

# Supplementary Material for Paper #1084

## A AsymDPOP

### A.1 Pseudo Code for AsymDPOP

Fig 1. gives the sketch of AsymDPOP, and the execution process can be divided into two phases: utility propagation phase and value propagation phase. The utility propagation phase begins with leaf agents sending their utility tables to their parents via UTIL messages (line 2-3). When an agent  $a_i$  receives a UTIL message from its child  $a_c$ , it joins its private functions w.r.t. its (pseudo) children in branch  $a_c$  (line 5-6), and eliminates all the belonging variables whose highest (pseudo) parent is  $a_i$  from the utility table (line 7). Here,  $EV(a_i, a_c)$  is given by

$$EV(a_i, a_c) = PC(a_i) \cap Desc(a_c) \cup \{a_c\} \setminus ID(a_i)$$

where  $ID(a_i)$  is the set of  $a_i$ 's interface descendants which are constrained with  $Sep(a_i)$ . After receiving all the UTIL messages from its children,  $a_i$  propagates the joint utility table to its parent if it is not the root agent. Otherwise, the value propagation phase starts.

The value propagation phase begins with the root agent selecting the optimal assignment for itself (line 10). Given the determined assignments either from its parent (line 14) or computed locally (line 11), agent  $a_i$  selects the optimal assignments for the eliminated variables in each branch  $a_c \in C(a_i)$  by a joint optimization over them (line 15-18), and propagates the assignments together with the determined assignments to  $a_c$  (line 19-20). The algorithm terminates when each leaf agent receives a VALUE message.

### A.2 An Example for AsymDPOP

Fig.2 gives a pseudo tree. For better understanding, we take  $a_2$  to explain the concepts in a pseudo tree. Since  $a_1$  is the only ancestor constrained with  $a_2$  via a tree edge, we have  $P(a_2) = \{a_1\}$ ,  $PP(a_2) = \emptyset$  and  $Sep(a_2) = \{a_1\}$ . Similarly, since  $a_3$  and  $a_4$  are the descendants constrained with  $a_2$  via tree edge, we have  $C(a_2) = \{a_3, a_4\}$ ,  $PC(a_2) = \emptyset$ ,  $Desc(a_2) = \{a_3, a_4\}$ . Particularly, since  $a_4 \in Desc(a_2)$  is constrained with  $a_1 \in Sep(a_2)$ , we have  $ID(a_2) = \{a_4\}$ .

Since  $a_3$  and  $a_4$  are the leaf nodes, they send UTIL messages with their utility tables  $util_3$  and  $util_4$  to their parent  $a_2$ , where  $util_3 = f_{32}$  and  $util_4 = f_{41} \otimes f_{42}$ .

When receiving a UTIL message from  $a_3$  (assume  $a_3$ 's message has arrived earlier),  $a_2$  stores the received message

Algorithm 1: AsymDPOP for  $a_i$

---

```

When Initialization:
1   $util_i \leftarrow \bigotimes_{a_j \in AP(a_i)} f_{ij}$ 
2  if  $a_i$  is a leaf then
3    | send UTIL( $util_i$ ) to  $P(a_i)$ 
When received UTIL( $util_c$ ) from  $a_c \in C(a_i)$ :
4   $util_i^c \leftarrow util_c$ 
5  foreach  $a_j \in (PC(a_i) \cap Desc(a_c)) \cup \{a_c\}$  do
6    |  $util_i^c \leftarrow util_i^c \otimes f_{ij}$ 
7   $util_i \leftarrow util_i \otimes \min_{EV(a_i, a_c)} util_i^c$ 
8  if  $a_i$  have received all UTIL from  $C(a_i)$  then
9    if  $a_i$  is the root then
10   |  $v_i^* \leftarrow \operatorname{argmin}_{x_i} util_i$ 
11   | PropagateValue( $\{(x_i = v_i^*)\}$ )
12   else
13   | send UTIL( $util_i$ ) to  $P(a_i)$ 
When received VALUE( $Assign$ ) from  $P(a_i)$ :
14 PropagateValue( $Assign$ )
Function PropagateValue( $Assign$ ):
15 foreach  $a_c \in C(a_i)$  do
16    $Assign_i^c \leftarrow Assign$ 
17   if  $EV(a_i, a_c) \neq \emptyset$  then
18     |  $V^* \leftarrow \operatorname{argmin}_{EV(a_i, a_c)} util_i^c(Assign_{|dims(util_i^c)|})$ 
19     |  $Assign_i^c \leftarrow Assign_i^c \cup \{(x_j = V_{[x_j]}^*) | \forall x_j \in EV(a_i, a_c)\}$ 
20   | send VALUE( $Assign_i^c$ ) to  $a_c$ 

```

---

Figure 1: The sketch of AsymDPOP

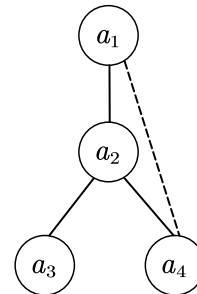


Figure 2: A pseudo-tree

from  $a_3$  (i.e.,  $util_2^3 = util_3 = f_{32}$ ), and then joins its private function  $f_{23}$  to update  $util_2^3$  (i.e.,  $util_2^3 = f_{32} \otimes f_{23}$ ). According to the definition of  $EV(a_i, a_c)$  that all the belonging variables' highest (pseudo) parent is  $a_i$  in branch  $a_c$ . Here,  $EV(a_2, a_3)$  is given by

$$EV(a_2, a_3) = PC(a_2) \cap Desc(a_3) \cup \{a_3\} \setminus ID(a_2) \\ = \emptyset \cap \emptyset \cup \{a_3\} \setminus \{a_4\} = \{a_3\}$$

Thus,  $a_2$  eliminates variable  $x_3$  from  $util_2^3$ . After that,  $a_2$  joins the eliminated result to the joint utility table  $util_2$ . Similarly, upon receipt of the UTIL message from  $a_4$ ,  $a_2$  saves the content as  $util_2^4$  (i.e.,  $util_2^4 = util_4 = f_{41} \otimes f_{42}$ ), and updates it with joining the private function  $f_{24}$  (i.e.,  $util_2^4 = f_{41} \otimes f_{42} \otimes f_{24}$ ). Since  $EV(a_2, a_4) = \emptyset$ ,  $a_2$  joins  $util_2^4$  to the  $util_2$  directly without any elimination operation. Since  $a_2$  have received all the UTIL messages from its children, it propagates  $util_2$  to its parent  $a_1$ . Here, we have

$$util_2 = f_{21} \otimes (f_{41} \otimes f_{42} \otimes f_{24}) \otimes (\min_{x_3} f_{23} \otimes f_{32})$$

When receiving the UTIL message from  $a_2$ ,  $a_1$  saves the content as  $util_1^2$  (i.e.,  $util_1^2 = util_2$ ) and joins the private functions  $f_{12}$  and  $f_{14}$  to update  $util_1^2$  (i.e.,  $util_1^2 = util_1^2 \otimes (f_{12} \otimes f_{14})$ ). After eliminating  $x_2$  and  $x_4$  ( $EV(a_1, a_2) = \{a_2, a_4\}$ ),  $a_1$  joins the eliminated result into  $util_1$ . Here, we have

$$util_1 = \min_{x_2, x_4} f_{12} \otimes f_{14} \otimes (f_{21} \otimes (f_{41} \otimes f_{42} \otimes f_{24}) \otimes (\min_{x_3} f_{23} \otimes f_{32}))$$

Since  $a_1$  is the root agent and receives all the UTIL messages from its children, it selects the optimal assignment  $v_1^*$  for itself and the optimal assignments  $v_2^*$  and  $v_4^*$  for the eliminated variables  $x_2$  and  $x_4$ . That is,

$$v_1^* = \underset{x_1}{\operatorname{argmin}} util_1, \\ (v_2^*, v_4^*) = \underset{x_2, x_4}{\operatorname{argmin}} util_1^2(x_1 = v_1^*)$$

Then  $a_1$  propagates a VALUE message including the optimal assignment to its child  $a_2$ , where the optimal assignment is  $\{(x_1 = v_1^*), (x_2 = v_2^*), (x_3 = v_3^*)\}$ .

When receiving the VALUE message from  $a_1$ ,  $a_2$  assigns the value  $v_2^*$  for itself. Since  $EV(a_2, a_3) = \{a_3\}$  and  $EV(a_2, a_4) = \emptyset$ ,  $a_2$  selects the optimal assignment for  $a_3$  by performing optimization over  $util_2^3$  with the determined assignment of  $x_2$ , that is

$$v_3^* = \underset{x_3}{\operatorname{argmin}} (util_2^3(x_2 = v_2^*))$$

Then  $a_2$  propagates the optimal assignment ( $x_3 = v_3^*$ ) and ( $x_4 = v_4^*$ ) to  $a_3$  and  $a_4$ , respectively. Once  $a_3$  and  $a_4$  receives the VALUE messages, they assign for themselves. Since all leaf agents receives VALUE messages, the algorithm terminates.

## B AsymDPOP with MBPS and MBES

### B.1 Pseudo Code for AsymDPOP with MBPS and MBES

Fig.3 gives the sketch of AsymDPOP with MBPS and MBSE. Like Algorithm 1, the execution process consists of two

#### Algorithm 2: AsymDPOP(MBPS+MBES) for $a_i$

```

When Initialization:
1   $util_i \leftarrow \emptyset$ 
2  if  $a_i$  is a leaf then
3    send UTIL(PartitionF()) to  $P(a_i)$ 
When received UTIL( $util_c$ ) from  $a_c \in C(a_i)$ :
4   $util_i^c \leftarrow util_c$ 
5  foreach  $a_j \in (PC(a_i) \cap Desc(a_c)) \cup \{a_c\}$  do
6    if  $\exists u \in util_i^c, s.t. dims(f_{ij}) \subset dims(u)$  then
7       $u \leftarrow u \otimes f_{ij}$ 
8    break
9   $util_i \leftarrow util_i \cup \text{Min}(util_i^c, EV(a_i, a_c))$ 
10 if  $a_i$  have received all UTIL from  $C(a_i)$  then
11   if  $a_i$  is the root then
12      $v_i^* \leftarrow \underset{x_i}{\operatorname{argmin}} (\otimes_{u \in util_i} u)$ 
13     PropagateValue( $\{(x_i = v_i^*)\}$ )
14   else
15      $util_i \leftarrow util_i \cup \text{PartitionF}()$ 
16     foreach  $u \in util_i$  do
17       if  $\exists u' \in util_i \setminus \{u\}, s.t. dims(u') \subset dims(u)$  then
18          $u \leftarrow u \otimes u'$ 
19        $util_i \leftarrow util_i \setminus \{u'\}$ 
20     send UTIL( $util_i$ ) to  $P(a_i)$ 
When received VALUE(Assign) from  $P(a_i)$ :
21 PropagateValue(Assign)
Function PropagateValue(Assign):
22 foreach  $a_c \in C(a_i)$  do
23    $Assign_i^c \leftarrow Assign$ 
24   if  $EV(a_i, a_c) \neq \emptyset$  then
25      $u' \leftarrow \otimes_{u \in util_i^c} u(Assign_{[dims(u)]})$ 
26      $V^* \leftarrow \underset{EV(a_i, a_c)}{\operatorname{argmin}} (u')$ 
27      $Assign_i^c \leftarrow Assign_i^c \cup \{(x_j = V_{[x_j]}^*) | \forall x_j \in EV(a_i, a_c)\}$ 
28   send VALUE( $Assign_i^c$ ) to  $a_c$ 
Function Min( $util, EV$ ):
29  $U \leftarrow util, G \leftarrow \text{GroupEV}(EV)$ 
30  $EVSet \leftarrow \bigcup_{\forall EV' \in G} \text{PartitionEV}(EV', U)$ 
31 foreach  $EV' \in EVSet$  do
32    $F \leftarrow \{f | \forall f \in U, dims(f) \cap EV' \neq \emptyset\}$ 
33    $U \leftarrow (U \setminus F) \cup \{\min_{EV'} \otimes_{f \in F} f\}$ 
34 return  $U$ 
Function PartitionF():
35  $u \leftarrow \emptyset, U \leftarrow \emptyset$ 
36 order  $AP(a_i)$  according to their levels
37 foreach  $a_j \in AP(a_i)$  do
38   if  $\exists u' \in util_i, s.t. dims(f_{ij}) \subset dims(u')$  then
39      $u' \leftarrow u' \otimes f_{ij}$ 
40   else
41     if  $|dims(u)| \geq k_p$  then
42        $U \leftarrow U \cup \{u\}$ 
43        $u \leftarrow f_{ij}$ 
44     else
45        $u \leftarrow u \otimes f_{ij}$ 
46 if  $u \neq \emptyset$  then
47   random select an element  $u' \in U$ 
48    $u' \leftarrow u' \otimes u$ 
49 return  $U$ 
Function GroupEV( $EV, util$ ):
50  $Dims \leftarrow \{dims(u) \cap EV | \forall u \in util\}$ 
51 while  $\exists D, D' \in Dims, s.t. D \cap D' \neq \emptyset$  do
52    $D \leftarrow D \cup D', Dims \leftarrow Dims \setminus D'$ 
53 return  $Dims$ 
Function PartitionEV( $EV$ ):
54  $EV' \leftarrow \emptyset, EVSet \leftarrow \emptyset$ 
55 foreach  $ev \in EV$  do
56   if  $|EV'| \geq k_e$  then
57      $EVSet \leftarrow EVSet \cup \{EV'\}$ 
58      $EV' \leftarrow \{ev\}$ 
59   else
60      $EV' \leftarrow EV' \cup \{ev\}$ 
61 if  $EV' \neq \emptyset$  then
62   random select an element  $EV'' \in EVSet$ 
63    $EV'' \leftarrow EV'' \cup EV'$ 
64 return  $EVSet$ 

```

Figure 3: The sketch of AsymDPOP with MBPS and MBES

phases: utility propagation phase and value propagation phase. The utility propagation phase also begins with leaf agents sending a set of utility tables to their parents via UTIL messages (line 2-3). The set of utility tables obtain from the Function **PartitionF**. In this function, agent  $a_i$  divides its private functions which are constrained with  $AP(a_i)$  into several utility tables, and joins the private functions in the same utility table into one utility table (line 37-45). If there is any residue,  $a_i$  joins them into an arbitrary utility table (line 46-48). Note that  $k_p$  is a parameter which controls the minimal number of dimensions of each utility table.

When receiving a UTIL message from its child  $a_c$ ,  $a_i$  joins its private functions w.r.t. its (pseudo) children in branch  $a_c$  (line 6-8). Notice that the join operation of its private functions w.r.t. its children does not increase the number of dimensions and should be applied accordingly to the related utility tables. When performing eliminations,  $a_i$  uses Function **Min** to implement (line 9,29-34). To be more specific, instead of eliminating all the variables in  $EV(a_i, a_c)$  directly,  $a_i$  first divides elimination variables into several groups whose variables share at least a common utility table in Function **GroupEV** (line 29). And then, for each variable group,  $a_i$  divides the variables into several sets (or batches) with the Function **PartitionEV** in the similar way as Function **PartitionF** (line 30). Specifically,  $k_e$  in **PartitionEV** specifies the minimal number of variables optimized in a min operator (i.e., the size of a batch). After getting elimination variable set ( $EVSet$ ),  $a_i$  traverses the set to optimize  $util$  that has been passed to Function **Min** (line 31). In detail, for each batch, we perform optimization to the utility tables that are related to the variables in the batch over the batch and replace these utility tables with the results. The process terminates when the all the variable batches are exhausted (line 32-33). Subsequently,  $a_i$  returns the new utility table set which has been eliminated properly (line 34).

When receiving all the UTIL messages from its children,  $a_i$  propagates the joint utility table set to its parent if it is not the root agent (line 14-20). Notice that, the size of the joint utility table set is reduced by the join operation of the utility tables that does not increase the number of dimensions and should be applied accordingly to the related utility tables (15-19). Otherwise, the value propagation phase starts (line 13). The process is roughly as the same as Algorithm 1. Agent  $a_i$  selects the optimal assignments for itself and the eliminated variables in each branch  $a_c \in C(a_i)$ . And since it uses the MBPS, there may be not one joint utility table (in Algorithm 1) but a set of joint utility tables with the smaller size. So before selecting the optimal assignments for branch  $a_c$ ,  $a_i$  joins all the tables in  $util_i^c$  with the assignments received from its parent (line 21) or computed locally (line 13). After selecting the assignments for variables of  $EV(a_i, a_c)$  in branch  $a_c$  (line 26-27), it sends a VALUE message to  $a_c$  (line 28). The algorithm terminates when each leaf agent receives a VALUE message.

## B.2 An Example of AsymDPOP with MBPS and MBES

For the convenience of further explanation, we denote the joint utility table  $u_{ijk}$  as a function of three variables  $x_i, x_j$

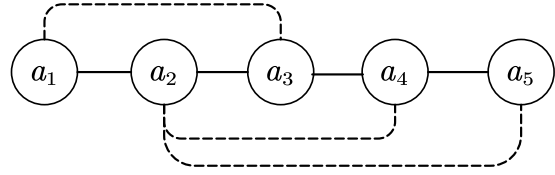


Figure 4: A chain-like pseudo tree

and  $x_k$ . And the index of the function is sorted in alphabetical order.

We take Fig.4 as an example to demonstrate the Algorithm 2. Suppose that  $k_p = 3$  and  $k_e = 1$ . Since  $a_5$  is the leaf node, it sends a UTIL message including a set of utility tables obtained from Function **PartitionF** to its parent  $a_4$ . In **PartitionF**,  $a_5$  first orders  $AP(a_5)$  according to their levels. And  $a_5$  divides its private functions which are related with  $AP(a_5)$  into the utility tables. Since  $|AP(a_5)| = 2$ , the biggest dimension of the joint utility table is three which satisfies  $k_p$ . And thus there is only one utility table. Then,  $a_5$  joins the private functions in the utility table to get a 3-ary utility table  $u_{541}$  (i.e.,  $u_{541} = f_{54} \otimes f_{51}$ ), and sends the set including the joint utility table  $u_{541}$  to its parent  $a_4$ .

When  $a_4$  receives the UTIL message from its child  $a_5$ , it saves the utility table set and updates the element  $u_{541}$  by joining its private function  $f_{45}$  (i.e.,  $u_{541} = u_{541} \otimes f_{45}$ ). And since  $a_4$  is not the highest (pseudo) parent of any descendants ( $EV(a_4, a_5) = \emptyset$ ), there is no elimination operation. Next, since  $a_4$  has received all the UTIL messages from its children, it sends a set containing both  $u_{432}$  and  $u_{541}$  to its parent  $a_3$ . Here,  $u_{432}$  is given by joining the private functions  $f_{43}$  and  $f_{42}$ .

When receiving the UTIL message from  $a_4$ , similarly,  $a_3$  saves the utility table set and updates  $u_{432}$  by joining its private function  $f_{34}$  (i.e.,  $u_{432} = u_{432} \otimes f_{34}$ ). Since  $a_3$  is not the highest (pseudo) parent of  $a_4$  or any other descendants, there is no elimination at the agent. Since the residual private functions  $f_{31}$  and  $f_{32}$  could not join into  $u_{432}$  without increase the number of dimensions,  $a_3$  deals with  $f_{31}$  and  $f_{32}$  with Function **PartitionF** by joining them into a 3-ary utility table  $u_{321}$ . And it sends the utility table set  $\{u_{541}, u_{432}, u_{321}\}$  to its parent  $a_2$ .

Once  $a_2$  receives the UTIL message from  $a_3$ , it also saves the utility table set firstly, and joins the relative private function  $f_{23}$  into  $u_{321}$  (i.e.,  $u_{321} = u_{321} \otimes f_{23}$ ). Since  $a_2$  is the highest (pseudo) parent of  $a_4$  and  $a_5$ , that is  $EV(a_2, a_3) = \{a_4, a_5\}$ , the eliminations are preformed by Function **Min**. Firstly,  $a_2$  groups variables  $x_4$  and  $x_5$  into one group, as they share a common utility table  $f_{451}$ . Since  $k_e = 1$ , variables  $x_4$  and  $x_5$  are separated in two batches, and they are eliminated from the utility functions in the set  $\{u_{541}, u_{432}, u_{321}\}$  one by one. After eliminating  $x_5$  (assume  $x_5$  is the first eliminate variable), we get a new utility table set  $\{u_{41}, u_{432}, u_{321}\}$ . Then we derive  $u'_{321}$  by eliminating  $x_4$  from the utility tables (i.e.,  $u'_{321} = \min_{x_4}(u_{41} \otimes u_{421})$ ). Finally,  $a_2$  gets a utility table set containing  $u_{321}$  and  $u'_{321}$ . After that,  $a_2$  updates  $u_{321}$  by a joint operation of  $u'_{321}$  (i.e.,  $u_{321} = u_{321} \otimes u'_{321}$ ) and

sends the new set including the updated utility table  $u_{321}$  to its parent  $a_1$ .

When  $a_1$  receives the UTIL message from  $a_2$ , it saves the utility table set, and then joins the relative private functions  $f_{12}$  and  $f_{13}$  into  $u_{321}$  (i.e.,  $u_{321} = u_{321} \otimes f_{12} \otimes f_{13}$ ). Since  $EV(a_1, a_2) = \{a_2, a_3\}$  and the variables  $x_2$  and  $x_3$  are both relative to the utility table  $u_{321}$ , they are grouped into one set. But since  $k_e = 1$ ,  $a_1$  still eliminates the variables  $x_2$  and  $x_3$  from the utility table in the set  $\{u_{321}\}$ , respectively, and gets a new set as its own utility table set. Since it is the root agent and has received all the UTIL messages from its children,  $a_1$  joins all the utility tables in the set derived from eliminating  $x_2$  and  $x_3$ , and chooses the optimal value  $v_1^*$  for itself. Thus, the value propagation phase starts.  $a_1$  selects the optimal values  $v_2^*$  and  $v_3^*$  for  $a_2$  and  $a_3$ , and propagates the assignment  $\{(x_1 = v_1^*), (x_2 = v_2^*), (x_3 = v_3^*)\}$  to  $a_2$ . Then,  $a_2$  selects the optimal assignments for  $a_4$  and  $a_5$  after receiving the VALUE message from  $a_1$ , and sends the assignment  $\{(x_1 = v_1^*), (x_2 = v_2^*), (x_3 = v_3^*), (x_4 = v_4^*), (x_5 = v_5^*)\}$  to  $a_3$ . Upon receipt of the VALUE message,  $a_3$  assigns for itself and sends the assignments received from  $a_2$  to its child  $a_4$ . And  $a_4$  performs just like  $a_3$ . The algorithm terminates after the leaf agent  $a_5$  receives the VALUE message and assigns for itself.

## C Experiment Results

### C.1 The Experimental Configuration

#### The Experiment with Different Agent Numbers

- Problem type: Random ADCOPs
- Agent numbers: [8, 24]
- Density: 0.25
- Domain size: 3

#### The Experiment with Different Densities

- Problem type: Random ADCOPs
- Agent numbers: 8
- Density: [0.25, 1.0]
- Domain size: 8

#### The Experiment with Different Domain size

- Problem type: Random ADCOPs
- Agent numbers: 8
- Density: 0.4
- Domain size: [4, 14]

#### The Experiment with Different Tightness

- Problem type: ADCSPs
- Agent numbers: 10
- Density: 0.4
- Domain size: 10
- Tightness: [0.1, 0.8]

### C.2 The Experiment Results

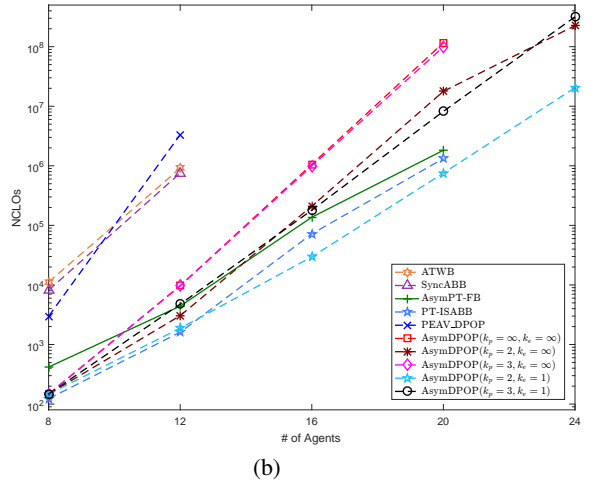
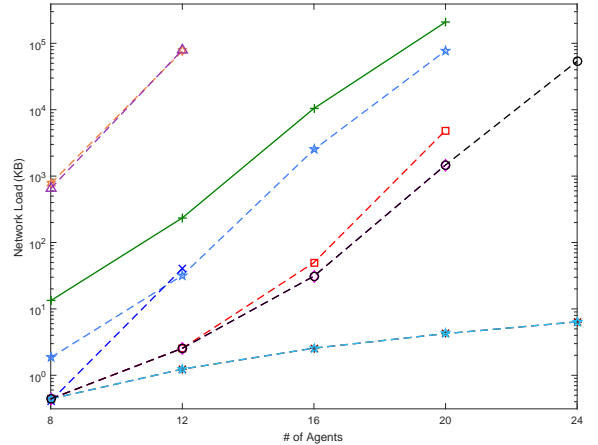
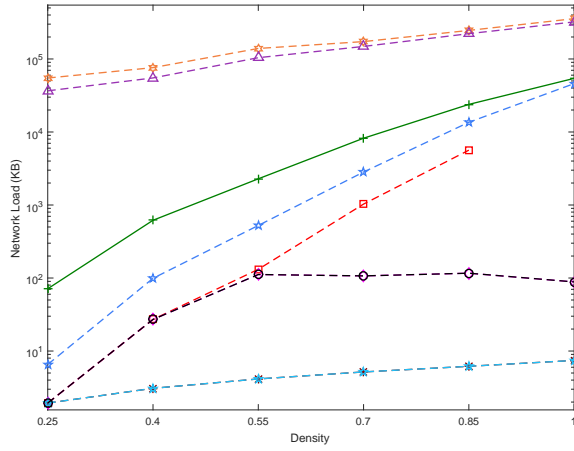
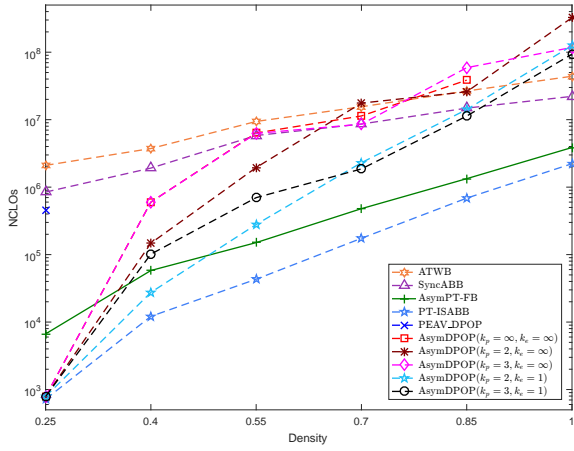


Figure 5: Performance comparison under different agent numbers

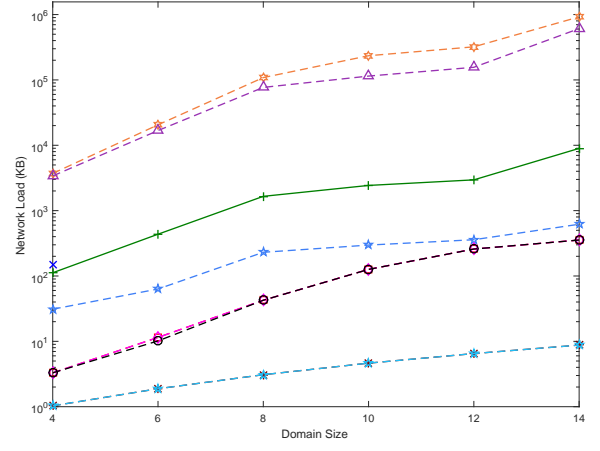


(a)

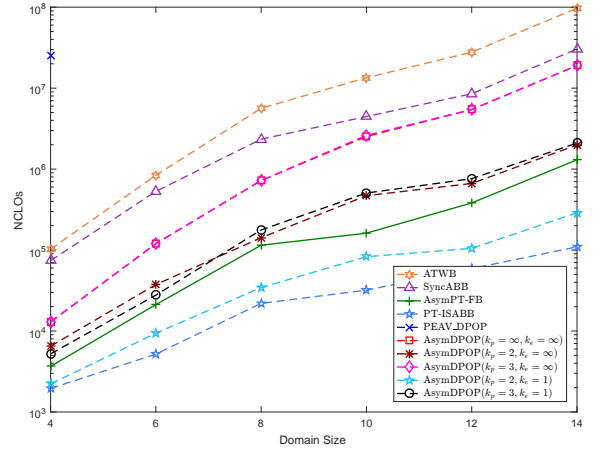


(b)

Figure 6: Performance comparison under different densities

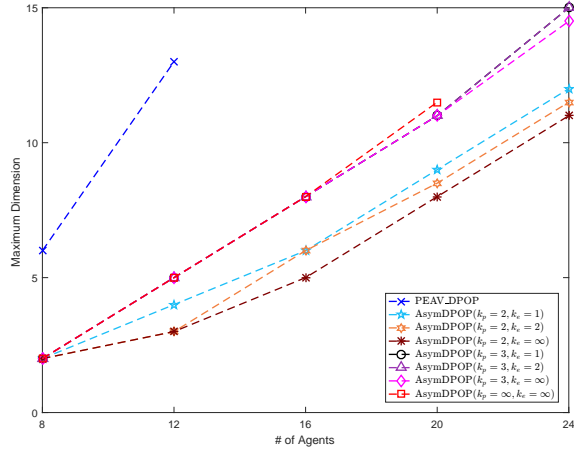


(a)

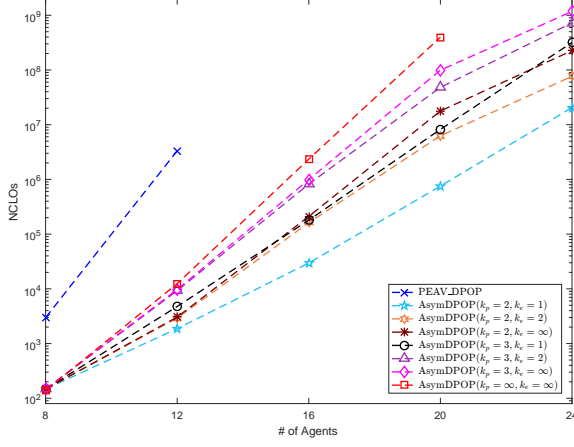


(b)

Figure 7: Performance comparison under different domain sizes

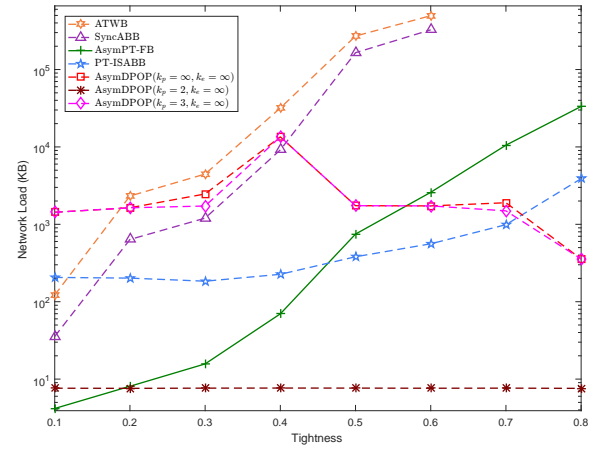


(a)

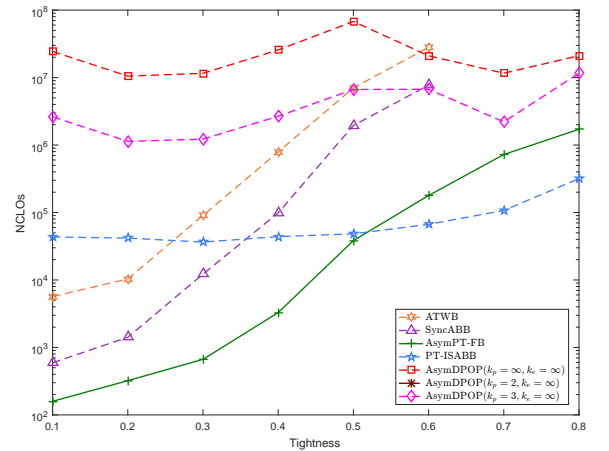


(b)

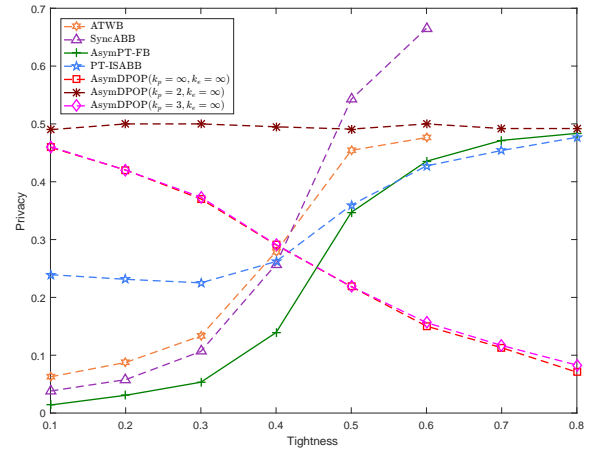
Figure 8: Performance comparison under different batch sizes



(a)



(b)



(c)

Figure 9: Performance comparison under different tightness