**
UB_7	5	5	**6**
$ub_7(0, x_8)$	5	5	**6**
x_8			
d_8	0	0	0
CX_8	$\{(x_1, 0),$ $(x_7, 0)\}$	$\boldsymbol{\{(x_1, 1),}$ $\boldsymbol{(x_7, 0)\}}$	$\{(x_1, 1),$ $(x_7, 0)\}$
LB_8	5	**6**	6
UB_8	5	**6**	6
x_9			
d_9	0	0	0
CX_9	$\{\}$	$\{\}$	$\{\}$
$cx_9(0, x_{10})$	$\{\}$	$\{\}$	$\{\}$
LB_9	0	0	0
$lb_9(0, x_{10})$	0	0	0
UB_9	∞	∞	∞
$ub_9(0, x_{10})$	∞	∞	∞
x_{10}			
d_{10}	0	0	0
CX_{10}	$\{\}$	$\{\}$	$\{\}$
LB_{10}	0	0	0
UB_{10}	∞	∞	∞

Table 11: States of agents in Steps 6 (5)–6 (7).

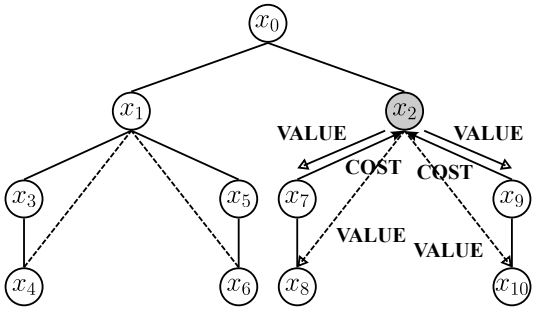

Figure 32: Step 6 (8). The same process is performed in the subtree rooted at x_2.

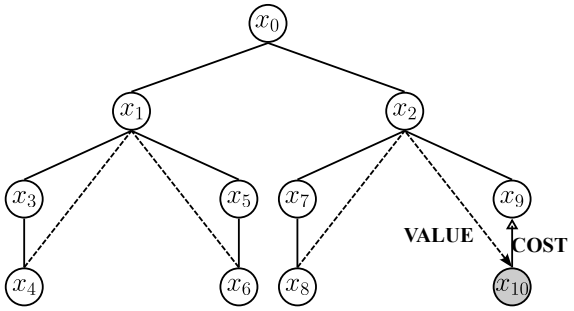

Figure 33: Step 6 (9). The same process is performed in the subtree rooted at x_2.

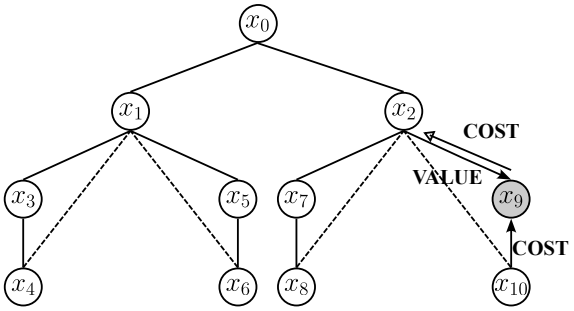

Figure 34: Step 6 (10). The same process is performed in the subtree rooted at x_2.

Variables	Step 6 (8)	Step 6 (9)	Step 6 (10)
x_0			
d_0	0	0	0
LB_0	0	0	0
$lb_0(0, x_1)$	0	0	0
$lb_0(0, x_2)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(0, x_2)$	∞	∞	∞
TH_0	7	7	7
$th_0(0, x_1)$	7	7	7
$th_0(0, x_2)$	0	0	0
x_2			
d_2	**0**	0	0
LB_2	**5**	5	5
$lb_2(0, x_7)$	5	5	5
$lb_2(0, x_9)$	0	0	0
$lb_2(1, x_7)$	**6**	6	6
$lb_2(1, x_9)$	0	0	0
UB_2	∞	∞	∞
$ub_2(0, x_7)$	5	5	5
$ub_2(0, x_9)$	∞	∞	∞
$ub_2(1, x_7)$	**6**	6	6
$ub_2(1, x_9)$	∞	∞	∞
TH_2	**5**	5	5
x_7			
d_7	0	0	0
CX_7	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$
$cx_7(0, x_8)$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$
LB_7	6	6	6
$lb_7(0, x_8)$	6	6	6
UB_7	6	6	6
$ub_7(0, x_8)$	6	6	6
x_8			
d_8	0	0	0
CX_8	$\{(x_1, 1),$ $(x_7, 0)\}$	$\{(x_1, 1),$ $(x_7, 0)\}$	$\{(x_1, 1),$ $(x_7, 0)\}$
LB_8	6	6	6
UB_8	6	6	6
x_9			
d_9	0	0	0
CX_9	$\{\}$	$\{\}$	$\boldsymbol{\{(x_2, 0)\}}$
$cx_9(0, x_{10})$	$\{\}$	$\{\}$	$\boldsymbol{\{(x_2, 0)\}}$
LB_9	0	0	**2**
$lb_9(0, x_{10})$	0	0	**2**
UB_9	∞	∞	**2**
$ub_9(0, x_{10})$	∞	∞	**2**
x_{10}			
d_{10}	0	0	0
CX_{10}	$\{\}$	$\boldsymbol{\{(x_1, 0),}$ $\boldsymbol{(x_9, 0)\}}$	$\{(x_1, 0),$ $(x_9, 0)\}$
LB_{10}	0	**2**	2
UB_{10}	∞	**2**	2

Table 12: States of agents in Steps 6 (8)–6 (10).

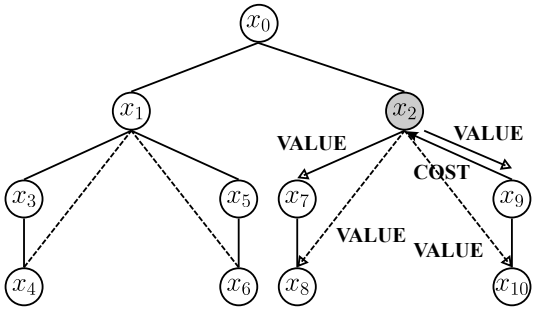

Figure 35: Step 6 (11). The same process is performed in the subtree rooted at x_2.

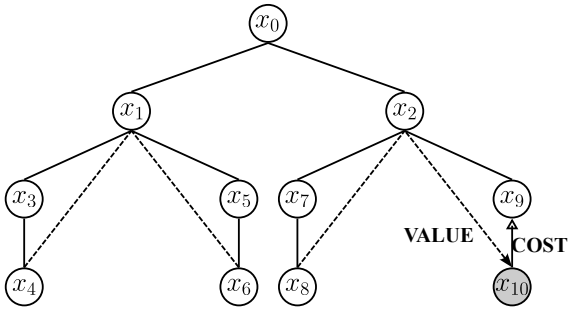

Figure 36: Step 6 (12). The same process is performed in the subtree rooted at x_2.

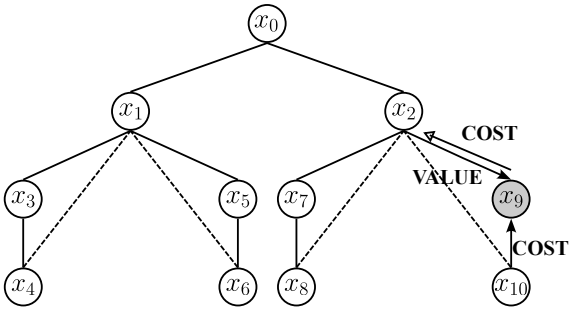

Figure 37: Step 6 (13). The same process is performed in the subtree rooted at x_2.

Variables	Step 6 (11)	Step 6 (12)	Step 6 (13)
x_0			
d_0	0	0	0
LB_0	0	0	0
$lb_0(0, x_1)$	0	0	0
$lb_0(0, x_2)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(0, x_2)$	∞	∞	∞
TH_0	7	7	7
$th_0(0, x_1)$	7	7	7
$th_0(0, x_2)$	0	0	0
x_2			
d_2	**1**	1	1
LB_2	**6**	6	6
$lb_2(0, x_7)$	5	5	5
$lb_2(0, x_9)$	**2**	2	2
$lb_2(1, x_7)$	6	6	6
$lb_2(1, x_9)$	0	0	0
UB_2	**7**	7	7
$ub_2(0, x_7)$	5	5	5
$ub_2(0, x_9)$	**2**	2	2
$ub_2(1, x_7)$	6	6	6
$ub_2(1, x_9)$	∞	∞	∞
TH_2	**6**	6	6
x_7			
d_7	0	0	0
CX_7	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$
$cx_7(0, x_8)$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$
LB_7	6	6	6
$lb_7(0, x_8)$	6	6	6
UB_7	6	6	6
$ub_7(0, x_8)$	6	6	6
x_8			
d_8	0	0	0
CX_8	$\{(x_1, 1),$ $(x_7, 0)\}$	$\{(x_1, 1),$ $(x_7, 0)\}$	$\{(x_1, 1),$ $(x_7, 0)\}$
LB_8	6	6	6
UB_8	6	6	6
x_9			
d_9	0	0	0
CX_9	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$	$\boldsymbol{\{(x_1, 1)\}}$
$cx_9(0, x_{10})$	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$	$\boldsymbol{\{(x_1, 1)\}}$
LB_9	2	2	**3**
$lb_9(0, x_{10})$	2	2	**3**
UB_9	2	2	**3**
$ub_9(0, x_{10})$	2	2	**3**
x_{10}			
d_{10}	0	0	0
CX_{10}	$\{(x_1, 0),$ $(x_9, 0)\}$	$\boldsymbol{\{(x_1, 1),}$ $\boldsymbol{(x_9, 0)\}}$	$\{(x_1, 1),$ $(x_9, 0)\}$
LB_{10}	2	**3**	3
UB_{10}	2	**3**	3

Table 13: States of agents in Steps 6 (11)–6 (13).

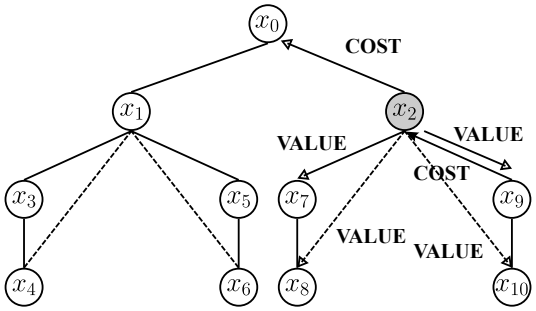

Figure 38: Step 6 (14). The same process is performed in the subtree rooted at x_2.

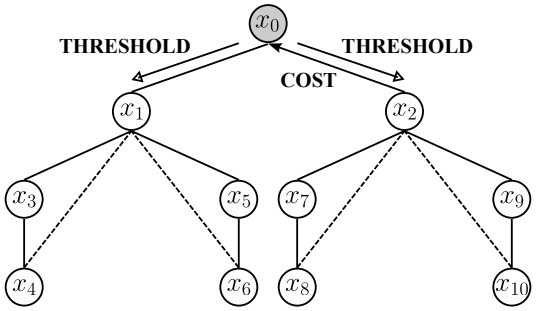

Figure 39: Step 6 (15). The same process is performed in the subtree rooted at x_2.

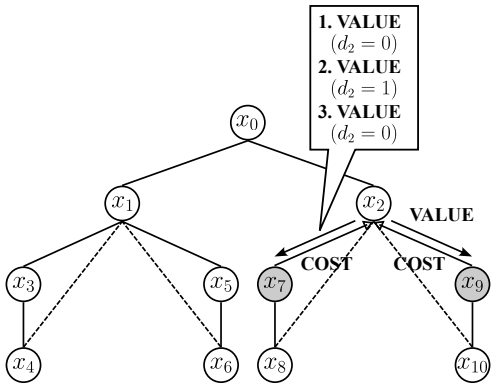

Figure 40: Step 6 (16). The same process is performed in the subtree rooted at x_2.

Variables	Step 6 (14)	Step 6 (15)	Step 6 (16)
x_0			
d_0	0	0	0
LB_0	0	**7**	7
$lb_0(0, x_1)$	0	0	0
$lb_0(0, x_2)$	0	**7**	7
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(0, x_2)$	∞	**7**	7
TH_0	7	7	7
$th_0(0, x_1)$	7	0	0
$th_0(0, x_2)$	0	**7**	7
x_2			
d_2	**0**	0	0
LB_2	**7**	7	7
$lb_2(0, x_7)$	5	5	5
$lb_2(0, x_9)$	2	2	2
$lb_2(1, x_7)$	6	6	6
$lb_2(1, x_9)$	**3**	3	3
UB_2	7	7	7
$ub_2(0, x_7)$	5	5	5
$ub_2(0, x_9)$	2	2	2
$ub_2(1, x_7)$	6	6	6
$ub_2(1, x_9)$	**3**	3	3
TH_2	**7**	7	7
x_7			
d_7	0	0	0
CX_7	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\boldsymbol{\{(x_1, 0)\}}$
$cx_7(0, x_8)$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	**{}**
LB_7	6	6	**0**
$lb_7(0, x_8)$	6	6	**0**
UB_7	6	6	$\boldsymbol{\infty}$
$ub_7(0, x_8)$	6	6	$\boldsymbol{\infty}$
x_8			
d_8	0	0	0
CX_8	$\{(x_1, 1), (x_7, 0)\}$	$\{(x_1, 1), (x_7, 0)\}$	$\{(x_1, 1), (x_7, 0)\}$
LB_8	6	6	6
UB_8	6	6	6
x_9			
d_9	0	0	0
CX_9	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	$\boldsymbol{\{(x_1, 0)\}}$
$cx_9(0, x_{10})$	$\{(x_1, 1)\}$	$\{(x_1, 1)\}$	**{}**
LB_9	3	3	**0**
$lb_9(0, x_{10})$	3	3	**0**
UB_9	3	3	$\boldsymbol{\infty}$
$ub_9(0, x_{10})$	3	3	$\boldsymbol{\infty}$
x_{10}			
d_{10}	0	0	0
CX_{10}	$\{(x_1, 1), (x_9, 0)\}$	$\{(x_1, 1), (x_9, 0)\}$	$\{(x_1, 1), (x_9, 0)\}$
LB_{10}	3	3	3
UB_{10}	3	3	3

Table 14: States of agents in Steps 6 (14)–6 (16).

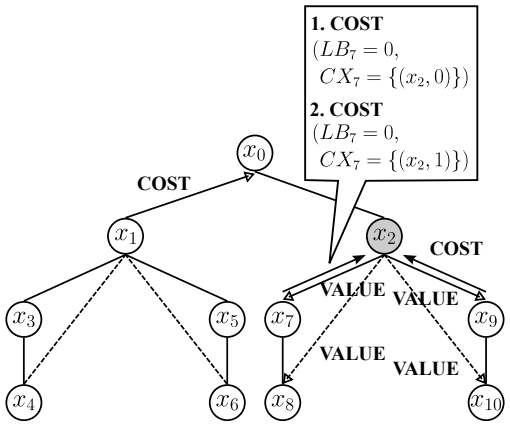

Figure 41: Step 6 (17). The same process is performed in the subtree rooted at x_2.

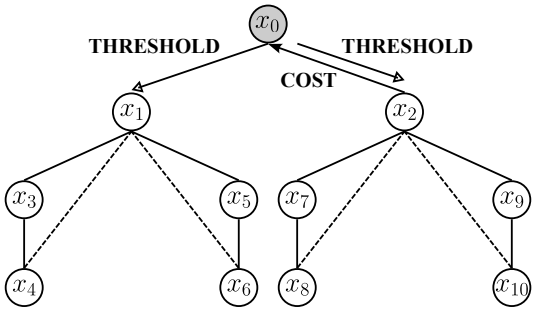

Figure 42: Step 6 (18). The same process is performed in the subtree rooted at x_2.

Variables	Step 6 (17)	Step 6 (18)
x_0		
d_0	0	0
LB_0	7	**0**
$lb_0(0, x_1)$	0	0
$lb_0(0, x_2)$	7	**0**
UB_0	∞	∞
$ub_0(0, x_1)$	∞	∞
$ub_0(0, x_2)$	7	$\boldsymbol{\infty}$
TH_0	7	7
$th_0(0, x_1)$	0	0
$th_0(0, x_2)$	7	7
x_2		
d_2	0	0
LB_2	**0**	0
$lb_2(0, x_7)$	**0**	0
$lb_2(0, x_9)$	**0**	0
$lb_2(1, x_7)$	**0**	0
$lb_2(1, x_9)$	3	3
UB_2	$\boldsymbol{\infty}$	∞
$ub_2(0, x_7)$	$\boldsymbol{\infty}$	∞
$ub_2(0, x_9)$	$\boldsymbol{\infty}$	∞
$ub_2(1, x_7)$	$\boldsymbol{\infty}$	∞
$ub_2(1, x_9)$	3	3
TH_2	7	7
x_7		
d_7	0	0
CX_7	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$
$cx_7(0, x_8)$	$\{\}$	$\{\}$
LB_7	0	0
$lb_7(0, x_8)$	0	0
UB_7	∞	∞
$ub_7(0, x_8)$	∞	∞
x_8		
d_8	0	0
CX_8	$\{(x_1, 1),$ $(x_7, 0)\}$	$\{(x_1, 1),$ $(x_7, 0)\}$
LB_8	6	6
UB_8	6	6
x_9		
d_9	0	0
CX_9	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$
$cx_9(0, x_{10})$	$\{\}$	$\{\}$
LB_9	0	0
$lb_9(0, x_{10})$	0	0
UB_9	∞	∞
$ub_9(0, x_{10})$	∞	∞
x_{10}		
d_{10}	0	0
CX_{10}	$\{(x_1, 1),$ $(x_9, 0)\}$	$\{(x_1, 1),$ $(x_9, 0)\}$
LB_{10}	3	3
UB_{10}	3	3

Table 15: States of agents in Steps 6 (17)–6 (18).

B.2 Counterexample to Optimality Caused by Initialization

Figures 43–46 and Tables 16–17 show the trace of the counterexample to optimality caused by initialization, described in 3.2.

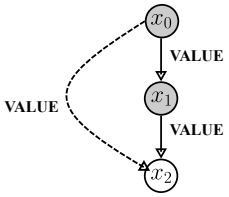

Figure 43: Step 1 (1). Agents send VALUE messages to their children and pseudo-children.

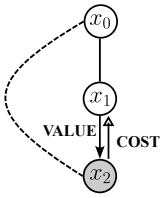

Figure 44: Step 1 (2). x_2 receives the VALUE message only from its parent x_1, which means that the messages from the pseudo-parent x_0 are delayed. Here, x_2 updates the current context as $CX_2 = \{(x_1, 0)\}$. From the definition of the local cost $\delta_i(d, CX)$, x_2 computes the local costs as $\delta_2(0, \{(x_1, 0)\}) = f_{1,2}(0, 0) = 0$ and $\delta_2(1, \{(x_1, 0)\}) = f_{1,2}(0, 1) = 0$. Thus, the bounds of x_2 are obtained as $LB_2 = UB_2 = 0$. Then x_2 sends a COST message with $LB_2 = UB_2 = 0$ to x_1.

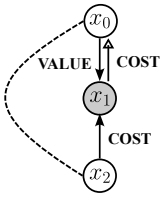

Figure 45: Step 2 (1). After x_1 receives the VALUE message from x_0 and the COST message from x_2, x_1 computes the bounds as $LB_1 = UB_1 = 0$.

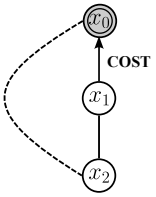

Figure 46: Step 2 (2). x_0 updates the bounds as $LB_0(0) = UB_0(0) = 0$ through the COST message sent from x_1. Since $LB_0 = TH_0 = UB_0 = UB_0(0) = 0$ due to ThresholdInvariant, x_0 keeps its value as $d_0 = 0$ and satisfies the termination condition. However, the variable value $d_0 = 0$ is suboptimal.

Variables	Step 0	Step 1 (1)	Step 1 (2)
x_0			
d_0	0	0	0
LB_0	0	0	0
$lb_0(0, x_1)$	0	0	0
$lb_0(1, x_1)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(1, x_1)$	∞	∞	∞
TH_0	0	0	0
x_1			
d_1	0	0	0
CX_1	$\{\}$	$\{\}$	$\{\}$
$cx_1(0, x_2)$	$\{\}$	$\{\}$	$\{\}$
LB_1	0	0	0
$lb_1(0, x_2)$	0	0	0
UB_1	∞	∞	∞
$ub_1(0, x_2)$	∞	∞	∞
TH_1	0	0	0
x_2			
d_2	0	0	0
CX_2	$\{\}$	$\{\}$	$\mathbf{\{(x_1, 0)\}}$
LB_2	0	0	0
UB_2	∞	∞	**0**

Table 16: States of agents in Steps 0–1 (2).

Variables	Step 2 (1)	Step 2 (2)
x_0		
d_0	0	0
LB_0	0	0
$lb_0(0, x_1)$	0	0
$lb_0(1, x_1)$	0	0
UB_0	∞	**0**
$ub_0(0, x_1)$	∞	**0**
$ub_0(1, x_1)$	∞	∞
TH_0	0	0
x_1		
d_1	0	0
CX_1	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
$cx_1(0, x_2)$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
LB_1	0	0
$lb_1(0, x_2)$	0	0
UB_1	**0**	0
$ub_1(0, x_2)$	**0**	0
TH_1	0	0
x_2		
d_2	0	0
CX_2	$\{(x_1, 0)\}$	$\{(x_1, 0)\}$
LB_2	0	0
UB_2	0	0

Table 17: States of agents in Steps 2 (1)–2 (2).

B.3 Counterexample to Optimality Caused by TERMINATE Messages

Figures 47–62 and Tables 18–23 show the trace of the counterexample to optimality caused by TERMINATE messages, described in 3.3. Additionally, the trace where x_2 performs reinitialization when it receives a TERMINATE message, described in "Cause of Counterexample", is shown as Step 6' in Figures 63–65 and Table 24.

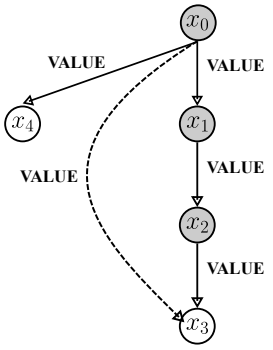

Figure 47: Step 0 (1). Agents calculate the cost in the case where x_0 takes the value 0.

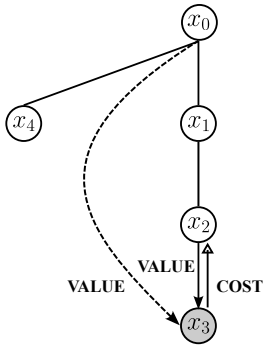

Figure 48: Step 0 (2). Agents calculate the cost in the case where x_0 takes the value 0.

Variables	Step 0 (1)	Step 0 (2)
x_0		
d_0	0	0
LB_0	0	0
$lb_0(0, x_1)$	0	0
$lb_0(0, x_4)$	0	0
$lb_0(1, x_1)$	0	0
$lb_0(1, x_4)$	0	0
UB_0	∞	∞
$ub_0(0, x_1)$	∞	∞
$ub_0(0, x_4)$	∞	∞
$ub_0(1, x_1)$	∞	∞
$ub_0(1, x_4)$	∞	∞
TH_0	0	0
$th_0(0, x_1)$	0	0
$th_0(0, x_4)$	0	0
$th_0(1, x_1)$	0	0
$th_0(1, x_4)$	0	0
x_1		
d_1	0	0
CX_1	$\{\}$	$\{\}$
$cx_1(0, x_2)$	$\{\}$	$\{\}$
LB_1	0	0
$lb_1(0, x_2)$	0	0
UB_1	∞	∞
$ub_1(0, x_2)$	∞	∞
TH_1	0	0
x_2		
d_2	0	0
CX_2	$\{\}$	$\{\}$
$cx_2(0, x_3)$	$\{\}$	$\{\}$
$cx_2(1, x_3)$	$\{\}$	$\{\}$
LB_2	0	0
$lb_2(0, x_3)$	0	0
$lb_2(1, x_3)$	0	0
UB_2	∞	∞
$ub_2(0, x_3)$	∞	∞
$ub_2(1, x_3)$	∞	∞
TH_2	0	0
x_3		
d_3	0	0
CX_3	$\{\}$	$\{(x_0, 0), (x_2, 0)\}$
LB_3	0	**1**
UB_3	∞	**1**
x_4		
d_4	0	0
CX_4	$\{\}$	$\{\}$
LB_4	0	0
UB_4	∞	∞

Table 18: States of agents in Steps 0 (1)–0 (2).

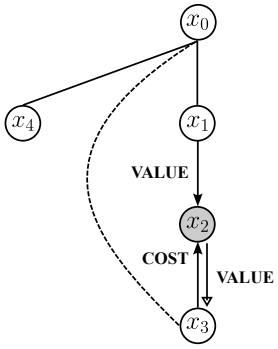

Figure 49: Step 0 (3). Agents calculate the cost in the case where x_0 takes the value 0.

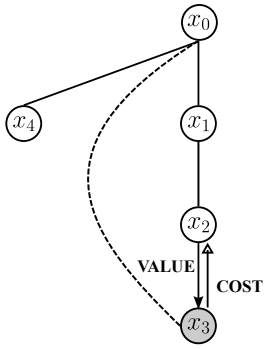

Figure 50: Step 0 (4). Agents calculate the cost in the case where x_0 takes the value 0.

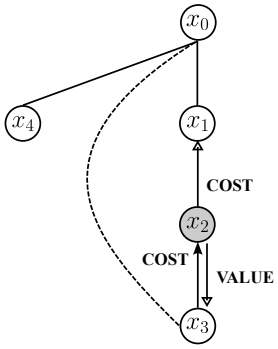

Figure 51: Step 0 (5). Agents calculate the cost in the case where x_0 takes the value 0.

Variables	Step 0 (3)	Step 0 (4)	Step 0 (5)
x_0			
d_0	0	0	0
LB_0	0	0	0
$lb_0(0, x_1)$	0	0	0
$lb_0(0, x_4)$	0	0	0
$lb_0(1, x_1)$	0	0	0
$lb_0(1, x_4)$	0	0	0
UB_0	∞	∞	∞
$ub_0(0, x_1)$	∞	∞	∞
$ub_0(0, x_4)$	∞	∞	∞
$ub_0(1, x_1)$	∞	∞	∞
$ub_0(1, x_4)$	∞	∞	∞
TH_0	0	0	0
$th_0(0, x_1)$	0	0	0
$th_0(0, x_4)$	0	0	0
$th_0(1, x_1)$	0	0	0
$th_0(1, x_4)$	0	0	0
x_1			
d_1	0	0	0
CX_1	$\{\}$	$\{\}$	$\{\}$
$cx_1(0, x_2)$	$\{\}$	$\{\}$	$\{\}$
LB_1	0	0	0
$lb_1(0, x_2)$	0	0	0
UB_1	∞	∞	∞
$ub_1(0, x_2)$	∞	∞	∞
TH_1	0	0	0
x_2			
d_2	**1**	1	**0**
CX_2	$\boldsymbol{\{(x_0, 0),}$ $\boldsymbol{(x_1, 0)\}}$	$\{(x_0, 0),$ $(x_1, 0)\}$	$\{(x_0, 0),$ $(x_1, 0)\}$
$cx_2(0, x_3)$	$\boldsymbol{\{(x_0, 0)\}}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
$cx_2(1, x_3)$	$\{\}$	$\{\}$	$\boldsymbol{\{(x_0, 0)\}}$
LB_2	0	0	**1**
$lb_2(0, x_3)$	**1**	1	1
$lb_2(1, x_3)$	0	0	**11**
UB_2	**1**	1	1
$ub_2(0, x_3)$	**1**	1	1
$ub_2(1, x_3)$	∞	∞	**11**
TH_2	0	0	**1**
x_3			
d_3	0	0	0
CX_3	$\{(x_0, 0),$ $(x_2, 0)\}$	$\boldsymbol{\{(x_0, 0),}$ $\boldsymbol{(x_2, 1)\}}$	$\{(x_0, 0),$ $(x_2, 1)\}$
LB_3	1	**11**	11
UB_3	1	**11**	11
x_4			
d_4	0	0	0
CX_4	$\{\}$	$\{\}$	$\{\}$
LB_4	0	0	0
UB_4	∞	∞	∞

Table 19: States of agents in Steps 0 (3)–0 (5).

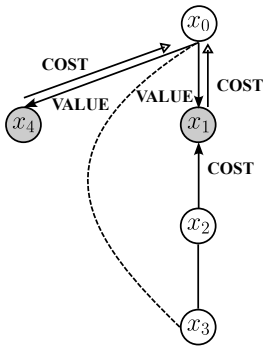

Figure 52: Step 0 (6). Agents calculate the cost in the case where x_0 takes the value 0.

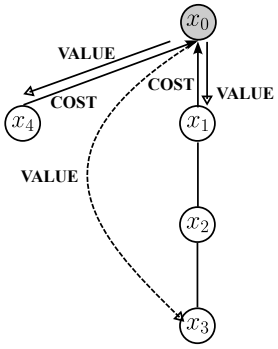

Figure 53: Step 1 (1). x_0 sends VALUE messages with $d_0 = 1$ to x_1, x_3, and x_4.

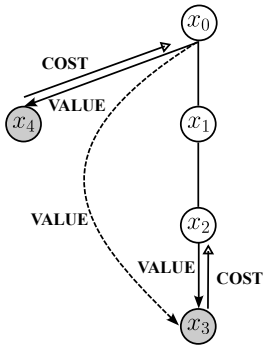

Figure 54: Step 1 (2). x_3 and x_4 receive the VALUE messages. At this moment, x_3 updates the context and the bounds as $CX_3 = \{(x_0, 1), (x_2, 0)\}$ and $LB_3 = UB_3 = 200$, and x_4 also updates them as $CX_4 = \{(x_0, 1)\}$ and $LB_4 = UB_4 = 1000$. Then they send COST messages to their parents: from x_3 to x_2 and from x_4 to x_0.

Variables	Step 0 (6)	Step 1 (1)	Step 1 (2)
x_0			
d_0	0	**1**	1
LB_0	0	0	0
$lb_0(0, x_1)$	0	**1**	1
$lb_0(0, x_4)$	0	0	0
$lb_0(1, x_1)$	0	0	0
$lb_0(1, x_4)$	0	0	0
UB_0	∞	**1**	1
$ub_0(0, x_1)$	∞	**1**	1
$ub_0(0, x_4)$	∞	**0**	0
$ub_0(1, x_1)$	∞	∞	∞
$ub_0(1, x_4)$	∞	∞	∞
TH_0	0	0	0
$th_0(0, x_1)$	0	**1**	1
$th_0(0, x_4)$	0	0	0
$th_0(1, x_1)$	0	0	0
$th_0(1, x_4)$	0	0	0
x_1			
d_1	0	0	0
CX_1	$\mathbf{\{(x_0, 0)\}}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
$cx_1(0, x_2)$	$\mathbf{\{(x_0, 0)\}}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
LB_1	**1**	1	1
$lb_1(0, x_2)$	**1**	1	1
UB_1	**1**	1	1
$ub_1(0, x_2)$	**1**	1	1
TH_1	**1**	1	1
x_2			
d_2	0	0	0
CX_2	$\{(x_0, 0), (x_1, 0)\}$	$\{(x_0, 0), (x_1, 0)\}$	$\{(x_0, 0), (x_1, 0)\}$
$cx_2(0, x_3)$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
$cx_2(1, x_3)$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
LB_2	1	1	1
$lb_2(0, x_3)$	1	1	1
$lb_2(1, x_3)$	11	11	11
UB_2	1	1	1
$ub_2(0, x_3)$	1	1	1
$ub_2(1, x_3)$	11	11	11
TH_2	1	1	1
x_3			
d_3	0	0	**1**
CX_3	$\{(x_0, 0), (x_2, 1)\}$	$\{(x_0, 0), (x_2, 1)\}$	$\mathbf{\{(x_0, 1), (x_2, 0)\}}$
LB_3	11	11	**200**
UB_3	11	11	**200**
x_4			
d_4	0	0	0
CX_4	$\mathbf{\{(x_0, 0)\}}$	$\{(x_0, 0)\}$	$\mathbf{\{(x_0, 1)\}}$
LB_4	0	0	**1000**
UB_4	**0**	0	**1000**

Table 20: States of agents in Steps 0 (6)–1 (2).

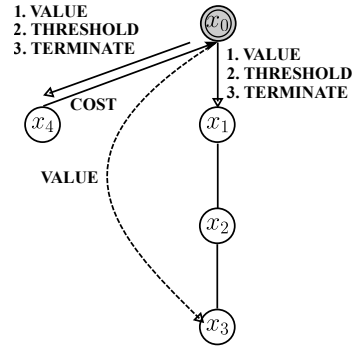

Figure 55: Step 1 (3). After x_0 receives the COST message from x_4, x_0 changes its value d_0 back to 0 since the bounds are obtained as $LB_0(1) = 1000$ and $LB_0 = TH_0 = UB_0 = UB_0(0) = 1$ due to ThresholdInvariant. Therefore, x_0 satisfies the termination condition and then terminates. When x_0 executes termination, x_0 sends VALUE messages with $d_0 = 0$ to its lower neighbors (i.e., x_1, x_3, and x_4) and THRESHOLD and TERMINATE messages to its children (i.e., x_1 and x_4) in this order.

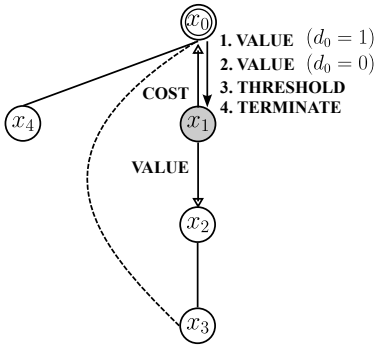

Figure 56: Step 2. x_1 receives the VALUE messages from x_0, including the message that x_0 sent when $d_0 = 1$, in the order of sending. Then x_1 changes CX_1 from $\{(x_0, 0)\}$ into $\{(x_0, 1)\}$ and returns it to $\{(x_0, 0)\}$. Since $\{(x_0, 1)\}$ is incompatible with $cx_1(0, x_2) = \{(x_0, 0)\}$, x_1 reinitializes the bounds as $LB_1 = 0$ and $UB_1 = \infty$. By contrast, TH_1 is not changed from 1. Furthermore, x_1 receives the THRESHOLD and TERMINATE messages from x_0, and then x_1 records receiving the TERMINATE message but does not terminate because $TH_1 = 1 < UB_1 = \infty$. Additionally, x_1 sends a VALUE message to x_2.

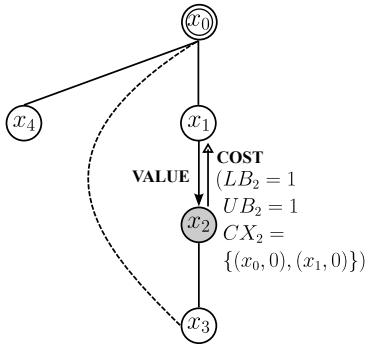

Figure 57: Step 3. x_2 receives only the message from x_1, not from x_3, which means that the messages from x_3 are delayed. Then x_2 sends a COST message to x_1 with $CX_2 = \{(x_0, 0), (x_1, 0)\}$ and $LB_2 = UB_2 = 1$, which are the same states as in Step 0.

Variables	Step 1 (3)	Step 2	Step 3
x_0			
d_0	**0**	0	0
LB_0	**1**	1	1
$lb_0(0, x_1)$	1	1	1
$lb_0(0, x_4)$	0	0	0
$lb_0(1, x_1)$	0	0	0
$lb_0(1, x_4)$	**1000**	1000	1000
UB_0	1	1	1
$ub_0(0, x_1)$	1	1	1
$ub_0(0, x_4)$	0	0	0
$ub_0(1, x_1)$	∞	∞	∞
$ub_0(1, x_4)$	**1000**	1000	1000
TH_0	**1**	1	1
$th_0(0, x_1)$	1	1	1
$th_0(0, x_4)$	0	0	0
$th_0(1, x_1)$	0	0	0
$th_0(1, x_4)$	**1000**	1000	1000
x_1			
d_1	0	0	0
CX_1	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
$cx_1(0, x_2)$	$\{(x_0, 0)\}$	**{}**	$\{\}$
LB_1	1	**0**	0
$lb_1(0, x_2)$	1	**0**	0
UB_1	1	**∞**	∞
$ub_1(0, x_2)$	1	**∞**	∞
TH_1	1	1	1
x_2			
d_2	0	0	0
CX_2	$\{(x_0, 0), (x_1, 0)\}$	$\{(x_0, 0), (x_1, 0)\}$	$\{(x_0, 0), (x_1, 0)\}$
$cx_2(0, x_3)$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
$cx_2(1, x_3)$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
LB_2	1	1	1
$lb_2(0, x_3)$	1	1	1
$lb_2(1, x_3)$	11	11	11
UB_2	1	1	1
$ub_2(0, x_3)$	1	1	1
$ub_2(1, x_3)$	11	11	11
TH_2	1	1	1
x_3			
d_3	1	1	1
CX_3	$\{(x_0, 1), (x_2, 0)\}$	$\{(x_0, 1), (x_2, 0)\}$	$\{(x_0, 1), (x_2, 0)\}$
LB_3	200	200	200
UB_3	200	200	200
x_4			
d_4	0	0	0
CX_4	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$
LB_4	1000	1000	1000
UB_4	1000	1000	1000

Table 21: States of agents in Steps 1 (3)–3.

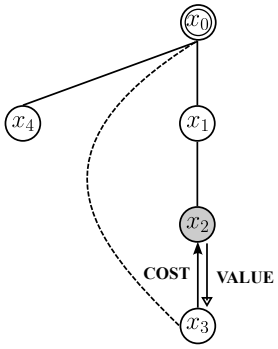

Figure 58: Step 4 (1). x_2 receives the COST message from x_3 with $CX_3 = \{(x_0, 1), (x_2, 0)\}$ and $LB_3 = UB_3 = 200$. Since x_2 is not a neighbor of x_0, x_2 updates CX_2 from $\{(x_0, 0), (x_1, 0)\}$ to $\{(x_0, 1), (x_1, 0)\}$. Then the bounds of x_2 are reinitialized and updated by the bounds in the message: $LB_2(0) = UB_2(0) = 200, LB_2(1) = 0$, and $UB_2(1) = \infty$. x_2 also changes its value d_2 to 1 because $LB_2(0) > TH_2 = 1$, and sends a VALUE message to x_3.

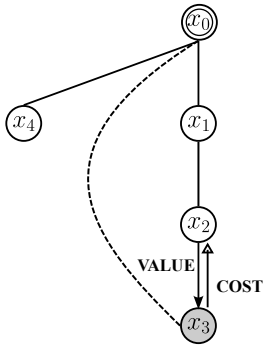

Figure 59: Step 4 (2). After x_3 receives only the VALUE message from x_2 but not the messages from x_1, x_3 updates the current context and the bounds as $CX_3 = \{(x_0, 1), (x_2, 1)\}$ and $LB_3 = UB_3 = 100$. Next, x_3 sends a COST message to x_2 again.

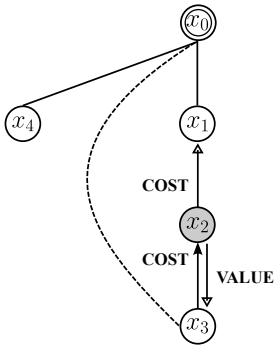

Figure 60: Step 4 (3). x_2 receives it. Then x_2 computes the bounds as $LB_2(1) = UB_2(1) = 100$ and updates the threshold as $TH_2 = 100$ because of ThresholdInvariant.

Trace	Step 4 (1)	Step 4 (2)	Step 4 (3)
x_0			
d_0	0	0	0
LB_0	1	1	1
$lb_0(0, x_1)$	1	1	1
$lb_0(0, x_4)$	0	0	0
$lb_0(1, x_1)$	0	0	0
$lb_0(1, x_4)$	1000	1000	1000
UB_0	1	1	1
$ub_0(0, x_1)$	1	1	1
$ub_0(0, x_4)$	0	0	0
$ub_0(1, x_1)$	∞	∞	∞
$ub_0(1, x_4)$	1000	1000	1000
TH_0	1	1	1
$th_0(0, x_1)$	1	1	1
$th_0(0, x_4)$	0	0	0
$th_0(1, x_1)$	0	0	0
$th_0(1, x_4)$	1000	1000	1000
x_1			
d_1	0	0	0
CX_1	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
$cx_1(0, x_2)$	$\{\}$	$\{\}$	$\{\}$
LB_1	0	0	0
$lb_1(0, x_2)$	0	0	0
UB_1	∞	∞	∞
$ub_1(0, x_2)$	∞	∞	∞
TH_1	1	1	1
x_2			
d_2	**1**	1	1
CX_2	$\boldsymbol{\{(x_0, 1),}$ $\boldsymbol{(x_1, 0)\}}$	$\{(x_0, 1),$ $(x_1, 0)\}$	$\{(x_0, 1),$ $(x_1, 0)\}$
$cx_2(0, x_3)$	$\boldsymbol{\{(x_0, 1)\}}$	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$
$cx_2(1, x_3)$	$\boldsymbol{\{\}}$	$\{\}$	$\boldsymbol{\{(x_0, 1)\}}$
LB_2	**0**	0	**100**
$lb_2(0, x_3)$	**200**	200	200
$lb_2(1, x_3)$	**0**	0	**100**
UB_2	**200**	200	**100**
$ub_2(0, x_3)$	**200**	200	200
$ub_2(1, x_3)$	$\boldsymbol{\infty}$	∞	**100**
TH_2	1	1	**100**
x_3			
d_3	1	1	1
CX_3	$\{(x_0, 1),$ $(x_2, 0)\}$	$\boldsymbol{\{(x_0, 1),}$ $\boldsymbol{(x_2, 1)\}}$	$\{(x_0, 1),$ $(x_2, 1)\}$
LB_3	200	**100**	100
UB_3	200	**100**	100
x_4			
d_4	0	0	0
CX_4	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$
LB_4	1000	1000	1000
UB_4	1000	1000	1000

Table 22: States of agents in Steps 4 (1)–4 (3).

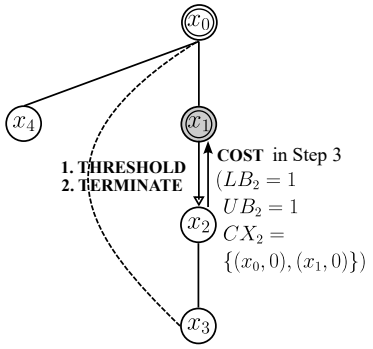

Figure 61: Step 5. x_1 receives the COST message with $CX_2 = \{(x_0, 0), (x_1, 0)\}$ and $LB_2 = UB_2 = 1$, sent from x_2 in Step 3. Since this context is compatible with $CX_1 = \{(x_0, 0)\}$, x_1 updates the bounds as $LB_1 = UB_1 = 1$. At this moment, the termination condition is satisfied because $TH_1 = UB_1 = 1$. Thus, x_1 sends two messages to x_2: a THRESHOLD message with $th_1(0, x_2) = 1$ and $CX_1 = \{(x_0, 0)\}$ and a TERMINATE message with $CX_1 \cup \{(x_1, 0)\} = \{(x_0, 0), (x_1, 0)\}$. Then x_1 terminates.

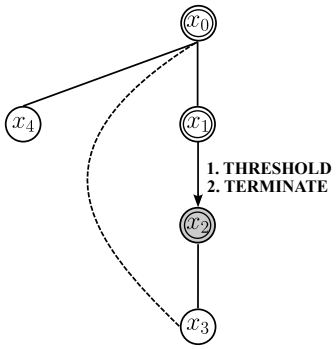

Figure 62: Step 6. When x_2 receives the THRESHOLD message from x_1, x_2 does not update TH_2 since context $\{(x_0, 0)\}$ in the message is incompatible with $CX_2 = \{(x_0, 1), (x_1, 0)\}$, and therefore retains its threshold as $TH_2 = 100$. Next, x_2 receives the TERMINATE message from x_1. Although CX_2 is changed to $\{(x_0, 0), (x_1, 0)\}$, the bounds of x_2 are not changed since reinitialization is not performed when an agent receives a TERMINATE message. Therefore, x_2 terminates with the suboptimal value $d_2 = 1$ because x_2 has already satisfied the termination condition with $UB_2 = UB_2(1) = TH_2 = 100$ and received the TERMINATE message.

Variables	Step 5	Step 6
x_0		
d_0	0	0
LB_0	1	1
$lb_0(0, x_1)$	1	1
$lb_0(0, x_4)$	0	0
$lb_0(1, x_1)$	0	0
$lb_0(1, x_4)$	1000	1000
UB_0	1	1
$ub_0(0, x_1)$	1	1
$ub_0(0, x_4)$	0	0
$ub_0(1, x_1)$	∞	∞
$ub_0(1, x_4)$	1000	1000
TH_0	1	1
$th_0(0, x_1)$	1	1
$th_0(0, x_4)$	0	0
$th_0(1, x_1)$	0	0
$th_0(1, x_4)$	1000	1000
x_1		
d_1	0	0
CX_1	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
$cx_1(0, x_2)$	$\mathbf{\{(x_0, 0)\}}$	$\{(x_0, 0)\}$
LB_1	**1**	1
$lb_1(0, x_2)$	**1**	1
UB_1	**1**	1
$ub_1(0, x_2)$	**1**	1
TH_1	1	1
x_2		
d_2	1	1
CX_2	$\{(x_0, 1), (x_1, 0)\}$	$\mathbf{\{(x_0, 0), (x_1, 0)\}}$
$cx_2(0, x_3)$	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$
$cx_2(1, x_3)$	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$
LB_2	100	100
$lb_2(0, x_3)$	200	200
$lb_2(1, x_3)$	100	100
UB_2	100	100
$ub_2(0, x_3)$	200	200
$ub_2(1, x_3)$	100	100
TH_2	100	100
x_3		
d_3	1	1
CX_3	$\{(x_0, 1), (x_2, 1)\}$	$\{(x_0, 1), (x_2, 1)\}$
LB_3	100	100
UB_3	100	100
x_4		
d_4	0	0
CX_4	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$
LB_4	1000	1000
UB_4	1000	1000

Table 23: States of agents in Steps 5–6.

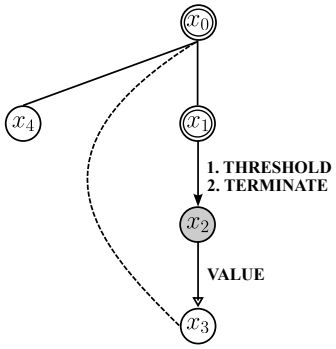

Figure 63: Step 6' (1). The bounds of x_2 are reinitialized when x_2 receives the TERMINATE message from x_1 in Step 6. In this case, the bounds of x_2 are obtained as $LB_2(0) = LB_2(1) = 0$ and $UB_2(0) = UB_2(1) = \infty$; and the threshold of x_2 is obtained as $TH_2 = 100$.

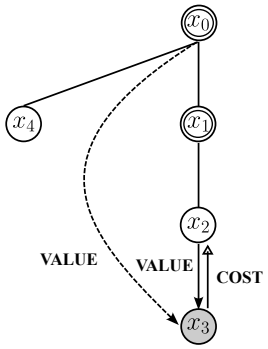

Figure 64: Step 6' (2). After x_3 receives the VALUE message with the value $d_0 = 0$ from x_0, x_3 updates the current context as $CX_3 = \{(x_0, 0), (x_2, 1)\}$ and the bounds as $LB_3 = UB_3 = 11$. Then x_3 sends a COST message to x_2.

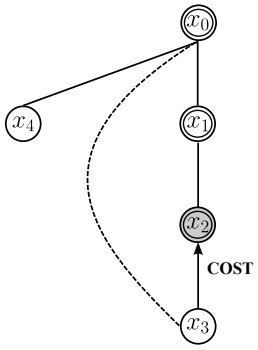

Figure 65: Step 6' (3). x_2 receives the COST message from x_3. At this point, x_2 computes the bounds as $LB_2(1) = UB_2(1) = 11$ because context $\{(x_0, 0), (x_2, 1)\}$ in the COST message is compatible with $CX_2 = \{(x_0, 0), (x_1, 0)\}$. Then x_2 updates the threshold as $TH_2 = 11$ due to ThresholdInvariant. Since $UB_2 = UB_2(1) = TH_2$, x_2 does not change its variable value d_2 from 1. Therefore, x_2 satisfies the termination condition and terminates with the sub-optimal value $d_2 = 1$.

Variables	Step 6' (1)	Step 6' (2)	Step 6' (3)
x_0			
d_0	0	0	0
LB_0	1	1	1
$lb_0(0, x_1)$	1	1	1
$lb_0(0, x_4)$	0	0	0
$lb_0(1, x_1)$	0	0	0
$lb_0(1, x_4)$	1000	1000	1000
UB_0	1	1	1
$ub_0(0, x_1)$	1	1	1
$ub_0(0, x_4)$	0	0	0
$ub_0(1, x_1)$	∞	∞	∞
$ub_0(1, x_4)$	1000	1000	1000
TH_0	1	1	1
$th_0(0, x_1)$	1	1	1
$th_0(0, x_4)$	0	0	0
$th_0(1, x_1)$	0	0	0
$th_0(1, x_4)$	1000	1000	1000
x_1			
d_1	0	0	0
CX_1	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
$cx_1(0, x_2)$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$	$\{(x_0, 0)\}$
LB_1	1	1	1
$lb_1(0, x_2)$	1	1	1
UB_1	1	1	1
$ub_1(0, x_2)$	1	1	1
TH_1	1	1	1
x_2			
d_2	1	1	1
CX_2	**$\{(x_0, 0), (x_1, 0)\}$**	$\{(x_0, 0), (x_1, 0)\}$	$\{(x_0, 0), (x_1, 0)\}$
$cx_2(0, x_3)$	**$\{\}$**	$\{\}$	$\{\}$
$cx_2(1, x_3)$	**$\{\}$**	$\{\}$	**$\{(x_0, 0)\}$**
LB_2	**0**	0	0
$lb_2(0, x_3)$	**0**	0	0
$lb_2(1, x_3)$	**0**	0	**11**
UB_2	**∞**	∞	**11**
$ub_2(0, x_3)$	**∞**	∞	∞
$ub_2(1, x_3)$	**∞**	∞	**11**
TH_2	100	100	**11**
x_3			
d_3	1	**0**	0
CX_3	$\{(x_0, 1), (x_2, 1)\}$	**$\{(x_0, 0), (x_2, 1)\}$**	$\{(x_0, 0), (x_2, 1)\}$
LB_3	100	**11**	11
UB_3	100	**11**	11
x_4			
d_4	0	0	0
CX_4	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$	$\{(x_0, 1)\}$
LB_4	1000	1000	1000
UB_4	1000	1000	1000

Table 24: States of agents in Steps 6' (1)–6' (3).

C Pseudocode of the amended version of ADOPT

Algorithms 1 and 2 show the pesudocodes of the amended version of ADOPT, described in Section 4. As described in Section 4, the amendment consists of three parts. The modification of the update rules for the lower and upper bounds for a child is in lines 32–33; the modification of initialization is in line 3; and the modifications of TERMINATE messages and their receiving procedure are in lines 43, 46–49, and 60.

Here, the predicate $\mathbf{Compatible}(CX, CX')$ means whether two contexts CX and CX' are compatible or not. This is represented as follows:

$$\mathbf{Compatible}(CX, CX') \equiv \neg\exists_{(x_i,d_i)\in CX,(x_j,d_j)\in CX'}(x_i = x_j \wedge d_i \neq d_j).$$

Algorithm 1 Amended version of ADOPT

1: **procedure Start**()
2: $TH_i := 0$;
3: $CX_i := \{(x_p, ValInit(x_p)) \mid x_p \in SCP(x_i)\}$;
4: **for all** $d \in D_i, x_c \in C(x_i)$ **do**
5: **InitChild**(d, x_c);
6: $d_i := \arg\min_{d\in D_i} LB_i(d)$;
7: **MaintainThresholdInvariant**();
8: **Backtrack**();

9: **procedure InitChild**(d, x_c)
10: $lb_i(d, x_c) := 0$;
11: $ub_i(d, x_c) := \infty$;
12: $th_i(d, x_c) := 0$;
13: $cx_i(d, x_c) := \varnothing$;

14: **procedure Received**(VALUE, x_p, d)
15: **if** TERMINATE not received from parent **then**
16: add (x_p, d) to CX_i (and remove the old assigment of x_p);
17: **for all** $d \in D_i, x_c \in C(x_i)$ **do**
18: **if** $\neg\mathbf{Compatible}(CX_i, cx_i(d, x_c))$ **then**
19: **InitChild**(d, x_c);
20: **MaintainThresholdInvariant**();
21: **Backtrack**();

22: **procedure Received**(COST, x_c, CX, lb, ub)
23: $d := d'_i$ s.t. $(x_i, d'_i) \in CX$;
24: remove (x_i, d) from CX;
25: **if** TERMINATE not received from parent **then**
26: **for all** $(x_j, d_j) \in CX$ and x_j is not x_i's neighbor **do**
27: add (x_j, d_j) to CX_i (and remove the old assigment of x_j);
28: **for all** $d' \in D_i, x'_c \in C(x_i)$ **do**
29: **if** $\neg\mathbf{Compatible}(CX_i, cx_i(d', x'_c))$ **then**
30: **InitChild**(d', x'_c);
31: **if Compatible**(CX, CX_i) **then**
32: $lb_i(d, x_c) := \max\{lb_i(d, x_c), lb\}$;
33: $ub_i(d, x_c) := \min\{ub_i(d, x_c), ub\}$;
34: $cx_i(d, x_c) := CX$;
35: **MaintainChildThresholdInvariant**();
36: **MaintainThresholdInvariant**();
37: **Backtrack**();

38: **procedure Received**(THRESHOLD, th, CX)
39: **if Compatible**(CX, CX_i) **then**
40: $TH_i := th$;
41: **MaintainThresholdInvariant**();
42: **Backtrack**();

43: **procedure Received**(TERMINATE, th, CX)
44: record TERMINATE received;
45: $CX_i := CX$
46: $TH_i := th$;
47: **for all** $d \in D_i, x_c \in C(x_i)$ **do**
48: **if** $\neg\mathbf{Compatible}(CX_i, cx_i(d, x_c))$ **then**
49: **InitChild**(d, x_c);
50: **Backtrack**();

Algorithm 2 Amended version of ADOPT (continued)

51: **procedure Backtrack**()
52: **if** $TH_i = UB_i$ **then**
53: $d_i := \arg\min_{d\in D_i} UB_i(d)$ (choose the previous d_i if possible);
54: **else if** $LB_i(d_i) > TH_i$ **then**
55: $d_i := \arg\min_{d\in D_i} LB_i(d)$;
56: **Send**(VALUE, x_i, d_i) to each $x_c \in CD(x_i)$;
57: **MaintainAllocationInvariant**();
58: **if** $TH_i = UB_i$ **then**
59: **if** TERMINATE received or x_i is root **then**
60: **Send**(TERMINATE, $th_i(d_i, x_c), CX_i \cup \{(x_i, d_i)\}$) to each $x_c \in C(x_i)$;
61: terminate execution;
62: **Send**(COST, x_i, CX_i, LB_i, UB_i) to $pa(x_i)$;

63: **procedure MaintainThresholdInvariant**()
64: **if** $TH_i < LB_i$ **then**
65: $TH_i := LB_i$;
66: **if** $TH_i > UB_i$ **then**
67: $TH_i := UB_i$;

68: **procedure MaintainAllocationInvariant**()
69: **while** $TH_i > \delta_i(d_i) + \sum_{x_c\in C(x_i)} th_i(d_i, x_c)$ **do**
70: choose $x_c \in C(x_i)$ where $ub_i(d_i, x_c) > th_i(d_i, x_c)$;
71: $th_i(d_i, x_c) := th_i(d_i, x_c) + 1$;
72: **while** $TH_i < \delta_i(d_i) + \sum_{x_c\in C(x_i)} th_i(d_i, x_c)$ **do**
73: choose $x_c \in C(x_i)$ where $lb_i(d_i, x_c) < th_i(d_i, x_c)$;
74: $th_i(d_i, x_c) := th_i(d_i, x_c) - 1$;
75: **Send**(THRESHOLD, $th_i(d_i, x_c), CX_i$) to each $x_c \in C(x_i)$;

76: **procedure MaintainChildThresholdInvariant**()
77: **for all** $d \in D_i, x_c \in C(x_i)$ **do**
78: **while** $lb_i(d, x_c) > th_i(d, x_c)$ **do**
79: $th_i(d, x_c) := th_i(d, x_c) + 1$;
80: **for all** $d \in D_i, x_c \in C(x_i)$ **do**
81: **while** $th_i(d, x_c) > ub_i(d, x_c)$ **do**
82: $th_i(d, x_c) := th_i(d, x_c) - 1$;

D Proof of Termination and Optimality for the Amended Version of ADOPT

We prove the termination and optimality of the amended version of ADOPT by two theorems, i.e., Theorem 1 for termination and Theorem 2 for optimality. The arguments of the proofs are based on the study of BnB-ADOPT [Yeoh *et al.*, 2010] with some modifications. First, we modify some statements and arguments because of the differences between the amended version of ADOPT and BnB-ADOPT, e.g., ID is not introduced in our version. Second, we add lemmata with regard to thresholds, which are crucial for the termination condition. Finally, we modify the statement of Theorem 2 to guarantee that all agents obtain the optimal values and costs at termination although the study of BnB-ADOPT only guarantees that the root agent obtains the optimal cost.

In the proofs, the notations concerning agents involved by agent $x_i \in X$ in a pseudo-tree are as follows: $CD(x_i) \subseteq X$ is a set of the children and pseudo-children of x_i; $C(x_i) \subseteq CD(x_i)$ is a set of the children of x_i; $pa(x_i) \in X$ is the parent of x_i; $P(x_i) \subseteq X$ is a set of the ancestors of x_i; $SCP(x_i) \subseteq P(x_i)$ is a set of the ancestors of x_i that are the parents or pseudo-parents of agents (including x_i) in the subtree rooted at x_i; $CP(x_i) \subseteq SCP(x_i)$ is a set of the ancestors of x_i that are the parent or pseudo-parents of x_i.

Furthermore, as shown in Section 3, we assume that message transfer is based on a cycle. ϵ is the largest duration of a cycle, i.e., the largest duration between receiving a message and sending new messages. Additionally, Δ is the largest duration between sending a message and receiving it.

Lemma 1. *If two contexts CX and CX' of any agent $x_i \in X$ contain the values of all agents $x_p \in SCP(x_i)$, and if these values coincide for each $x_p \in SCP(x_i)$, then $\gamma_i(CX) = \gamma_i(CX')$.*

Proof. By induction on the height (denoted by k) of the subtree rooted at x_i.

Base Case (for $k = 0$). Consider the case where x_i is a leaf agent. By definition,

$$\begin{aligned}\gamma_i(CX) &= \min_{d \in D_i} \gamma_i(d, CX) \\ &= \min_{d \in D_i} \delta_i(d, CX).\end{aligned}$$

Since $\delta_i(d, CX)$ is the local cost between x_i and its higher neighbors, $\delta_i(d, CX)$ is determined by d and the values of $x_p \in CP(x_i) \subseteq SCP(x_i)$ contained in CX. By assumption, given CX contains all the values of $x_p \in SCP(x_i)$,

$$\gamma_i(CX) = \min_{d \in D_i} \left(\sum_{x_p \in CP(x_i)} f_{i,p}(d, d_p) \right)$$

such that $(x_p, d_p) \in CX$. Similarly,

$$\gamma_i(CX') = \min_{d \in D_i} \left(\sum_{x_p \in CP(x_i)} f_{i,p}(d, d'_p) \right)$$

such that $(x_p, d'_p) \in CX'$. By assumption, the values of any $x_p \in SCP(x_i)$ in CX and CX' coincide. Thus, for any value $d \in D_i$,

$$\sum_{x_p \in CP(x_i)} f_{i,p}(d, d_p) = \sum_{x_p \in CP(x_i)} f_{i,p}(d, d'_p).$$

Therefore, $\gamma_i(CX) = \gamma_i(CX')$.

Indution Step (for $k > 0$). By definition and a similar argument as in the base case,

$$\begin{aligned}\gamma_i(CX) &= \min_{d \in D_i} \gamma_i(d, CX) \\ &= \min_{d \in D_i} \Bigg(\delta_i(d, CX) \\ &\quad + \sum_{x_c \in C(x_i)} \gamma_c\left(CX \cup \{(x_i, d)\}\right) \Bigg) \\ &= \min_{d \in D_i} \Bigg(\sum_{x_p \in CP(x_i)} f_{i,p}(d, d_p) \\ &\quad + \sum_{x_c \in C(x_i)} \gamma_c\left(CX \cup \{(x_i, d)\}\right) \Bigg)\end{aligned}$$

such that $(x_p, d_p) \in CX$. Similarly,

$$\begin{aligned}\gamma_i(CX') = \min_{d \in D_i} \Bigg(&\sum_{x_p \in CP(x_i)} f_{i,p}(d, d'_p) \\ &+ \sum_{x_c \in C(x_i)} \gamma_c\left(CX' \cup \{(x_i, d)\}\right) \Bigg)\end{aligned}$$

such that $(x_p, d'_p) \in CX'$. By assumption, the values contained in CX and CX' coincide for any $x_p \in SCP(x_i)$. Thus, for any value $d \in D_i$,

$$\sum_{x_p \in CP(x_i)} f_{i,p}(d, d_p) = \sum_{x_p \in CP(x_i)} f_{i,p}(d, d'_p).$$

Also, for any value $d \in D_i$ and any child $x_c \in C(x_i)$, both $CX \cup \{(x_i, d)\}$ and $CX' \cup \{(x_i, d)\}$ contain the value for any $x'_p \in SCP(x_c) \subseteq SCP(x_i) \cup \{x_i\}$, and $CX \cup \{(x_i, d)\}$ and $CX' \cup \{(x_i, d)\}$ coincide for the value of any $x'_p \in SCP(x_c)$. Thus, by the induction hypothesis,

$$\gamma_c\left(CX \cup \{(x_i, d)\}\right) = \gamma_c\left(CX' \cup \{(x_i, d)\}\right).$$

Therefore, $\gamma_i(CX) = \gamma_i(CX')$. □

Lemma 2. *If context CX_i does not change for any agent $x_i \in X$ from a time T to a time $t \geq T$, then for any time t' such that $T \leq t' \leq t$, lower bounds $lb_i(d, x_c)$, $LB_i(d)$, and LB_i and upper bounds $ub_i(d, x_c)$, $UB_i(d)$, and UB_i are monotonically non-decreasing and monotonically non-increasing, respectively, for any value $d \in D_i$ and any child $x_c \in C(x_i)$.*

Proof. Since CX_i does not change at any time t' such that $T \leq t' \leq t$, the value $\delta_i(d, CX_i)$ also does not change for any $d \in D_i$. In addition, for any $d \in D_i$ and any $x_c \in C(x_i)$, the lower and upper bounds are updated by the equations (1)–(6) after the initialization. Therefore, the lower and upper bounds are monotonically non-decreasing and monotonically non-increasing, respectively, for any $d \in D_i$ and any $x_c \in C(x_i)$. □

Definition 1. *The current context CX_i of agent x_i is correct iff CX_i contains the assignments of all agents in $SCP(x_i)$, and the values of all agents in CX_i are equal to the values of the agents, i.e.,*

$$\forall_{x_p \in SCP(x_i)} \left((x_p, d) \in CX_i\right) \\ \wedge \forall_{(x'_j, d'_j) \in CX_i} \exists_{x_j \in X} \left(x_j = x'_j \wedge d_j = d'_j\right).$$

Lemma 3. *If the value of any agent $x_p \in SCP(x_i)$ for any agent $x_i \in X$ does not change from a time T to a time t such that $T + |X| \cdot (\Delta + \epsilon) + \epsilon \leq t$, the value of x_p contained in the context of x_i is equal to the value taken by x_p between some time $t' \leq t$ and t.*

Proof.

Case 1. Consider the case where x_p is a parent or a pseudo-parent of x_i. Since the duration of one cycle is less than or equal to ϵ, x_p sends a VALUE message containing its own value to x_i at a time t'' such that $T \leq t'' \leq T + \epsilon$. Since the maximum delay for a message to arrive is Δ, x_i receives the VALUE message before time $t'' + \Delta$. After that, the value of x_p contained in the context of x_i is updated. Thus, there exists a time t' with $t'' \leq t' \leq t'' + \Delta \leq T + \Delta + \epsilon$ such that the value of x_p contained in the context of x_i are equal to the value taken by x_p.

Case 2. Consider the case where x_p is neither the parent nor the pseudo-parent of x_i. Since $x_p \in SCP(x_i)$, among the descendant agents of x_i, there exists an agent (say, x_c) which is a child or a pseudo-child of x_p. As in Case 1, x_p sends a VALUE message containing its own value to x_c at a time t'' such that $T \leq t'' \leq T + \epsilon$, and then x_c receives this message before time $t'' + \Delta$. After that, the value of x_p contained in the context of x_c is updated, and then x_c sends a COST message containing the updated context to $pa(x_c)$ before $t'' + \Delta + \epsilon$. Subsequently, $pa(x_c)$ receives the message before $t'' + 2\Delta + \epsilon$. After that, the value of x_p contained in the context of $pa(x_c)$ is updated, and then $pa(x_c)$ sends a COST message containing the updated context to $pa(pa(x_c))$ before $t'' + 2 \cdot (\Delta + \epsilon)$. A similar process continues until the value of x_p contained in the context of x_i is updated. Specifically, for the number (say, $k \leq |X|$) of chains of message passing, the context of x_i is updated by time $t'' + k \cdot \Delta + (k-1) \cdot \epsilon$. Therefore, from time t' to time t such that $t'' \leq t' \leq t'' + k \cdot \Delta + (k-1) \cdot \epsilon < T + |X| \cdot (\Delta + \epsilon) + \epsilon \leq t$, the value of x_p contained in the context of x_i is equal to the value taken by x_p. □

Corollary 1. *If the values of all agents $x_p \in SCP(x_i)$ for any agent $x_i \in X$ do not change between some time T and time t such that $T + |X| \cdot (\Delta + \epsilon) + \epsilon \leq t$, then the context of x_i is correct between some time $t' \leq t$ and t.*

Lemma 4. *For any agent $x_i \in X$, any value $d \in D_i$, and any child $x_c \in C(x_i)$, if $LB_c \leq \gamma_c(CX_c) \leq UB_c$ holds at any time, then $lb_i(d, x_c) \leq \gamma_c(CX_i \cup \{(x_i, d)\}) \leq ub_i(d, x_c)$ also holds at any time.*

Proof. By induction on the number (say, $k \geq 0$) of times that agent x_i changes its context or updates its bounds $lb_i(d, x_c)$ and $ub_i(d, x_c)$ for any $d \in D_i$ and any $x_c \in C(x_i)$ after x_i initializes the bounds.

Base Case (for $k = 0$). In this case, x_i initializes $lb_i(d, x_c)$ and $ub_i(d, x_c)$, where

$$\begin{aligned} lb_i(d, x_c) &= 0 \\ &\leq \gamma_c(CX_i \cup \{(x_i, d)\}) \\ &\leq \infty \\ &= ub_i(d, x_c). \end{aligned}$$

Induction Step (for $k > 0$). For the case where x_i changes the context or updates $lb_i(d, x_c)$ and $ub_i(d, x_c)$ for any $d \in D_i$ and any $x_c \in C(x_i)$ without initializing these bounds, there are the following two cases: one is when the context is changed, i.e., when x_i receives a VALUE or TERMINATE message, or receives a COST message and executes lines 25–27 of Algorithm 1; the other is when the bounds are updated, i.e., when x_i receives a COST message and executes lines 31–33 of Algorithm 1.

Case 1. Let us assume x_i receives either VALUE, COST, or TERMINATE message from x_c, and then changes the context from CX_i to $\hat{CX}_i$ without reinitializing the bounds. This case is divided into the following subcases (i) and (ii) depending on whether $cx_i(d, x_c) = \varnothing$ or not.

(i) For $cx_i(d, x_c) = \varnothing$. If $SCP(x_c) = \{x_i\}$, then $cx_i(d, x_c) = \varnothing$ always holds. In this case, however, the context of x_i does not change. Since the context of x_i is changed and $cx_i(d, x_c) = \varnothing$, $lb_i(d, x_c)$ and $ub_i(d, x_c)$ remain initialized. Therefore, as in the base case,

$$\begin{aligned} lb_i(d, x_c) &= 0 \\ &\leq \gamma_c(CX_i \cup \{(x_i, d)\}) \\ &\leq \infty \\ &= ub_i(d, x_c). \end{aligned}$$

(ii) For $cx_i(d, x_c) \neq \varnothing$. In this case, $cx_i(d, x_c)$ is compatible with CX_i and $\hat{CX}_i$. Moreover, by the initialization of the context of x_i, each CX_i and $\hat{CX}_i$ contains the value of all $x_p \in SCP(x_i)$. Similarly, since CX_c contains the value of all $x'_p \in SCP(x_c) \subseteq SCP(x_i) \cup \{x_i\}$, $cx_i(d, x_c)$ contains the value of all $x'_p \in SCP(x_c) \backslash \{x_i\}$. Therefore, each $CX_i \cup \{(x_i, d)\}$ and $\hat{CX}_i \cup \{(x_i, d)\}$

contains the value of all $x'_p \in SCP(x_c)$, where the values of any $x'_p \in SCP(x_c)$ contained in $CX_i \cup \{(x_i, d)\}$ and $\hat{CX}_i \cup \{(x_i, d)\}$ coincide. Thus, by Lemma 1 and the induction hypothesis,

$$\begin{aligned} lb_i(d, x_c) &\leq \gamma_c(CX_i \cup \{(x_i, d)\}) \\ &= \gamma_c(\hat{CX}_i \cup \{(x_i, d)\}) \\ ub_i(d, x_c) &\geq \gamma_c(CX_i \cup \{(x_i, d)\}) \\ &= \gamma_c(\hat{CX}_i \cup \{(x_i, d)\}). \end{aligned}$$

Case 2. Let us assume x_i receives a COST message from x_c, and then updates $lb_i(d, x_c)$ and $ub_i(d, x_c)$ to $\hat{lb}_i(d, x_c)$ and $\hat{ub}_i(d, x_c)$, respectively, without initialization. In this case, $CX_i \cup \{(x_i, d)\}$ is compatible with CX_c in the COST message. Moreover, by the initialization of the contexts of x_i and x_c, $CX_i \cup \{(x_i, d)\}$ and CX_c contain the value of any $x'_p \in SCP(x_c) \subseteq SCP(x_i) \cup \{x_i\}$, where the values of all $x'_p \in SCP(x_c)$ contained in $CX_i \cup \{(x_i, d)\}$ and CX_c coincide. Thus, by Lemma 1 and the induction hypothesis,

$$\begin{aligned} \hat{lb}_i(d, x_c) &= \max\{lb_i(d, x_c), LB_c\} \\ &\leq \max\{\gamma_c(CX_i \cup \{(x_i, d)\}), \\ &\quad \gamma_c(CX_c)\} \\ &= \max\{\gamma_c(CX_i \cup \{(x_i, d)\}), \\ &\quad \gamma_c(CX_i \cup \{(x_i, d)\})\} \\ &= \gamma_c(CX_i \cup \{(x_i, d)\}) \\ \hat{ub}_i(d, x_c) &= \min\{ub_i(d, x_c), UB_c\} \\ &\geq \min\{\gamma_c(CX_i \cup \{(x_i, d)\}), \\ &\quad \gamma_c(CX_c)\} \\ &= \min\{\gamma_c(CX_i \cup \{(x_i, d)\}), \\ &\quad \gamma_c(CX_i \cup \{(x_i, d)\})\} \\ &= \gamma_c(CX_i \cup \{(x_i, d)\}). \end{aligned}$$

Therefore, for any value $d \in D_i$ and any child $x_c \in C(x_i)$, $lb_i(d, x_c) \leq \gamma_c(CX_i \cup \{(x_i, d)\}) \leq ub_i(d, x_c)$ always holds. □

Lemma 5. *For any value $d \in D_i$ of any agent $x_i \in X$, $LB_i(d) \leq \gamma_i(d, CX_i) \leq UB_i(d)$ and $LB_i \leq \gamma_i(CX_i) \leq UB_i$ hold at any time.*

Proof. By induction on the height (say, k) of the subtree rooted at agent x_i.

Base Case (for $k = 0$). For the case where x_i is a leaf agent. For any value $d \in D_i$,

$$\begin{aligned} LB_i(d) &= \delta_i(d, CX_i) \\ &= \gamma_i(d, CX_i) \\ UB_i(d) &= \delta_i(d, CX_i) \\ &= \gamma_i(d, CX_i). \end{aligned}$$

Thus, for any value $d \in D_i$, $LB_i(d) \leq \gamma_i(d, CX_i) \leq UB_i(d)$ always holds. Also,

$$\begin{aligned} LB_i &= \min_{d \in D_i} LB_i(d) \\ &= \min_{d \in D_i} \gamma_i(d, CX_i) \\ &= \gamma_i(CX_i) \\ UB_i &= \min_{d \in D_i} UB_i(d) \\ &= \min_{d \in D_i} \gamma_i(d, CX_i) \\ &= \gamma_i(CX_i). \end{aligned}$$

Thus, $LB_i \leq \gamma_i(CX_i) \leq UB_i$ always holds.

Induction Step (for $k > 0$). By the induction hypothesis and Lemma 4, for any $d \in D_i$,

$$\begin{aligned} LB_i(d) &= \delta_i(d, CX_i) + \sum_{x_c \in C(x_i)} lb_i(d, x_c) \\ &\leq \delta_i(d, CX_i) + \sum_{x_c \in C(x_i)} \gamma_c(CX_i \cup \{(x_i, d)\}) \\ &= \gamma_i(d, CX_i) \\ UB_i(d) &= \delta_i(d, CX_i) + \sum_{x_c \in C(x_i)} ub_i(d, x_c) \\ &\geq \delta_i(d, CX_i) + \sum_{x_c \in C(x_i)} \gamma_c(CX_i \cup \{(x_i, d)\}) \\ &= \gamma_i(d, CX_i). \end{aligned}$$

Thus, for any $d \in D_i$, $LB_i(d) \leq \gamma_i(d, CX_i) \leq UB_i(d)$ always holds. Also,

$$\begin{aligned} LB_i &= \min_{d \in D_i} LB_i(d) \\ &\leq \min_{d \in D_i} \gamma_i(d, CX_i) \\ &= \gamma_i(CX_i) \\ UB_i &= \min_{d \in D_i} UB_i(d) \\ &\geq \min_{d \in D_i} \gamma_i(d, CX_i) \\ &= \gamma_i(CX_i). \end{aligned}$$

Thus, $LB_i \leq \gamma_i(CX_i) \leq UB_i$ always holds. □

Definition 5. *For agent $x_i \in X$, the potential of x_i is $\sum_{d \in D_i} (UB_i(d) - LB_i(d))$.*

Lemma 6. *If the context CX_i of any agent $x_i \in X$ does not change after a time t, then the potential of the agent is monotonically non-increasing and decreases by more than a positive constant every time the agent changes its value after t.*

Proof. Since CX_i does not change after t, by Lemma 2, the lower bound $LB_i(d)$ is monotonically non-decreasing, and the upper bound $UB_i(d)$ is monotonically non-increasing for all $d \in D_i$. Therefore, the potential of x_i is monotonically non-increasing. In addition, x_i changes the value d_i to a new value $\hat{d}_i$ only if $LB_i(\hat{d}_i) = \min_{d \in D_i} LB_i(d) < LB_i(d_i)$ or

$UB_i(\hat{d}_i) = \min_{d \in D_i} UB_i(d) < UB_i(d_i)$. Thus, between time t and the time $t' > t$ when d_i changes to $\hat{d}_i$, $LB_i(d_i)$ increases or $UB_i(\hat{d}_i)$ decreases. Therefore, the potential of x_i decreases by more than a positive constant after t. □

Lemma 7. *All agents change their values only a finite number of times.*

Proof. Assume that agent $x_i \in X$ infinitely changes its value. Since $SCP(x_r) = \varnothing$ holds for the root agent x_r and we can prove the lemma by the induction on the depth of x_i in the pseudo-tree based on the argument below, without loss of generality, we can assume that all agents $x_p \in SCP(x_i)$ change their values only a finite number of times. By this assumption, there exists a time t such that all $x_p \in SCP(x_i)$ do not change their values after t. Thus, by Corollary 1, there exists a time $t' \geq t$ such that the context CX_i of x_i is correct and the agent does not change the values in the context after t'. Furthermore, by Lemma 6, every time agent x_i changes its value afterwards, its potential decreases by more than a positive constant. By assumption, since x_i infinitely changes the value, the potential of x_i decreases towards $-\infty$. However, by Lemma 5, $LB_i(d) \leq UB_i(d)$ holds for all $d \in D_i$. Thus, the potential of x_i cannot become negative, which is a contradiction. Therefore, all agents change their values only a finite number of times. □

Lemma 8. *If all agents $x_i \in X$ does not change d_i, CX_i, $lb_i(d, x_c)$, $LB_i(d)$, LB_i, $ub_i(d, x_c)$, $UB_i(d)$, and UB_i for all values $d \in D_i$ and all children $x_c \in C(x_i)$ and CX_i is correct after some time T, then there exists a time $t \geq T$ such that TH_i and $th_i(d_i, x_c)$ do not change and $th_i(d_i, x_c) = TH_c$ holds for any x_i and any its child x_c after t.*

Proof. By assumption, TH_i and $th_i(d_i, x_c)$ may change only if $x_p = pa(x_i)$ sends a THRESHOLD or TERMINATE message containing $th_p(d_p, x_i) \neq TH_i$ to x_i and then x_i receives it. In the following, we use the induction on the depth of x_i in the pseudo-tree.

Base Case (for $k = 0$). For the case where x_i is the root agent. Since x_i does not receive a THRESHOLD or TERMINATE message, TH_i and $th_i(d_i, x_c)$ do not change for any $x_c \in C(x_i)$ after T. Let us consider the time (say, t') when a COST message sent from x_c after T is received. By the process conducted at t' and ChildThresholdInvariant, we have

$$\begin{aligned} th_i(d_i, x_c) &\geq lb_i(d_i, x_c) \\ &= \max\{lb'_i(d_i, x_c), LB_c\} \\ &\geq LB_c, \\ th_i(d_i, x_c) &\leq ub_i(d_i, x_c) \\ &= \min\{ub'_i(d_i, x_c), UB_c\} \\ &\leq UB_c, \end{aligned}$$

where $lb'_i(d_i, x_c)$ and $ub'_i(d_i)$ are the previous bounds. Thus, $th_i(d_i, x_c)$ satisfies the ThresholdInvariant of x_c, i.e., $LB_c \leq th_i(d_i, x_c) \leq UB_c$, after t'. Additionally, by the assumption, CX_i and CX_c are compatible. Therefore, at a time $t \geq t'$ when any $x_c \in C(x_i)$ receives the THRESHOLD message sent from x_i after t', x_c updates TH_c to $th_i(d_i, x_c)$ in the message. Thus, $th_i(d_i, x_c) = TH_c$ at t. Here, after t, TH_c changes only if x_c receives a THRESHOLD or TERMINATE message with $th_i(d_i, x_c) \neq TH_c$. Since $th_i(d_i, x_c)$ does not change after t, TH_c also does not change. Therefore, $th_i(d_i, x_c) = TH_c$ holds after t.

Induction Step (for $k > 0$). By the induction hypothesis, after some time $t' \geq T$, $x_p = pa(x_i)$ does not change $th_p(d_p, x_i)$ and $th_p(d_p, x_i) = TH_i$ holds. By the same argument as in the base case, there exists a time $t \geq t'$ such that TH_i and $th_i(d_i, x_c)$ do not change and $th_i(d_i, x_c) = TH_c$ holds after t. □

Lemma 9. *There exists a time t such that $TH_i = UB_i$ holds for all agents x_i after t.*

Proof. By Lemma 7, there exists a time t_0 such that any agent x_i does not change its value d_i after t_0. Thus, by Corollary 1, there exists a time $t_1 \geq t_0$ such that the context CX_i of any x_i is correct and does not changed after t_1. Also, by the following reasons (1)–(3), there exists a time $t_2 \geq t_1$ such that for any x_i and for all its values $d \in D_i$ and all its children $x_c \in C(x_i)$, $lb_i(d, x_c), LB_i(d), LB_i, ub_i(d, x_c), UB_i(d)$, and UB_i do not change after t_2.

(1) By Lemma 2, the lower bounds $lb_i(d, x_c), LB_i(d)$, and LB_i are monotonically non-decreasing, and the upper bounds $ub_i(d, x_c), UB_i(d)$, and UB_i are monotonically non-increasing.

(2) By Lemma 5, $LB_i(d) \leq \gamma_i(d, CX_i) \leq UB_i(d) \wedge LB_i \leq \gamma_i(CX_i) \leq UB_i$.

(3) By Lemma 4, $lb_i(d, x_c) \leq ub_i(d, x_c)$.

Therefore, by Lemma 8, there exists a time $t_3 \geq t_2$ such that for any x_i and any $x_c \in C(x_i)$, TH_i and $th_i(d_i, x_c)$ do not change and $th_i(d_i, x_c) = TH_c$ holds after t_3. Let us consider the time $t_4 \geq t_3$ when the first COST messages sent to all agents after t_3 are received and processed. In the following, we use the induction on the height (say, k) of the subtree rooted at agent x_i.

Base Case (for $k = 0$). For the case where x_i is a leaf agent. By definition,

$$\begin{aligned} LB_i &= \min_{d \in D_i} LB_i(d) = \min_{d \in D_i} \delta_i(d, CX_i) \\ UB_i &= \min_{d \in D_i} UB_i(d) = \min_{d \in D_i} \delta_i(d, CX_i). \end{aligned}$$

Thus, $LB_i = UB_i$. Also, by ThresholdInvariant, $LB_i \leq TH_i \leq UB_i$. Therefore, $TH_i = UB_i$ always holds.

Induction Step. By the induction hypothesis, we have the

following:

$$\begin{aligned}
UB_i &= \min_{d \in D_i} UB_i(d) \\
&\leq UB_i(d_i) \\
&= \delta_i(d_i, CX_i) + \sum_{x_c \in C(x_i)} ub_i(d_i, x_c) \\
&= \delta_i(d_i, CX_i) + \sum_{x_c \in C(x_i)} (\min\{ub'_i(d_i, x_c), \\
&\quad UB_c\}) \\
&= \delta_i(d_i, CX_i) + \sum_{x_c \in C(x_i)} (\min\{ub'_i(d_i, x_c), \\
&\quad TH_c\}) \\
&= \delta_i(d_i, CX_i) + \sum_{x_c \in C(x_i)} (\min\{ub'_i(d_i, x_c), \\
&\quad th_i(d_i, x_c)\}) \\
&\leq \delta_i(d_i, CX_i) + \sum_{x_c \in C(x_i)} th_i(d_i, x_c) \\
&= TH_i,
\end{aligned}$$

where $ub'_i(d_i, x_c)$ is the previous upper bound. Also, by ThresholdInvariant, $TH_i \leq UB_i$ holds. Thus, $TH_i = UB_i$ holds at t_4. Since TH_i and UB_i do not change after t_4, $TH_i = UB_i$ holds after t_4. □

Theorem 1. *The amended version of ADOPT terminates after a finite amount of time.*

Proof. By Lemma 9, there exists a time t such that $TH_i = UB_i$ holds for all agents x_i after t. If $TH_i = UB_i$ holds at the root agent x_r, then the agent sends TERMINATE messages to all $x_c \in C(x_r)$ and then terminates. These TERMINATE messages arrive at all x_c in a finite time. At that time, since $TH_c = UB_c$ also holds at all x_c, the agent sends TERMINATE messages to all $x'_c \in C(x_c)$ and then terminates. By the same process as above, all agents terminate after a finite amount of time. □

Lemma 10. *$LB_r = TH_r$ always holds at the root agent x_r.*

Proof. Since x_r does not receive THRESHOLD and TERMINATE messages, TH_r changes so as to satisfy ThresholdInvariant only if LB_r or UB_r changes when the agent receives a COST message. Furthermore, for any $d \in D_r$ and any $x_c \in C(x_c)$, $cx_r(d, x_c) = \varnothing$ always holds. Since $cx_r(d, x_c)$ is compatible with any context, the bounds $lb_r(d, x_c)$ and $ub_r(d, x_c)$ are initialized only at the beginning of the algorithm execution. In the following, we use the induction on the number (say, $k \geq 0$) of changes of LB_r and UB_r after x_r initializes $lb_r(d, x_c)$ and $ub_r(d, x_c)$.

Base Case (for $k = 0$). In this case, $lb_r(d, x_c)$ and $ub_r(d, x_c)$ are initialized for all $d \in D_r$ and all $x_c \in C(x_r)$. Thus, $lb_r(d, x_c) = 0$. Also, since $CX_r = \varnothing$, $\delta_r(d, CX_r) = 0$. Therefore,

$$\begin{aligned}
LB_r &= \min_{d \in D_r} LB_r(d) \\
&= \min_{d \in D_r} \left(\delta_r(d, CX_r) + \sum_{x_c \in C(x_r)} lb_r(d, x_c) \right) \\
&= 0.
\end{aligned}$$

Furthermore, $TH_r = 0$ holds when x_r initializes $lb_r(d, x_c)$ and $ub_r(d, x_c)$ at the beginning of the algorithm execution. Therefore, $LB_r = TH_r$.

Induction Step (for $k > 0$). Since $CX_r = \varnothing$ always holds at x_r, by Lemma 2, LB_r and UB_r are always monotonically non-decreasing and monotonically non-increasing, respectively. Thus, the following two cases should be considered.

Case 1. For the case where LB_r increases. By the procedure MaintainThresholdInvariant, TH_r is updated as $TH_r := LB_r$.

Case 2. For the case where UB_r decreases. By the induction hypothesis and Lemma 5, $LB_r = TH_r \leq UB_r$. Thus, TH_r does not change.

Therefore, $LB_r = TH_r$ always holds. □

Theorem 2. *For any agent $x_i \in X$ when the amended version of ADOPT terminates, the current context CX_i is correct, and $TH_i = \gamma_i(d_i, CX_i) = \gamma_i(CX_i)$.*

Proof. By Theorem 1, the amended version of ADOPT terminates after a finite amount of time, and then $TH_i = UB_i$ holds for any $x_i \in X$. At that time, by $d_i = \arg\min_{d \in D_i} UB_i(d)$, $TH_i = UB_i(d_i)$ also holds. Moreover, by definition and Lemma 5,

$$\begin{aligned}
\gamma_i(CX_i) &= \min_{d \in D_i} \gamma_i(d, CX_i) \\
&\leq \gamma_i(d_i, CX_i) \\
&\leq UB_i(d_i) \\
&= TH_i.
\end{aligned}$$

Therefore, if $TH_i = \gamma_i(CX_i)$, then $\gamma_i(d_i, CX_i) = \gamma_i(CX_i)$. In the following, we use the induction on the depth (say, k) of agent x_i in the pseudo-tree.

Base Case (for $k = 0$). In this case, since $SCP(x_i) = \varnothing$ holds and $CX_i = \varnothing$ always holds, CX_i is correct. Furthermore, by Lemma 10, $TH_i = LB_i$. Thus, when the amended version of ADOPT terminates,

$$LB_i = TH_i = UB_i.$$

By Lemma 5, $LB_i \leq \gamma_i(CX_i) \leq UB_i$. Thus, $TH_i = \gamma_i(CX_i)$ holds.

Induction Step (for $k > 0$). By the induction hypothesis, for $x_p = pa(x_i)$, $TH_p = \gamma_p(d_p, CX_p)$. Moreover,

since $TH_p = UB_p(d_p)$, we have the following:

$$\begin{aligned}
&\delta_p(d_p, CX_p) + \sum_{x_i \in C(x_p)} th_p(d_p, x_i) \\
=&\delta_p(d_p, CX_p) \\
&+ \sum_{x_i \in C(x_p)} \gamma_i \left(CX_p \cup \{(x_p, d_p)\}\right) \\
=&\delta_p(d_p, CX_p) + \sum_{x_i \in C(x_p)} ub_p(d_p, x_i).
\end{aligned}$$

Furthermore, by Lemma 4 and Lemma 5, and ChildThresholdInvariant, for all $x_i \in C(x_p)$,

$$\begin{gathered}
\gamma_i \left(CX_p \cup \{(x_p, d_p)\}\right) \leq ub_p(d_p, x_i), \\
th_p(d_p, x_i) \leq ub_p(d_p, x_i).
\end{gathered}$$

Thus, for all $x_i \in C(x_p)$,

$$th_p(d_p, x_i) = \gamma_i \left(CX_p \cup \{(x_p, d_p)\}\right) = ub_p(d_p, x_i). \tag{7}$$

When x_p terminates at a time t, the agent sends TERMINATE messages to all $x_i \in C(x_p)$. At a time $t' > t$, if any x_i receives the TERMINATE message, then the agent updates the context CX_i as $CX_p \cup \{(x_p, d_p)\}$, which is contained in the TERMINATE message. By the induction hypothesis, CX_p is correct at t. Also, when x_p terminates, since all $x'_p \in P(x_p)$ have already terminated, the values of all x'_p do not change after t, and thus CX_p is correct after t. Moreover, since the value of x_p does not change after t and $SCP(x_i) \subseteq SCP(x_p) \cup \{x_p\}$, and CX_i does not change after a TERMINATE message is received, CX_i is correct after t'. Furthermore, when x_i receives the TERMINATE message from x_p, TH_i is updated. Thus, $TH_i = th_p(d_p, x_i)$. By formula (7) and Lemma 1, at t',

$$TH_i = \gamma_i \left(CX_p \cup \{(x_p, d_p)\}\right) = \gamma_i(CX_i).$$

At that time, x_p terminates. Thus, since x_i does not receive any THRESHOLD message and TERMINATE message after t', TH_i may change only if LB_i, UB_i change. By Lemma 5,

$$LB_i \leq TH_i = \gamma_i(CX_i) \leq UB_i.$$

Thus, since ThresholdInvariant is always satisfied after t', TH_i does not change. Therefore, at termination, CX_i is correct and

$$TH_i = \gamma_i(CX_i) = \gamma_i(d_i, CX_i). \qquad \square$$