On the Necessity and Design of Coordination Mechanism for Cognitive Autonomous Networks

Anubhab Banerjee1,2, Stephen S. Mwanje1, and Georg Carle 2
1Nokia Bell Labs, Munich, Germany
2Dept. of Informatics, Technical University of Munich, Germany
Email:anubhab.banerjee@tum.de

Abstract—Cognitive Autonomous Networks (CAN) [1] are promoted to advance Self Organizing Network (SON), replacing rule-based SON Functions (SFs) with Cognitive Functions (CFs), which learn optimal behavior by interacting with the network. As in SON, CFs do encounter conflicts due to overlap in parameters or objectives. However, owing to the non-deterministic behavior of CFs, these conflicts cannot be resolved using rule-based methods and new solutions are required. This paper investigates the CF deployments with and without a coordination mechanism, and proves both heuristically and mathematically that a coordination mechanism is required. Using a two-CF Multi-Agent-System model with the possible types of conflicts, we show that the challenge is a typical bargaining problem, for which the optimal response is the Nash bargaining Solution (NBS). We use NBS to propose a coordination mechanism design that is capable of resolving the conflicts and show via simulations how implementation of the proposed solution is feasible in real life scenario.

Index Terms—Cognitive Autonomous Networks, Conflict Resolution, Game Theory, NBS, Prisoner’s Dilemma

I. INTRODUCTION

Continuous increase of mobile network users and their online activities have motivated the deployment of several Radio Access Technologies (RATs) to improve spectral efficiency and Quality of Service (QoS). To address the increasing operational complexity coming from introduction of new RATs, network automation exemplified by Self-Organizing Networks (SON) [2] is applied. SON proposed to deploy several closed-loop control-based functions, called SON Functions (SFs), each of which deals with a specific problem like Mobility Robustness Optimization (MRO), Mobility Load Balancing (MLB) etc. Based on the network states (e.g., changes in Key Performance Indicators (KPIs)), an SF determines individual network configuration parameters following some predefined rules. Since there are multiple SFs present in a SON, conflicts of interest may arise among them due to overlap of parameters or objectives. At the top of these SFs, rule based SON coordination ensures that the SFs do not conflict with one another during operations.

SON has two primary disadvantages - i) rule based SFs have limited capability in adapting themselves in a changing environment, ii) large number of rules makes maintenance and upgradation of the system difficult. Cognitive Autonomous Networks (CAN) [1] overcome the problems of SON and provides a more flexible system by replacing SFs with CFs. Unlike SF, CF does not exhibit rule-based behavior, rather it learns from the environment and acts based on its learning. As CFs show a non-deterministic behavior, conflicts among CFs cannot be resolved using existing rule-based methods and some new approach is necessary. There already exist prior research works which propose different ways on how the CFs can coordinate and work in a decentralized manner [3], [4], but, these works show neither the necessity of collaboration among the CFs nor the optimality of the solution provided.

In this paper our contributions are three-fold - 1. we mathematically prove, using Prisoner’s Dilemma from Game Theory [5], that coordination mechanism among the CFs is needed for better performance of the system, 2. we design an easily implementable but effective coordination mechanism capable of determining the optimal configuration for a certain state of the network, and, 3. we also model a heterogeneous Multi Agent System (MAS) in Python that exhibits all types of conflicts [2] among the CFs and implement our proposed solution in that model to show the its feasibility in real life.

II. CONFLICT MODEL IN CAN

Network automation functions typically exhibit three types of conflicts -
• Category A. Configuration conflict which occurs on either, (A1) Input, or, (A2) Output parameter(s).
• Category B. Measurement conflict where action of one function influences measurement of output of another.
• Category C. Characteristic conflict which are of two types - (C1)Direct characteristic conflict and (C2) Logical dependency conflict.

Let us assume that there are two CFs, $F_1$ and $F_2$ working at a Base Station (BS) with objectives $o_1$ and $o_2$ respectively and without communicating with each other. These are the two agents of the MAS. There are two inputs to $F_1$ - $p_1$ and $p_2$, and $F_1$ tries to optimize its output $o_1$. There are two inputs to $F_2$ - $p_1$ and $o_1$, and $F_2$ tries to optimize its output $o_2$. The system model is depicted in Fig. 1 and the functions with their related information are listed in Table I.

Now, we can see that all three types of conflicts, are present in this model -
• As both $F_1$ and $F_2$ share the same input parameter ($p_1$), so if they have different levels of interest, it is an input parameter conflict (A1).
• As actions of $F_1$ affects the measurement of $o_2$, output of $F_2$, it is a measurement conflict (B1).
Fig. 1: System model

| Function | Inputs | Outputs | Objective |
|----------|--------|---------|-----------|
| \( F_1 \) | \( p_1, p_2 \) | \( o_1 \) | Optimize \( o_1 \) |
| \( F_2 \) | \( p_1, p_2 \) | \( o_2 \) | Optimize \( o_2 \) |

**TABLE I: CF Descriptions**

- Changing \( p_2 \) affects \( o_1 \) which in turn changes \( o_2 \) hence it is a logical dependency conflict (C2).

In the next sections we use this model to prove necessity of coordination mechanism among the CFs and to implement our proposed optimal solution.

### III. NECESSITY OF COORDINATION MECHANISM IN CAN

Let us consider the model described in section II and formulate it as a normal-form game [6] where \( F_1 \) and \( F_2 \) are the players of the game. When there is a conflict of interest (A1, B1 or C2) between the players, each player can take one of these following strategies - either, the player continues to work on the interest (\( T \)), or, it gives up the interest (\( G \)). The payoffs are defined as follows:

- Both of them choose \( G \): no one changes the interest and it remains constant. Both the CFs get equal payoff out of it denoted by \( r_1 \).
- Both of them choose \( T \): the payoff both of them get is \( r_2 \). The benefit of fighting for the interest is worse than keeping it constant because it may change to worse outcomes for any of the parties and so, \( r_2 < r_1 \).
- One of them selects \( T \) and the other selects \( G \): one who selects \( T \) gets a payoff \( r_3 \) and the other one gets a payoff \( r_4 \) with \( r_3 > r_4 \).

**Relation among\( r_1, r_2, r_3 \) and \( r_4 \):** It is obvious that payoff is higher when the CF can adjust the interest than when it remains constant, i.e., \( r_3 > r_1 \). The payoff for one is also higher when either the interest is controlled by itself or the interest remains constant than when it is changed according to the other’s will, i.e., \( r_2 > r_4 \) and \( r_1 > r_4 \). The payoff for a CF is higher again when only the CF changes it than when both of them change it, i.e., \( r_3 > r_1 \). The above observations can be summarized as:

\[
 r_3 > r_1 > r_2 > r_4
\]

As mentioned in [5], two criteria for a problem to qualify as Prisoner’s Dilemma are:

- regardless of what the other players do, each player receives a higher payoff for defecting behavior than for cooperating behavior.
- all agents get lower payoff if all defect than cooperate.

Now, a conflict between CFs can be formulated exactly as a Prisoner’s Dilemma where the defecting behaviors is choosing \( T \) and cooperative behavior is choosing \( G \), because, from Eq. [1] we observe that:

- Regardless of what the other CF does, each CF receives a higher payoff for selecting \( T \) than selecting \( G \).
- All CFs receive a lower payoff if all choose \( T \) than if all choose \( G \).

Following the solution of Prisoner’s Dilemma, where best action for each player is to choose defecting behavior, the best action for each CF is: selecting \( T \).

On the contrary, if the CFs work with the existence of a coordinating mechanism, they find that the best possible action for each CF is - choosing \( G \), because, when both of them select \( G \), both of them get higher payoff (\( r_1 \)) than the payoff they get (\( r_2 \)) when they select \( T \). This proves the necessity of a coordination mechanism in the network.

### IV. PROPOSED SOLUTION AND IMPLEMENTATION

Let us again consider the model described in section I and assume that both \( F_1 \) and \( F_2 \) try to maximize their objectives and both \( o_1 \) and \( o_2 \) are Gaussian distribution functions given by the following equations:

\[
o_1 = e^{-\frac{r_1^2}{2\sigma_1^2}}
\]

\[
o_2 = e^{-\frac{(o_1-o_2)^2}{\sigma_1^2}}
\]

We also assume that default value (value at the initial stage when BS first becomes operational) of \( p_1 \) is 4 and \( p_2 \) is 100. These functions \( o_1, o_2 \) and values \( p_1, p_2 \) are assumed in such a way so that all types of conflicts are present between \( F_1 \) and \( F_2 \).

#### A. Proposed solution

To resolve the conflicts between \( F_1 \) and \( F_2 \) over the values of \( p_1 \) and \( p_2 \), we propose a bargaining between them and obtain the solution using Nash Bargaining Solution (NBS) [7]. NBS can be applied when - i) number of players is two or more, ii) there is a conflict of interest, and iii) there exists a solution if if negotiation breaks down, and the solution is obtained by finding the value for which product of outcome of the players is maximum. As the number of persons is two and there is a conflict of interest on agreement, Nash Bargaining Solution (NBS) can be applied to obtain the solution in this scenario [6] and this solution is optimal [7]. The optimal value of \( p_1 \) is determined following these steps sequentially:

- Both \( F_1 \) and \( F_2 \) generate a set of possible values for \( p_1 \) based on their learning history.
- For each value of \( p_1 \) in this set, the product of \( o_1 \) and \( o_2 \) is calculated while \( p_2 \) is kept constant.
- When the product is maximum, the corresponding value of \( p_1 \) is the optimal value.
After the optimal value of $p_1$ is determined, a set of possible values for $p_2$ is also generated by $F_1$ based on its learning history. For each value of $p_2$ in this set, the product of $o_1$ and $o_2$ is calculated while $p_1$ is kept constant at its optimal value. When the product is maximum, the corresponding value of $p_2$ is the optimal value. In this way the optimal configuration for the system can be determined.

In this solution we assume that $F_1$ and $F_2$ can generate sets of values for $p_1$ and $p_2$ based on their previous learning history. As an alternate, starting from the default, values of these parameters can be changed slowly and gradient descent approach can be used to reach the optimal.

B. Implementation and Observation

We build a framework in Python to implement the proposed solution in the MAS discussed in section III.

In Fig. 2 variations of $o_1$ and $o_2$ against $p_1$ and $p_2$ have been plotted. From Fig. 2a we see that for a fixed $p_2$, maximum value of $o_1$ is obtained when $p_1$ lies in between -10 and 10 and maximum value of $o_2$ is obtained when $p_1$ lies in between 0 and 10. Thus, in our simulation, we vary $p_1$ in between 0 and 10 in steps of 1 (step size can be made smaller for better accuracy) and plot the variation of $o_1 \cdot o_2$ against $p_1$ in Fig. 3a.

From Fig. 3a we see that $o_1 \cdot o_2$ is maximum when $p_1$ is 6, and so, according to NBS, this is the optimal value of $p_1$ for both $F_1$ and $F_2$.

Once the optimal value of $p_1$ is determined, it is kept constant at 6 and $p_2$ is varied to determine its optimal value. For a fixed $p_1$, when the value of $p_2$ increases, we see from Fig. 2a that the value of $o_1$ increases whereas from Fig. 2b we see that the value of $o_2$ remains almost constant. For this reason, when we plot $o_1 \cdot o_2$ against $p_2$ in Fig. 3b we see that $o_1 \cdot o_2$ increases with increase in value of $p_2$. As a specific range for $p_2$ cannot be determined, we vary $p_2$ between 50 and 300 in our simulation and according to NBS, optimal value of $p_2$ in this scenario is 300. However, in real life there is always a maximum and minimum value for a parameter between which it can be varied.

V. RELATED WORKS AND MOTIVATION

As the CAN with CFs has been formulated as a MAS in this paper, in this section we discuss relevant existing research works on MAS ([13], [21]) and removal of conflicts (reaching a consensus) in a MAS ([17], [22]).
them in a simulation environment to do a comparative study these cognitive functions and as a next step we want to test operational purposes. Currently we are building prototypes of resolve the conflicts and obtain the optimal settings for the show that our proposed solution can be implemented there to of conflicts among the CFs. We build a MAS in Python and system. We provide a solution to resolve the possible types coordinating mechanism is needed for better performance of the services (as shown in Table II). Ours is the first one which
functions (as shown in Table II). Ours is the first one which considers a MAS with all of these four properties.

Based on agent characteristics, we divide existing research works on MAS into several categories so that a combination of these features are covered in each category. These categories are listed in Table II. From Table II we see that there are a number of prior research articles which encompass one or some combinations of those four features described above, but there does not exist any paper which covers all the four features (as shown in Table II). Ours is the first one which considers a MAS with all of these four properties.

VI. CONCLUSION AND FUTURE DIRECTION

In this paper we discuss conflict resolution among cognitive functions and prove, using Prisoner’s Dilemma, that a coordinating mechanism is needed for better performance of the system. We provide a solution to resolve the possible types of conflicts among the CFs. We build a MAS in Python and show that our proposed solution can be implemented there to resolve the conflicts and obtain the optimal settings for the operational purposes. Currently we are building prototypes of these cognitive functions and as a next step we want to test them in a simulation environment to do a comparative study among possible coordination mechanisms.

REFERENCES

[1] S.S. Mwanje and C. Mannweiler. Towards cognitive autonomous networks in 5G. In 2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K), pages 1–8. IEEE, 2018.
[2] S. Hamalainen, H. Sanneck, and C. Sartori. LTE self-organising networks (SON): network management automation for operational efficiency. John Wiley & Sons, 2012.
[3] S.S. Mwanje and A. Mitschele-Thiel. Concurrent cooperative games for coordinating son functions in cognitive cellular networks. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 1298–1303. IEEE, 2015.
[4] S.S. Mwanje, H. Sanneck, and A. Mitschele-Thiel. Synchronized cooperative learning for coordinating cognitive network management functions. IEEE Transactions on Cognitive Communications and Networking, 4(2):244–256, 2018.
[5] R.M. Dawes. Social dilemmas. Annual review of psychology, 31(1):169–193, 1980.
[6] A. Banerjee and C. Mas Machuca. Game theory based content placement and pricing in connected edge networks. 2019.
[7] Eric Van Damme. The nash bargaining solution is optimal. Journal of Economic Theory, 38(1):78–100, 1986.
[8] S Sarika and V. Paul. Agenttab: An agent based approach to detect tabnabbing attack. Procedia Computer Science, 46:574–581, 2015.
[9] M. Khayyat and A. Awasthi. An intelligent multi-agent based model for collaborative logistics systems. Transp. Res. Procedia, 12:325–338, 2016.
[10] F. Rahimzadeh, L. Khani, and F. Mahan. High reliable and efficient task allocation in networked multi-agent systems. Autonomous Agents and Multi-Agent Systems, 29(6):1023–1040, 2015.
[11] A. Banerjee, N. Sastry, and C. Mas Machuca. Sharing content at the edge of the network using game theoretic centrality. In International Conference on Transparent Optical Networks ICTON, 2019.
[12] JP Mano and P Glize. Self-adaptive network of cooperative neuro-agents. In AISB04 Symposium on Adaptive Multi-Agent Systems, 2004.
[13] M. Gatti, P. Cavalin, S. Neto, C. Pinhanzez, C. dos Santos, D. Gribel, and A. Appel. Large-scale multi-agent-based modeling and simulation of microblogging-based online social networks. In International Workshop on Multi-Agent Systems and Agent-Based Simulation, pages 17–33. Springer, 2013.
[14] SS Manvi and MS Kakasageri. Multicast routing in mobile ad hoc networks by using a multiagent system. Information Sciences, 178(6):1611–1628, 2008.
[15] T. Morstyn, B. Hredzak, and V. Ageelidis. Cooperative multi-agent control of heterogeneous storage devices distributed in a dc microgrid. IEEE Transactions on Power Systems, 31(4):2974–2986, 2015.
[16] R. Bianchi, M. Martins, C. Ribeiro, and A. Costa. Heuristically-accelerated multiagent reinforcement learning. IEEE transactions on cybernetics, 44(2):252–265, 2013.
[17] S. Resmerita and M. Heymann. Conflict resolution in multi-agent systems. In 42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475), volume 3, pages 2537–2542. IEEE, 2003.
[18] E. Semsar-Kazerooni and K. Khorasani. A game theory approach to multi-agent team cooperation. In 2009 American Control Conference, pages 4512–4518. IEEE, 2009.
[19] J.K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In International Conference on Autonomous Agents and Multiagent Systems, pages 66–83. Springer, 2017.
[20] H. Liu, P. Zhang, B. Hu, and P. Moore. A novel approach to task assignment in a cooperative multi-agent design system. Applied Intelligence, 43(1):162–175, 2015.
[21] Q. Liu, X. Cui, and X. Hu. Conflict resolution within multi-agent system in collaborative design. In 2008 International Conference on Computer Science and Software Engineering. IEEE, 2008.
[22] M.R. Genesereth, M.L. Ginsberg, and J.S. Rosenschein. Cooperation without communication. In Readings in distributed artificial Intelligence, pages 220–226. Elsevier, 1988.

| F1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| F2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| F3 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| F4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

TABLE II: Existing works on MAS features