Minimizing Vertex Complexity of Tensor Product Digraphs via Quotient Graph for Multi-Agent Systems

Muhammad Ridwan Nasir Firdaus - 13525096
Informatics Engineering Study Program
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung 40132, Indonesia
13525096@std.stei.itb.ac.id

***Abstract*—Networked multi-agent frameworks frequently utilize graph-based communication structures to represent agent dynamics. When integrating these network configurations via a tensor product mechanism, the overall system suffers from a multiplicative escalation in processing complexity, which significantly hinders scalability. To alleviate this challenge, this study investigates the minimization of vertex complexity in tensor product directed graphs by exploiting the algebraic properties of quotient graphs. The introduced methodology identifies behaviorally equivalent states, specifically nodes that exhibit identical sets of incoming and outgoing connections, and consolidates them into unified macro-nodes via bisimulation partitioning. Analytical evaluation indicates that this quotient abstraction retains crucial structural features, such as strong connectivity and consensus performance, while considerably lowering overall computational overhead. Numerical experiments conducted on standard communication topologies confirm that the achieved complexity reduction correlates directly with the structural symmetry of the baseline digraphs. Highly symmetric configurations, including directed cycles and complete networks, demonstrate optimal compression, whereas irregular structures yield marginal improvements. Ultimately, these results indicate that the proposed framework delivers a mathematically rigorous and effective solution for expanding multi-agent architectures without compromising topological accuracy. *Keywords:* directed graph, tensor product, quotient graph, vertex complexity, multi-agent system, bisimulation, graph homomorphism**

I. Introduction

Recent advances in autonomous technologies have accelerated the adoption of Multi-Agent Systems (MAS) across a wide range of applications, including coordinated unmanned aerial vehicle formations, cooperative robotic swarms, and distributed sensing infrastructures. Capturing the interaction patterns among multiple agents with sufficient fidelity is essential for assessing coordination behavior, guaranteeing operational safety, and verifying the correctness of control strategies. Within the framework of networked control systems, the collective state space of a distributed MAS is commonly represented through the tensor product of individual directed graphs, a construction alternatively referred to as the categorical product or Kronecker product. Under this representation, each agent is associated with its own state-transition digraph, while vertices in the resulting product graph correspond to joint system configurations that describe the concurrent states of all participating agents. Despite its expressive power, this modeling approach encounters a fundamental scalability challenge. Consider a collection of k homogeneous agents, each characterized by a transition graph containing n states. The corresponding tensor product network $G^{\otimes k}$ contains n^k distinct configurations. Such exponential growth exemplifies the well-known phenomenon of combinatorial explosion. As the number of agents increases, the associated state space rapidly becomes prohibitively large, rendering procedures such as reachability verification, path analysis, and formal validation increasingly expensive and, in many cases, computationally infeasible under practical hardware constraints. Consequently, scalable verification frameworks require abstraction techniques capable of reducing state-space size while preserving essential structural and homomorphic characteristics. To address this issue, this study proposes a two-stage abstraction framework aimed at reducing vertex complexity in tensor product digraphs through quotient-based representations. The first stage performs state-space pruning using a reachability exploration process initiated from a designated initial state s_0. This procedure removes configurations that are theoretically valid yet unreachable during actual system operation. The second stage exploits the symmetry arising from agent homogeneity by introducing a permutation-based equivalence relation $\sim$ over the remaining states. Through this relation, states that differ only by agent permutations are identified as equivalent and subsequently merged into representative macro-vertices, producing a compact quotient graph. The primary contribution of this paper is the development of a scalable state-space reduction methodology grounded in principles from algebraic graph theory. It is shown that the resulting quotient representation preserves key structural properties of the original tensor product graph, particularly those related to reachability and coverage within the system. The effectiveness of the proposed approach is further examined through a multi-UAV case study in which each vehicle is modeled using four operational states. Experimental results demonstrate a substantial decrease in vertex-related complexity within the composite network. Beyond reducing memory requirements, the proposed abstraction

framework establishes a rigorous mathematical foundation for deriving reduction guarantees in larger and more sophisticated multi-agent systems.

II. Theoretical Background

A. Graphs

A graph is a mathematical structure used to model relationships between objects within a set. Formally, a graph is defined as follows.

Definition 1 (Graph)**.** A *graph* G is a pair $G = (V, E)$ where V is a finite non-empty set called the *vertex set*, and E is a set of unordered pairs of elements of V called the *edge set* [1], [2].

Definition 2 (Vertex Degree)**.** The *degree* of a vertex v in a graph G, denoted by $\deg(v)$, is the number of edges connecting vertex v with other vertices in G [1].

Graphs can be represented in various forms, one of which is the adjacency matrix $A(G)$ of size $n \times n$ where $n = |V|$, with entries $A_{ij} = 1$ if there is an edge between vertices v_i and v_j, and $A_{ij} = 0$ otherwise [1], [3].

B. Directed Graphs

A directed graph or *digraph* is an extension of an ordinary graph where each edge has a specific direction. This concept is highly relevant in modeling systems where the relationships between elements are asymmetric, such as communication networks in multi-agent systems.

Definition 3 (Directed Graph)**.** A *directed graph* or *digraph* G is an ordered pair $G = (V, E)$ where V is a finite non-empty set called the *vertex set*, and $E \subseteq V \times V$ is a set of ordered pairs called the *arc set*. Each arc $e = (u, v) \in E$ is directed from vertex u to vertex v [1], [4].

Definition 4 (In-Degree and Out-Degree)**.** In a digraph $G = (V, E)$, for each vertex $v \in V$, the following are defined:

- *In-degree*: $\deg^-(v) = |\{u \in V : (u, v) \in E\}|$
- *Out-degree*: $\deg^+(v) = |\{w \in V : (v, w) \in E\}|$

[1], [4]

Definition 5 (Strong Connectivity)**.** A digraph $G = (V, E)$ is said to be *strongly connected* if for every pair of vertices $u, v \in V$, there exists a directed path from u to v and from v to u [1], [4].

The adjacency matrix of a digraph G with $|V| = n$ is defined as a matrix $A(G) \in \{0, 1\}^{n \times n}$ with:

$$A(G)_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

It should be noted that in a digraph, the adjacency matrix does not have to be symmetric, unlike an undirected graph [1].

C. Tensor Product of Directed Graphs

The tensor product is a binary operation on graphs that yields a new graph from two given graphs. This operation is also known as the *Kronecker product* or *categorical product* [5], [6].

Definition 6 (Tensor Product)**.** Given two digraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. The *tensor product* of G_1 and G_2, denoted by $G_1 \otimes G_2$, is a digraph $H = (V, E)$ with:

- $V = V_1 \times V_2$, which is the Cartesian product of the vertex sets
- $E = \{((u_1, u_2), (v_1, v_2)) : (u_1, v_1) \in E_1 \text{ and } (u_2, v_2) \in E_2\}$

[5], [6]

Based on the definition above, the following properties are obtained.

Proposition 1. *For each vertex $(u, v) \in V(G_1 \otimes G_2)$, the following holds:*

$$\deg^+_H(u, v) = \deg^+_{G_1}(u) \cdot \deg^+_{G_2}(v) \tag{2}$$

$$\deg^-_H(u, v) = \deg^-_{G_1}(u) \cdot \deg^-_{G_2}(v) \tag{3}$$

[6]

Proposition 2. *The tensor product is commutative and associative up to isomorphism:*

$$G_1 \otimes G_2 \cong G_2 \otimes G_1 \tag{4}$$

$$(G_1 \otimes G_2) \otimes G_3 \cong G_1 \otimes (G_2 \otimes G_3) \tag{5}$$

[6]

If A_1 and A_2 are the adjacency matrices of G_1 and G_2, then the adjacency matrix of $G_1 \otimes G_2$ is the Kronecker product $A_1 \otimes A_2$ [6].

D. Vertex Complexity

The concept of vertex complexity is used to measure the computational load borne by each vertex in a digraph, particularly in the context of information processing in multi-agent systems.

Definition 7 (Vertex Complexity)**.** The *vertex complexity* $\kappa(v)$ of a vertex v in a digraph G is defined as:

$$\kappa(v) = \deg^-(v) \cdot \deg^+(v) \tag{6}$$

This value represents the number of incoming-outgoing interaction pairs that must be processed by vertex v.

Definition 8 (Total Complexity)**.** The *total complexity* of a digraph G is defined as:

$$\mathcal{K}(G) = \sum_{v \in V} \kappa(v) = \sum_{v \in V} \deg^-(v) \cdot \deg^+(v) \tag{7}$$

From the definitions of the tensor product and vertex complexity, the following important property is obtained.

Proposition 3. *For each vertex $(u, v) \in V(G_1 \otimes G_2)$, its vertex complexity is:*

$$\kappa_H(u,v) = \deg^-_{G_1}(u)\cdot\deg^-_{G_2}(v)\cdot\deg^+_{G_1}(u)\cdot\deg^+_{G_2}(v) = \kappa_{G_1}(u)\cdot\kappa_{G_2}(v) \tag{8}$$

Consequently, the total complexity of the tensor product grows multiplicatively:

$$\mathcal{K}(G_1 \otimes G_2) = \mathcal{K}(G_1) \cdot \mathcal{K}(G_2) \tag{9}$$

This growth is a major issue in large-scale multi-agent systems, motivating the use of quotient graphs as a reduction solution.

E. Quotient Graph

A quotient graph is the result of simplifying a graph by grouping behaviorally equivalent vertices into a single vertex that represents the group.

Definition 9 (Vertex Partition)**.** Given a digraph $G = (V, E)$. A *partition* $\Pi = \{V_1, V_2, \ldots, V_k\}$ of V is a collection of subsets satisfying:

- $V_i \cap V_j = \emptyset$ for every $i \neq j$ (mutually disjoint)
- $\bigcup_{i=1}^{k} V_i = V$ (covering all of V)

[7]

Definition 10 (Behavioral Equivalence)**.** Two vertices $w_1, w_2 \in V(G)$ are said to be *behaviorally equivalent*, denoted as $w_1 \sim w_2$, if and only if:

- $N^+(w_1) = N^+(w_2)$ (identical out-neighbor sets)
- $N^-(w_1) = N^-(w_2)$ (identical in-neighbor sets)

where $N^+(v)$ and $N^-(v)$ denote the out-neighbor and in-neighbor sets of vertex v, respectively [8].

Definition 11 (Quotient Graph)**.** Given a digraph $G = (V, E)$ and a partition $\Pi = \{V_1, V_2, \ldots, V_k\}$ of V. The *quotient graph* G/Π is a digraph with:

- Vertex set: $V(G/\Pi) = \{V_1, V_2, \ldots, V_k\}$
- Arc set: $(V_i, V_j) \in E(G/\Pi)$ if and only if there exist $u \in V_i$ and $v \in V_j$ such that $(u, v) \in E$

[7], [8]

Lemma 1. *Given a digraph $G = (V, E)$ and a partition Π that induces the quotient graph G/Π, the following holds:*

$$|V(G/\Pi)| \leq |V(G)|, \qquad |E(G/\Pi)| \leq |E(G)| \tag{10}$$

with equality holding if and only if Π is a trivial partition (each class contains exactly one element) [7].

The construction of a quotient graph is fundamentally a *graph homomorphism*, i.e., a mapping $\phi : V \to V(G/\Pi)$ with $\phi(u) = V_i$ for each $u \in V_i$, which preserves the adjacency relationships between vertices [7].

III. Discussion

A. Problem Formulation

Suppose there are two digraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ representing two different communication networks in a multi-agent system. Their tensor product yields a digraph $H = G_1 \otimes G_2$ with $|V_1| \cdot |V_2|$ vertices. The total complexity of H satisfies:

$$\mathcal{K}(H) = \mathcal{K}(G_1) \cdot \mathcal{K}(G_2) \tag{11}$$

This means that whenever the number of agents or network layers increases, the complexity grows multiplicatively—not additively. This becomes a serious issue when the system is large-scale. The purpose of this discussion is to demonstrate that by using a *quotient graph*, this complexity can be minimized without altering the overall behavior of the system.

B. Degree Profiles and Vertex Complexity

In a tensor product, each vertex $(u, v) \in V(H)$ is formed from a pair of one vertex in G_1 and one vertex in G_2. Its degrees follow the rules:

$$\deg^+ H(u,v) = \deg^+_{G_1}(u) \cdot \deg^+_{G_2}(v) \tag{12}$$

$$\deg^- H(u,v) = \deg^-_{G_1}(u) \cdot \deg^-_{G_2}(v) \tag{13}$$

Thus, the complexity of vertex (u, v) is:

$$\kappa_H(u,v) = \deg^-_{G_1}(u) \cdot \deg^-_{G_2}(v) \cdot \deg^+_{G_1}(u) \cdot \deg^+_{G_2}(v) \tag{14}$$

Important observation: two vertices (u_1, v_1) and (u_2, v_2) have exactly the same complexity if the degree profile $u_1 = u_2$ in G_1 and the degree profile $v_1 = v_2$ in G_2. Such vertices are candidates for consolidation in the quotient graph.

C. Construction of Quotient Graph from Tensor Product

The main idea of this method is: vertices that behave identically in the system do not need to be computed multiple times. They can be grouped into a single representative vertex via a *quotient graph*. Construction steps:

1) Determine the partition Π_1 on V_1: group vertices in G_1 that have the same in-neighbors and out-neighbors (behaviorally equivalent).
2) Determine the partition Π_2 on V_2 in the same manner for G_2.
3) Form the optimal partition on H: $(u_1, v_1) \sim (u_2, v_2)$ if and only if $u_1 \sim_{\Pi_1} u_2$ and $v_1 \sim_{\Pi_2} v_2$.
4) The quotient graph H/Π^* is formed by rendering each equivalence class as a single vertex, with arcs between classes if there is a connected pair of member vertices in H.

The number of vertices in the resulting quotient graph is $|V(H/\Pi^*)| = k_1 \cdot k_2$, where k_i is the number of equivalence classes in G_i.

D. Complexity Reduction Bounds

In general, the complexity reduction produced by a quotient graph follows the bound below.

Theorem 1 (Upper Bound on Complexity Reduction)**.** *Given $H = G_1 \otimes G_2$ with an optimal partition Π^*. The total complexity of the quotient graph satisfies:*

$$\mathcal{K}(H/\Pi^*) \leq \frac{\mathcal{K}(H)}{r_1 \cdot r_2} \tag{15}$$

where $r_i = |V_i| / k_i$ is the reduction ratio on the i-th component, which represents the average number of vertices collapsed into a single equivalence class.

Table I provides an overview of the complexity reduction for various topologies commonly encountered in MAS.

TABLE I: Comparison of complexity before and after quotient reduction for various MAS topologies

Topology G_1	*Topology G_2*	$\mathcal{K}(H)$	$\mathcal{K}(H/\Pi^*)$	*Reduction*
$\vec{C}_n$	$\vec{C}_m$	$n \cdot m$	1	$n \cdot m$ times
$\vec{C}_n$	$\vec{K}_2$	$2n$	1	$2n$ times
$\vec{P}_n$	$\vec{C}_m$	variable	≥ 1	depends on symmetry
Random graph	Random graph	$\mathcal{K}(H)$	$\approx \mathcal{K}(H)$	$\approx 1\times$

The last row in Table I shows that this method is most effective on regular and symmetric topologies, which are indeed common in structured MAS such as robot formations or sensor networks.

E. Interpretation in Multi-Agent Systems

The above results have a direct implication in the context of MAS. *Agent role equivalence.* Two agents (u_1, v_1) and (u_2, v_2) can be combined in a quotient graph if and only if they both receive information from the same group of agents and forward information to the same group of agents. In other words, they perform an identical functional role within the system. *Consensus protocol efficiency.* In consensus protocols that utilize the adjacency matrix, the state update operation:

$$x(t+1) = A(H)\, x(t) \tag{16}$$

can be replaced with a lighter version at the quotient level:

$$x^*(t+1) = A(H/\Pi^*)\, x^*(t) \tag{17}$$

where x^* is the reduced state vector. Since $|V(H/\Pi^*)| \ll |V(H)|$ on regular topologies, the computational cost per time step is significantly reduced. *Connectivity preservation.* The formation of a quotient graph through bisimulation partitioning does not destroy the essential connectivity structure of the system. If H is strongly connected, then H/Π^* is also strongly connected. Thus, properties such as information reachability and consensus convergence are preserved.

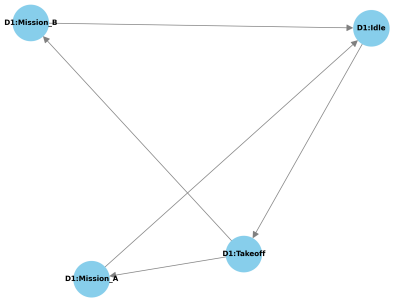
F. Application Example

To concretely illustrate the complexity reduction process, the application of the quotient graph method to a multi-agent system consisting of two autonomous drones is presented below. Each drone is modeled as a digraph with four operational *states*, then composed via the tensor product and reduced using a behavioral equivalence partition.

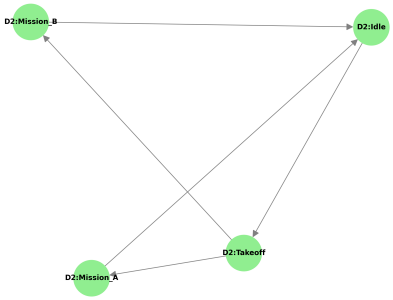
1) Component State Machine Graph Modeling: The system under test consists of two identical drones, namely Drone 1 (G_1) and Drone 2 (G_2). Each drone is modeled as a digraph with the vertex set:

$$\mathcal{V} = \{\text{Idle},\ \text{Takeoff},\ \text{Mission\_A},\ \text{Mission\_B}\}$$

The transition structure of both components is designed symmetrically: states *Mission_A* and *Mission_B* have similar in- and out-neighbor patterns—both are triggered from *Takeoff* and lead back to *Idle*. Thus, both mission states are strong candidates to be merged in the behavioral equivalence partition. The digraph structure of each component is shown in Figure 1.



(a) State Machine Drone 1 (G_1)



(b) State Machine Drone 2 (G_2)

Fig. 1: Directed graph model of the component drone state machines.

The complexity of each component is calculated as:

$$\mathcal{K}(G_1) = \mathcal{K}(G_2) = 6$$

2) Tensor Product Composition: The collective state space of the system is represented by the tensor product $H = G_1 \otimes G_2$. Based on the definition of the tensor product, the vertex set of the resulting composition is:

$$V(H) = V_1 \times V_2, \quad |V(H)| = 4 \times 4 = 16$$

with an arc $((u_1, u_2), (v_1, v_2)) \in E(H)$ if and only if $(u_1, v_1) \in E_1$ and $(u_2, v_2) \in E_2$ simultaneously. According to Theorem 1, the total complexity of the tensor product satisfies:

$$\mathcal{K}(H) = \mathcal{K}(G_1) \cdot \mathcal{K}(G_2) = 6 \times 6 = 36$$

This result is verified numerically by the implementation program (*PASS*). The tensor product graph with 16 vertices is shown in Figure 2.

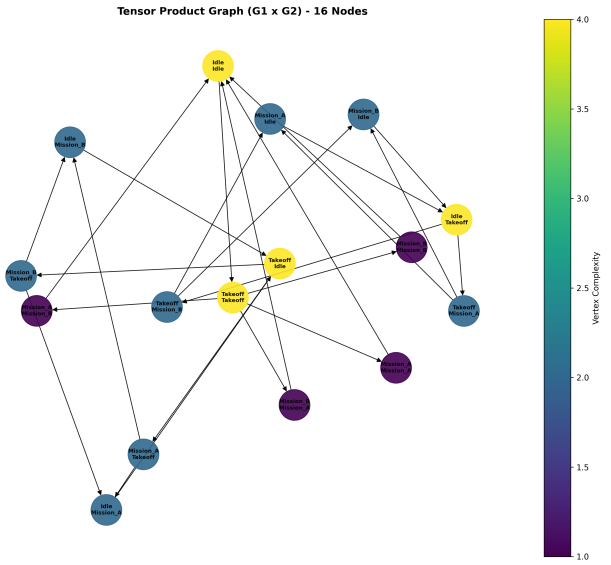


Fig. 2: Graph resulting from the tensor product $H = G_1 \otimes G_2$ with 16 vertices.

3) Reduction via Quotient Graph: To address state-space explosion, a behavioral equivalence partition Π^* is applied. Two vertices (u_1, u_2) and (v_1, v_2) are grouped into the same class if and only if they have identical in-neighbor and out-neighbor sets within H. Through this partition, the 16 vertices in H are successfully compressed into a quotient graph H/Π^* with only 9 equivalence classes (Class 0 to Class 8), as shown in Figure 3.

According to Theorem 1, the complexity of the quotient graph satisfies the upper bound:

$$\mathcal{K}(H/\Pi^*) \leq \frac{\mathcal{K}(H)}{r_1 \cdot r_2}$$

where $r_i = |V_i|/k_i$ is the reduction ratio. In this case:

$$r_1 = r_2 = \frac{4}{3} \approx 1.33 \implies \frac{\mathcal{K}(H)}{r_1 \cdot r_2} = \frac{36}{1.33 \times 1.33} \approx 20.25$$

The program calculation results show that $\mathcal{K}(H/\Pi^*) = 9$, thereby satisfying the inequality $9 \leq 20.25$ (*PASS*). A comparison of the complexity before and after reduction is presented in Figure 4.

4) Equivalence Class Decomposition: The details of the mapping from the 16 original states into the 9 equivalence classes are presented in Table II.

Class 8 in Table II provides the most representative illustration of the effectiveness of this reduction. When both

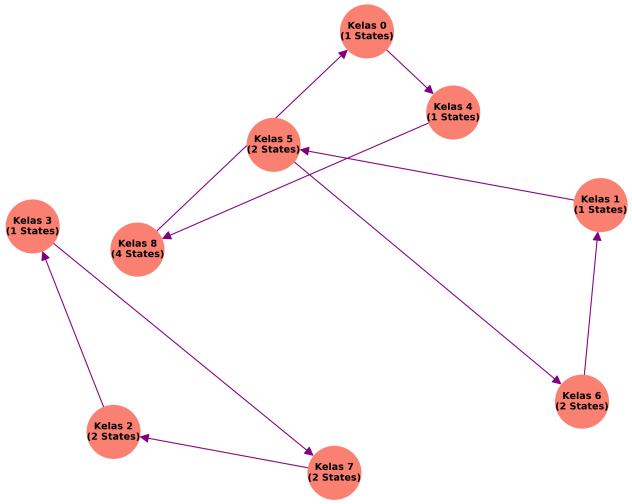


Fig. 3: Quotient graph H/Π^* resulting from symmetry reduction into 9 equivalence classes.

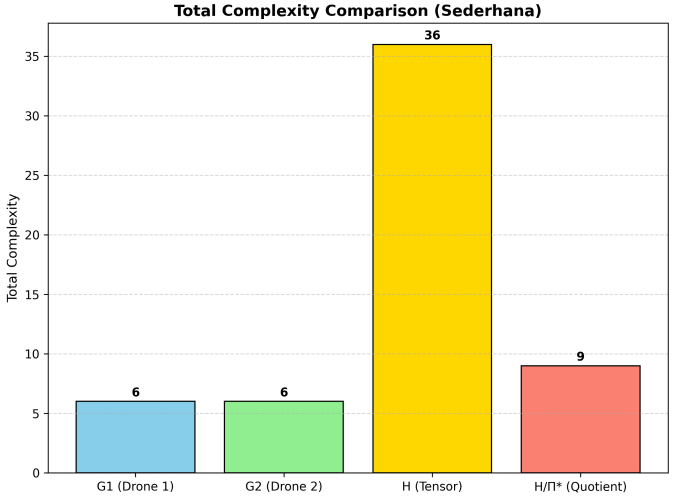


Fig. 4: Comparison of the total system complexity before and after quotient graph reduction.

TABLE II: Behavioral equivalence class decomposition on the quotient graph H/Π^*

Class	*Quantity*	*Members* (*Drone_1*, *Drone_2*)
0	1	(Idle, Idle)
1	1	(Idle, Takeoff)
2	2	(Idle, Mission_A), (Idle, Mission_B)
3	1	(Takeoff, Idle)
4	1	(Takeoff, Takeoff)
5	2	(Takeoff, Mission_A), (Takeoff, Mission_B)
6	2	(Mission_A, Idle), (Mission_B, Idle)
7	2	(Mission_A, Takeoff), (Mission_B, Takeoff)
8	4	(Mission_A, Mission_A), (Mission_A, Mission_B), (Mission_B, Mission_A), (Mission_B, Mission_B)

drones are in the mission execution phase, all variations of path combinations ((AA), (AB), (BA), (BB)) have completely identical neighbor patterns, causing these four states to be collapsed into a single macro vertex. Overall, the reduction from 16 vertices to 9 classes demonstrates that the system complexity drops from 36 to 9, which is equivalent to a computational saving of 75%, without losing the fundamental connectivity or the strongly connected property of the original graph.

G. Method Limitations

Several conditions that limit the effectiveness of this method should be noted. First, this method is most effective on digraphs that possess high symmetry. If G_1 or G_2 is an irregular graph without symmetry patterns, the resulting equivalence classes will be of size one, such that $H/\Pi^* \cong H$ and no reduction occurs. Second, not all graph properties are preserved by the quotient. For instance, the number of simple paths of a certain length may change. Therefore, users need to ensure that the properties relevant to their system—such as strong connectivity and communication reachability—belong to the class of properties preserved by the graph homomorphism used. Third, for MAS whose topology changes dynamically (agents joining or leaving the network), the bisimulation partition must be recomputed each time a topological change occurs. This introduces a computational overhead that must be accounted for in real-world implementations.

IV. Conclusion

This study examined a quotient graph abstraction founded on behavioral equivalence as a mechanism for reducing vertex complexity in tensor product digraphs arising from multi-agent systems. Since the overall complexity of a tensor product network increases according to $\mathcal{K}(H) = \mathcal{K}(G_1) \cdot \mathcal{K}(G_2)$, controlling state-space growth becomes essential when dealing with large collections of interacting agents. The proposed partitioning strategy Π groups vertices exhibiting identical incoming and outgoing neighborhood structures, producing a reduced representation that satisfies the theoretical bound $\mathcal{K}(H/\Pi^) \leq \mathcal{K}(H)/(r_1 \cdot r_2)$. Application of the method to a two-drone system, where each vehicle operates over four distinct states, demonstrated a reduction in total complexity from 36 to 9. This corresponds to a computational saving of 75%. Equally important, the quotient construction preserves fundamental topological characteristics through the associated graph homomorphism, particularly strong connectivity, thereby maintaining the integrity of information flow throughout the network. The effectiveness of the reduction is strongly influenced by the structural properties of the underlying topology. Highly symmetric graphs provide substantial compression, whereas irregular or randomly organized networks offer comparatively limited opportunities for simplification. Subsequent work may extend the proposed framework to layered multi-component architectures capable of representing richer patterns of agent interaction. Another promising direction involves the investigation of approximate bisimulation techniques for networks that exhibit partial symmetry rather than exact structural equivalence. Future studies may also explore the incorporation of quotient-based reductions directly into distributed consensus algorithms to further improve computational efficiency in large-scale multi-agent environments.

Acknowledgment

The author wishes to express profound gratitude to God Almighty for His guidance, mercy, and blessings throughout the learning process and the completion of this paper. Sincere appreciation is also extended to the lecturer of the Discrete Mathematics course (IF2120) at Institut Teknologi Bandung, Prof. Dr. Ir. Rinaldi, MT., whose expertise, instruction, and insightful guidance greatly contributed to the author's understanding of the subject matter. The author is likewise thankful to family members and friends for their unwavering encouragement, patience, and support during the preparation of this work. This paper was developed to fulfill the academic requirements of the Discrete Mathematics course and serves as an exploratory study on the application of algebraic graph-theoretic concepts within the domain of multi-agent systems.

A. References

References

[1] R. Munir, *Matematika Diskrit: Teori Graf*, Lecture Slides, Informatics Engineering Study Program, Institut Teknologi Bandung, Bandung, 2023. [Online]. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/

[2] F. Harary, *Graph Theory*. Reading, MA: Addison-Wesley, 1969.

[3] J. A. Bondy and U. S. R. Murty, *Graph Theory*. New York, NY: Springer, 2008.

[4] J. Bang-Jensen and G. Z. Gutin, *Digraphs: Theory, Algorithms and Applications*, 2nd ed. London: Springer, 2008.

[5] P. M. Weichsel, "The Kronecker product of graphs," *Proc. Amer. Math. Soc.*, vol. 13, no. 1, pp. 47–52, 1962.

[6] R. Hammack, W. Imrich, and S. Klavžar, *Handbook of Product Graphs*, 2nd ed. Boca Raton, FL: CRC Press, 2011.

[7] P. Hell and J. Nešetřil, *Graphs and Homomorphisms*. Oxford: Oxford University Press, 2004.

[8] R. Milner, *Communication and Concurrency*. Englewood Cliffs, NJ: Prentice Hall, 1989.

Declaration

The author hereby declares that this paper, entitled "Minimizing Vertex Complexity of Tensor Product Digraphs via Quotient Graph for Multi-Agent Systems," is an original piece of work written and conducted independently. All ideas, formulations, and data derived from other sources have been properly cited and acknowledged in accordance with standard academic practices.

Bandung, June 19, 2026

Muhammad Ridwan Nasir Firdaus
NIM: 13525096