Lifting in Multi-agent Systems under Uncertainty (Supplementary Material)

Tanya Braun[1] **Marcel Gehrke**[2] **Florian Lau**[3] **Ralf Möller**[2]

[1]Computer Science Department, University of Münster, Münster, Germany
[2]Institute of Information Systems, University of Lübeck, Lübeck, Germany
[3]Institute of Telematics, University of Lübeck, Lübeck, Germany

1 ADDITIVE COUNT CONVERSION

With additive semantics, the combination function becomes summation, which applies to the join, which is usually done using multiplication, as well as the count-conversion where multiplication is replaced by summation as well. We need some further definitions for the formal operator that is based on the operator defined by Taghipour [2013]. We use the framework of general parfactors under which reward functions with PRVs and CRVs as inputs fall, which is also based on the work cited above.

Definition 1 (Logvar, PRV, parfactor, model)**.** *Let* $\mathbf{R}$ *be a set of random variable (randvar) names,* $\mathbf{L}$ *a set of logical variable (logvar) names,* $\mathbf{\Phi}$ *a set of factor names, and* $\mathbf{U}$ *a set of constants. All sets are finite. Each logvar* L *has a domain* $dom(L) \subseteq \mathbf{U}$*. A* constraint *is a tuple* $(\mathcal{X}, C_{\mathcal{X}})$ *of a sequence of logvars* $\mathcal{X} = (X_1, \ldots, X_l)$ *and a set* $C_{\mathcal{X}} \subseteq \times_{i=1}^{l} dom(X_i)$*. The symbol* $\top$ *for* C *marks that no restrictions apply, i.e.,* $C_{\mathcal{X}} = \times_{i=1}^{l} dom(X_i)$*. A* PRV $R(L_1, \ldots, L_l), l \geq 0$*, is a construct of a randvar* $R \in \mathbf{R}$ *possibly combined with logvars* $L_1, \ldots, L_l \in \mathbf{L}$*. If* $l = 0$*, the PRV is parameterless and forms a propositional randvar. The term* $ran(A)$ *denotes the possible values (range) of a PRV* A*. We use bold symbols for sets and calligraphic symbols for sequences of PRVs or logvars. For a set* $\boldsymbol{A}$ *or sequence* $\mathcal{A}$*, the range refers to all possible combinations of range values of the PRVs therein. An* event $A = a$ *denotes the occurrence of PRV* A *with range value* $a \in ran(A)$*. If* A *is clear from the context, we write* a*.*
We denote a parametric factor (parfactor) *by* $\phi(\mathcal{A})_{|C}$ *with* $\mathcal{A}$ *a sequence of PRVs,* $\phi : ran(\mathcal{A}) \mapsto \mathbb{R}^+$ *a function with name* $\phi \in \mathbf{\Phi}$*, and* C *a constraint on the logvars in* g*. A PRV* A *or logvar* L *under constraint* C *is given by* $A_{|C}$ *or* $L_{|C}$*, respectively. We may omit* $|\top$ *in* $A_{|\top}$*,* $L_{|\top}$*, or* $\phi(\mathcal{A})_{|\top}$*. A* model G *is a set of parfactors* $\{g_i\}_{i=1}^{n}$*.*
A utility parfactor g_U *maps to a utility PRV* $U(\boldsymbol{X})$ *with a signature of* $\phi_{U(\boldsymbol{X})}(\mathcal{A})_{|C}$*, i.e., stores the result of* ϕ *in* U*.*

The term $lv(\Gamma)$ refers to the logvars in Γ, which may be a PRV, a constraint, a parfactor, or a model. The term $rv(\Gamma)$ refers to the set of PRVs in a parfactor or model. The term $gr(\Gamma)$ denotes the set of all instances of Γ w.r.t. given constraints. An instance is an instantiation (grounding) of Γ, substituting the logvars in Γ with a set of constants from given constraints. If Γ is a constraint, $gr(\Gamma)$ refers to the second component $C_{\mathcal{X}}$. We use the expression $lv_U(.)$ to refer to the logvars of the utility PRV(s) in the input, which can be a PRV, a parfactor, or a model.

We assume familiarity with operators of relational algebra such as projection π and join $\bowtie$. An alignment replaces the occurrences of one object with another. The notion of count-normalisation says that each instance of one logvar sequence $\mathbf{Z}$ needs to refer to the same number of instances of another logvar sequence $\mathbf{Y}$. If this number exists, we refer to it as $ncount_{\mathbf{Y}|\mathbf{Z}}$.

Operator 1 Count-conversion for utility parfactors

Operator COUNT-CONVERT
Inputs:
(1) Utility parfactor $g_u = \phi_{U(\boldsymbol{X})}(\mathcal{A})_{|C}$
(2) Logvar $X \in lv(\mathcal{A})$ and $X \in \boldsymbol{X}$, to count in g_u
Preconditions:
(1) There is exactly one atom $A_i \in \mathcal{A}$ with $X \in lv(A_i)$.
(2) X is count-normalised w.r.t. $\boldsymbol{Z} = lv(\mathcal{A}) \setminus \{X\}$ in C.
(3) For all counted logvars $X^{\#}$ in g:
$\pi_{X,X^{\#}}(C) = \pi_X(\pi_{\boldsymbol{X}}(C)) \times \pi_{X^{\#}}(\pi_{\boldsymbol{X}}(C))$.
Output: utility parfactor $\phi'_{U'}(\mathcal{A}')_{|C}$ such that
(1) $U' = \#_X[U(\boldsymbol{X})]$,
(2) $\mathcal{A}' = (A_1, \ldots, A_{i-1}) \circ A'_i \circ (A_{i+1}, \ldots, A_n)$, $A'_i = \#_X[A_i]$, and
(3) for each valuation $\boldsymbol{a}'$ to $\mathcal{A}'$ with $a'_i = h$,

$$\phi'_{U(\boldsymbol{X})}(\ldots, a_{i-1}, h, a_{i+1}, \ldots) = \sum_{a \in ran(A_i)} h(a_i)\phi_{U(\boldsymbol{X})}(\ldots, a_{i-1}, a_i, a_{i+1}, \ldots)$$

where h is a histogram as defined in Eq. (9)
Postcondition:
$G_U \equiv G_U \setminus \{g_u\} \cup$ COUNT-CONVERT(g_u, X)

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

Operator 2 Additive join of utility parfactors

Operator JOIN

Inputs:

(1) Utility parfactor $g_{u'} = \phi_{u'}(\mathcal{A}_{u'})_{|C_{u'}}$

(2) Utility parfactor $g_{u''} = \phi_{u''}(\mathcal{A}_{u''})_{|C_{u''}}$

(3) Alignment $\theta = \{\boldsymbol{Z}_{u''} \to \boldsymbol{Z}_{u'}\}$, between the logvars of $g_{u'}$ and $g_{u''}$

Preconditions:

(1) for $v = u', u'' : \boldsymbol{Y}_v = lv(\mathcal{A}_v) \setminus \boldsymbol{Z}_v$ is count-normalised w.r.t. $\boldsymbol{Z}_v$ in $\pi_{\boldsymbol{X}_v}(C_v)$

Output: utility parfactor $\phi_u(\mathcal{A}_u)_{|C}$ such that

(1) $u = U(\boldsymbol{X}), \boldsymbol{X} = lv(u') \cup lv(u'')\theta$,

(2) $\mathcal{A}_u = \mathcal{A}_{u'} \bowtie \mathcal{A}_{u''}\theta$,

(3) $C = C_{u'} \bowtie C_{u''}\theta$ (component-wise), and

(4) for each valuation $\boldsymbol{a} \in ran(\mathcal{A}_u)$ with $\boldsymbol{a}_{u'} = \pi_{\mathcal{A}_{u'}}(\boldsymbol{a})$ and $\boldsymbol{a}_{u''} = \pi_{\mathcal{A}_{u''}}(\boldsymbol{a})$

$$\phi_u(\boldsymbol{a}) = \frac{1}{r''_u}\phi_{u'}(\boldsymbol{a}_{u'}) + \frac{1}{r'_u}\phi_{u''}(\boldsymbol{a}_{u''})$$

where $r'_u = ncount_{\mathbf{Y}_{u'}|\mathbf{Z}_{u'}}(\pi_{\mathbf{X}_{u'}}(C_{u'}))$ and $r''_u = ncount_{\mathbf{Y}_{u''}|\mathbf{Z}_{u''}}(\pi_{\mathbf{X}_{u''}}(C_{u''}))$

Postcondition: $G_U \equiv G_U \setminus \{g_{u'}, g_{u''}\} \cup \text{JOIN}(g_{u'}, g_{u''}, \theta)$

Please refer to Taghipour [2013] for more details on count-normalisation and count-conversions.

Operator 1 shows the additive count-conversion. The operator transforms a parfactor with a logvar to be counted directly into a parfactor with the logvar counted. That is, it turns the PRV in which the logvar occurs into a CRV and transforms the mapped values accordingly.

The long version of this calculation is grounding the logvar, joining the (partially) grounded parfactors, determining counting symmetry, and transforming the grounded instances into CRVs. Joining parfactors with multiplicative semantics (mapping to probabilities or potentials) is done using multiplication, which is also the name of the operator that the following operator is based on [Taghipour, 2013]. Since rewards have additive semantics, combining two parfactors does not work with multiplication but with summation. Operator 2 shows a join for parfactors with additive semantics.

2 FULL DECTIGER EXAMPLE

We use the specification of the DecTiger benchmark from the MADP tool box.[1]. Listing 1 shows the original DecTiger version in the MADP input format (on the last page). The DecPOMDP model reads as follows:

- $\boldsymbol{I} = \{agent_1, agent_2\}$,
- S, $ran(S) = \{$*tiger-left*, *tiger-right*$\} = \{tl, tr\}$,
- $\boldsymbol{A} = \{A_i\}_{i \in \boldsymbol{I}}$, $\forall i : ran(A_i) = \{$*listen*, *open-left*, *open-right*$\} = \{li, ol, or\}$, and
- $\boldsymbol{O} = \{O_i\}_{i \in \boldsymbol{I}}$, $\forall i : ran(O_i) = \{$*hear-left*, *hear-right*$\} = \{hl, hr\}$.

with T, R, and Ω (= O in the listing) in tabular notation in full below (lines are reordered compared to the listing to match the order of the inputs in the definitions).

S	A_1	A_2	T_{tl}	T_{tr}
tl	li	li	1	0
tl	li	ol	0.5	0.5
tl	li	or	0.5	0.5
tl	ol	li	0.5	0.5
tl	ol	ol	0.5	0.5
tl	ol	or	0.5	0.5
tl	or	li	0.5	0.5
tl	or	ol	0.5	0.5
tl	or	or	0.5	0.5
tr	li	li	0	1
tr	li	ol	0.5	0.5
tr	li	or	0.5	0.5
tr	ol	li	0.5	0.5
tr	ol	ol	0.5	0.5
tr	ol	or	0.5	0.5
tr	or	li	0.5	0.5
tr	or	ol	0.5	0.5
tr	or	or	0.5	0.5

S	A_1	A_2	R
tl	li	li	-2
tl	li	ol	-101
tl	li	or	9
tl	ol	li	-101
tl	ol	ol	-50
tl	ol	or	-100
tl	or	li	9
tl	or	ol	-100
tl	or	or	20
tr	li	li	-2
tr	li	ol	9
tr	li	or	-101
tr	ol	li	9
tr	ol	ol	20
tr	ol	or	-100
tr	or	li	-101
tr	or	ol	-100
tr	or	or	-50

S	A_1	A_2	$\Omega_{hl,hl}$	$\Omega_{hr,hl}$	$\Omega_{hl,hr}$	$\Omega_{hr,hr}$
tl	li	li	0.7225	0.1275	0.1275	0.0225
tl	li	ol	0.25	0.25	0.25	0.25
tl	li	or	0.25	0.25	0.25	0.25
tl	ol	li	0.25	0.25	0.25	0.25
tl	ol	ol	0.25	0.25	0.25	0.25
tl	ol	or	0.25	0.25	0.25	0.25
tl	or	li	0.25	0.25	0.25	0.25
tl	or	ol	0.25	0.25	0.25	0.25
tl	or	or	0.25	0.25	0.25	0.25
tr	li	li	0.7225	0.1275	0.1275	0.0225
tr	li	ol	0.25	0.25	0.25	0.25
tr	li	or	0.25	0.25	0.25	0.25
tr	ol	li	0.25	0.25	0.25	0.25
tr	ol	ol	0.25	0.25	0.25	0.25
tr	ol	or	0.25	0.25	0.25	0.25
tr	or	li	0.25	0.25	0.25	0.25
tr	or	ol	0.25	0.25	0.25	0.25
tr	or	or	0.25	0.25	0.25	0.25

with $T_{s'} = T(s', s, a_1, a_2), s' \in \{tl, tr\}$ and $\Omega_{o_1,o_2} = \Omega((o_1, o_2), s, a_1, a_2)$, $o_1, o_2 \in \{hl, hr\}$. The transition function only states that as long as both agents only listen, the state does not change (identity). When at least one agent

[1] https://github.com/MADPToolbox/MADP/blob/master/problems/dectiger.dpomdp

opens a door, the game basically restarts with the new state being set according to a uniform distribution. It is basically a way of keeping the game infinite by resetting the state to an arbitrary one whenever the agents end the game by opening a door (to either lose—tiger, or win—gold). One could argue that opening a door only ends the game and might not necessarily imply a restart. In that case, one would keeping the state as is in all cases ($(1,0)$ distribution for all tl lines, $(0,1)$ distribution for all tr lines) and re-spawn the game with an arbitrary starting state, sampled from a $(0.5, 0.5)$ distribution, for do-overs.

The DecTiger model has the same action and observation sets for both agents and exhibits a counting symmetry. Thus, it can be rewritten into a counting model with $K = 1$. We index the one partition with c to distinguish it from the later case where we take an isomorphic viewpoint.

- $\mathfrak{I}_c = \{agent_1, agent_2\}$,
- $\bar{S} = S = \{tl, tr\}$,
- $\bar{A}_c = \{\#_X[A(X)]\}, ran(A(X)) = \{li, ol, or\}$,
- $T_c(S', S, \boldsymbol{A}) = P(S' \mid S, \boldsymbol{A})$,
- $R_c(S, \boldsymbol{A})$,
- $\bar{O}_c = \{\#_X[O(X)]\}, ran(O(X)) = \{hl, hr\}$.
- $\Omega_c(\boldsymbol{O}, S) = P(\boldsymbol{O} \mid S, \boldsymbol{A})$

with $\bar{T}_c$, $\bar{R}_c$, and $\bar{\Omega}_c$ as follows ($\#A$ short for $\#_X[A(X)]$; histogram positions: $[li, ol, or]$):

S	$\#A$	$T_{c,tl}$	$T_{c,tr}$
tl	$[2,0,0]$	1.0	0.0
tl	$[1,1,0]$	0.5	0.5
tl	$[1,0,1]$	0.5	0.5
tl	$[0,1,1]$	0.5	0.5
tl	$[0,2,0]$	0.5	0.5
tl	$[0,0,2]$	0.5	0.5
tr	$[2,0,0]$	0.0	1.0
tr	$[1,1,0]$	0.5	0.5
tr	$[1,0,1]$	0.5	0.5
tr	$[0,1,1]$	0.5	0.5
tr	$[0,2,0]$	0.5	0.5
tr	$[0,0,2]$	0.5	0.5

S	$\#A$	R_c
tl	$[2,0,0]$	-2
tl	$[1,1,0]$	-101
tl	$[0,2,0]$	-50
tl	$[0,1,1]$	-100
tl	$[0,0,2]$	20
tl	$[1,0,1]$	9
tr	$[2,0,0]$	-2
tr	$[1,1,0]$	9
tr	$[0,2,0]$	20
tr	$[0,1,1]$	-100
tr	$[0,0,2]$	-50
tr	$[1,0,1]$	-101

S	$\#A$	$\Omega_{c,[2,0]}$	$\Omega_{c,[1,1]}$	$\Omega_{c,[0,2]}$
tl	$[2,0,0]$	0.7225	0.1275	0.0225
tl	$[1,1,0]$	0.25	0.25	0.25
tl	$[0,2,0]$	0.25	0.25	0.25
tl	$[0,1,1]$	0.25	0.25	0.25
tl	$[0,0,2]$	0.25	0.25	0.25
tl	$[1,0,1]$	0.25	0.25	0.25
tr	$[2,0,0]$	0.7225	0.1275	0.0225
tr	$[1,1,0]$	0.25	0.25	0.25
tr	$[1,0,1]$	0.25	0.25	0.25
tr	$[0,2,0]$	0.25	0.25	0.25
tr	$[0,1,1]$	0.25	0.25	0.25
tr	$[0,0,2]$	0.25	0.25	0.25

As the histogram $[1,1]$ in Ω_c stands for two joint observations, (hl, hr) and (hr, hl), the probability for $\Omega_{c,[1,1]}$ counts twice. In general, a multinomial coefficient provides the number of inputs represented by a histogram, i.e., $n_k!/\prod_l n_l!$ in reference to Eq. (9).

For an isomorphic model, one would need to be able to factorise $\bar{T}_c$, $\bar{R}_c$, and $\bar{\Omega}_c$ into identical functions per agent, which is immediately possible for $\bar{\Omega}_c$ (and also provided like this in the book by Oliehoek and Amato [2016]):

S	$A(X)$	$\Omega_{i,hl}$	$\Omega_{i,hr}$
tl	li	0.85	0.15
tl	ol	0.5	0.5
tl	or	0.5	0.5
tr	li	0.85	0.15
tr	ol	0.5	0.5
tr	or	0.5	0.5

Functions $\bar{T}_c$ and $\bar{R}_c$ do not factorise accordingly. $\bar{T}_c$ does not factorise as we have this mix of uniform and identity distributions, which would lead to identity distributions whenever one *listen* operation is involved. However, the ground case only has identity distributions when both agents listen. If not encoding the reset into T, i.e., using identity distributions in all lines, then $\bar{T}_c$ would factorise trivially:

S	$A(X)$	$T_{i,tl}$	$T_{i,tr}$
tl	li	1	0
tl	ol	1	0
tl	or	1	0
tr	li	0	1
tr	ol	0	1
tr	or	0	1

A factorisation of $\bar{R}_c$ does not work out for all inputs. Matching the peak-shaped histograms in their reward, we could factorise parts of it as follows, leading to different values in the lines marked with $\lightning$.

S	$A(X)$	R_i
tl	li	-1
tl	ol	-25
tl	or	10
tr	li	-1
tr	ol	10
tr	or	-25

S	$\#_X[A(X)]$	R_c	
tl	$[2,0,0]$	-2	
tl	$[1,1,0]$	-26	$\lightning$
tl	$[0,2,0]$	-50	
tl	$[0,1,1]$	-15	$\lightning$
tl	$[0,0,2]$	20	
tl	$[1,0,1]$	9	
tr	$[2,0,0]$	-2	
tr	$[1,1,0]$	9	
tr	$[0,2,0]$	20	
tr	$[0,1,1]$	-15	$\lightning$
tr	$[0,0,2]$	-50	
tr	$[1,0,1]$	-26	$\lightning$

There are two other options. One option would be to match the rewards for the histograms that represent combinations of listening and opening a door:

S	$A(X)$	R_i
tl	li	-1
tl	ol	-100
tl	or	10
tr	li	-1
tr	ol	10
tr	or	-100

S	$\#_X[A(X)]$	R_c	
tl	$[2,0,0]$	-2	
tl	$[1,1,0]$	-101	
tl	$[0,2,0]$	-200	↯
tl	$[0,1,1]$	-90	↯
tl	$[0,0,2]$	20	
tl	$[1,0,1]$	9	
tr	$[2,0,0]$	-2	
tr	$[1,1,0]$	9	
tr	$[0,2,0]$	20	
tr	$[0,1,1]$	-90	↯
tr	$[0,0,2]$	-200	↯
tr	$[1,0,1]$	-101	

The other option would be to match the rewards for those histograms with combinations of opening different doors:

S	$A(X)$	R_i
tl	li	-1
tl	ol	-110
tl	or	10
tr	li	-1
tr	ol	10
tr	or	-110

S	$\#_X[A(X)]$	R_c	
tl	$[2,0,0]$	-2	
tl	$[1,1,0]$	-111	↯
tl	$[0,2,0]$	-220	↯
tl	$[0,1,1]$	-100	
tl	$[0,0,2]$	20	
tl	$[1,0,1]$	9	
tr	$[2,0,0]$	-2	
tr	$[1,1,0]$	9	
tr	$[0,2,0]$	20	
tr	$[0,1,1]$	-100	
tr	$[0,0,2]$	-220	↯
tr	$[1,0,1]$	-111	↯

So, an isomorphic model using one of the three options would not be able to capture that both agents agreeing on an action, even though it opens the door to the tiger, costs them less than opening different doors. Here, one would need the expressiveness of a counting model. This limitation is a direct consequence of what we observed in the proof on Lemma 3: A non-peak-shaped histogram cannot map to a larger value than a peak-shaped histogram when the CRV came from a PRV as given here.

However, an isomorphic model would automatically exclude any policy where one agent opens the door while the other agent either listens or opens the other door. The first one is sort of a waste of the listening operation as the game ends nonetheless and the observation result cannot be used. The second one means that the door to the tiger is definitely opened, therefore, it should be avoided, highlighting how isomorphic models can help to determine sensible policies.

Using the ground version, the counting version, and one of the isomorphic versions, let us have a brief look at the behaviour of the example under the complexities given under a rising N with one exception: We use the binomial coefficient to actually calculate the number of possible histograms and do not use the upper bound of n^a and n^o. In addition, we include artificial partitions of the same dimensions to see the effect of rising K. So, $a = 3$, $o = 2$, $s = 2$, $n = N$ if $K = 1$. Otherwise, $n = 100$ and $N = K \cdot n$. We set $\tau = 2$ arbitrarily. Figures 1 to 4 show the behaviour according to the complexities derived in the paper for T (as the worst case of the three functions in this setup), the evaluation cost, and the policy space. Please note the log-scale on the y-axis. The figures highlight impressively the differences between the different models in terms of complexity.

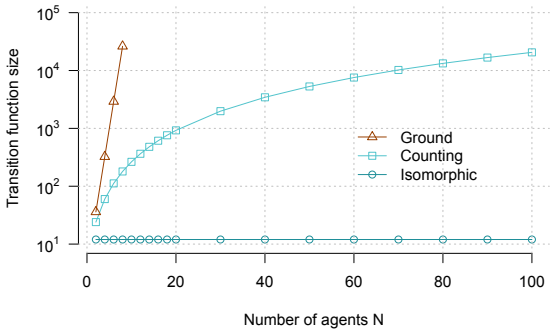

Figure 1: The transition function size $\mathbb{T}$ under rising N

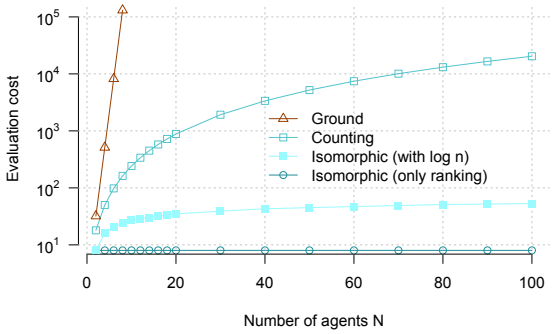

Figure 2: The policy evaluation cost $\mathbb{C}$ under rising N

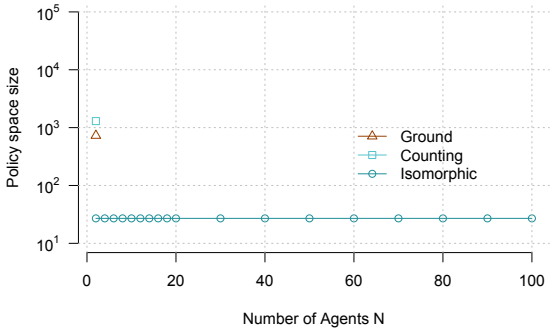

Figure 3: The policy space size $\mathbb{P}$ under rising N

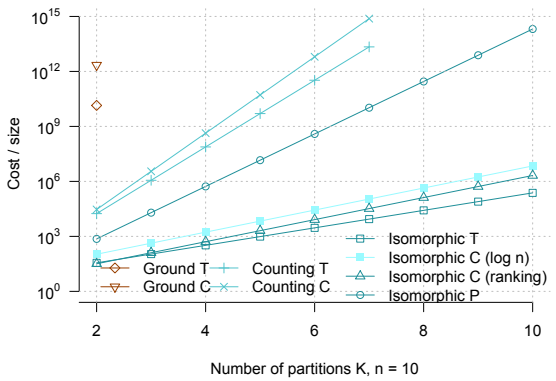

Figure 4: Transition function sizes and evaluation cost for all settings and policy space sizes for the isomorphic setting under rising K with $n = 10$

References

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralised POMDPs*. Springer, 2016.

Nima Taghipour. *Lifted Probabilistic Inference by Variable Elimination*. PhD thesis, KU Leuven, 2013.

Listing 1 DecTiger specification in the MADP toolbox (without the comments from the source)

```
agents: 2
discount: 1
values: reward
states: tiger-left tiger-right
start: uniform
actions:
listen open-left open-right
listen open-left open-right
observations:
hear-left hear-right
hear-left hear-right
# Transition probabilities
T: * : uniform
T: listen listen : identity
# Observation probabilities: <2actions> : <state> : <2observations> : probability
O: * : uniform
O: listen listen : tiger-left : hear-left hear-left : 0.7225
O: listen listen : tiger-left : hear-left hear-right : 0.1275
O: listen listen : tiger-left : hear-right hear-left : 0.1275
O: listen listen : tiger-left : hear-right hear-right : 0.0225
O: listen listen : tiger-right : hear-left hear-left : 0.7225
O: listen listen : tiger-right : hear-left hear-right : 0.1275
O: listen listen : tiger-right : hear-right hear-left : 0.1275
O: listen listen : tiger-right : hear-right hear-right : 0.0225
# Rewards: <2 actions> : <state> : * : * : reward
R: listen listen: * : * : * : -2
R: open-left open-left : tiger-left : * : * : -50
R: open-right open-right : tiger-right : * : * : -50
R: open-left open-left : tiger-right : * : * : 20
R: open-right open-right : tiger-left : * : * : 20
R: open-left open-right : tiger-left : * : * : -100
R: open-left open-right : tiger-right : * : * : -100
R: open-right open-left : tiger-left : * : * : -100
R: open-right open-left : tiger-right : * : * : -100
R: open-left listen : tiger-left : * : * : -101
R: listen open-right : tiger-right : * : * : -101
R: listen open-left : tiger-left : * : * : -101
R: open-right listen : tiger-right : * : * : -101
R: listen open-right : tiger-left : * : * : 9
R: listen open-left : tiger-right : * : * : 9
R: open-right listen : tiger-left : * : * : 9
R: open-left listen : tiger-right : * : * : 9
```