Volume 11, Issue 5, May – 2026 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1812

Structural Conditions and Algorithms for Cycle–Star Decompositions of Complete Bipartite Graphs

Imran R.[1]; Ilayaraja M.[2]; Bhuvaneswari R.[3]; Abiramasundari C.[4]

[1;2;3;4]Department of Mathematics Sona College of Arts and Science, Salem-5, India

Publication Date: 2026/06/05

Abstract: This paper establishes necessary and sufficient conditions for the existence of decompositions of complete bipartite graphs into cycles and stars in certain classes of cases. Explicit constructions are provided for all admissible parameter sets. In addition, a Python implementation is developed to verify the theoretical conditions and to generate corresponding decompositions.

How to Cite: Imran R.; Ilayaraja M.; Bhuvaneswari R.; Abiramasundari C. (2026) Structural Conditions and Algorithms for Cycle–Star Decompositions of Complete Bipartite Graphs. *International Journal of Innovative Science and Research Technology*, 11(5), 3277-3281. https://doi.org/10.38124/ijisrt/26may1812

I. INTRODUCTION

Graph decomposition is a central topic in graph theory, with applications in design theory, network analysis, and algorithmic graph theory. Complete bipartite graphs play an important role in decomposition problems. The complete bipartite graph $K_{m,n}$ consists of two disjoint vertex sets of sizes m and n, where each vertex in one part is adjacent to every vertex in the other. Owing to their regular structure, such graphs serve as natural objects for studying both theoretical and constructive aspects of graph decompositions.

In this paper, we investigate decompositions of complete bipartite graphs into cycles and stars. A cycle is a connected 2-regular graph, and a star is a tree isomorphic to $K_{1,t}$ for some integer $t \geq 1$. The main objective is to determine necessary and sufficient conditions for the existence of such decompositions and to provide explicit construction methods.

II. PRELIMINARIES

In this section, we introduce basic definitions and notation used throughout the paper.

- Definition 2.1. A *cycle* is a connected 2-regular graph. A cycle of length k is denoted by C_k.
- Definition 2.2. A *star* is a tree isomorphic to the complete bipartite graph $K_{1,t}$ for some integer

$t \geq 1$. The unique vertex of degree t is called the *center* of the star.

- Definition 2.3. Let G = (V, E) be a graph. A *decomposition* of G is a collection of subgraphs $\{G_1, G_2, \ldots, G_k\}$ such that:

✓ $E(G_i) \cap E(G_j) = \emptyset$ for all $i \neq j$, and
✓ $E(G) = \bigcup_{i=1}^{k} E(G_i)$.

- Definition 2.4. A *cycle-star decomposition* of a graph G is a decomposition in which each subgraph is either a cycle or a star.
- Definition 2.5. The *complete bipartite graph* Km,n is a graph whose vertex set can be partitioned into two disjoint sets of sizes m and n, such that every vertex in one set is adjacent to every vertex in the other.

➢ *Main Results*

In this section, we present necessary and sufficient conditions for the existence of cycle–star decompositions of complete bipartite graphs.

➢ *Necessary Conditions for* $\{pC_4, qS_4\}$ *−Decomposition*

- $K_{2,n}$ with n even: q must be even.
- $K_{4,n}$ with $n \geq 4$: $p \neq 1$ and $q \neq 1$.
- $K_{m,n}$ with $m, n \geq 6$ even: $q \neq 1$.
- Exactly one partition odd:

✓ m odd, $n \equiv 0 \ (mod\ 4)$: $q \geq n/4$.
✓ n odd, $m \equiv 0 \ (mod\ 4)$: $q \geq m/4$.

➢ *Main Theorem*

Let $m, n \in \mathbb{Z}^+$ with $m \leq n$ and $p, q \geq 0$. $K_{m,n}$ admits a $\{pC_4, qS_4\}$ −decomposition if and only if one of the following holds:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1812

- $m = 2$ even, n even: q even.
- $m = 4, n \geq 4$ even: $p, q \neq 1$.
- $m, n \geq 6$ even: $q \neq 1$.
- Exactly one partition odd:

✓ m odd, $n \equiv 0 \ (mod\ 4)$: $q \geq n/4$.
✓ n odd, $m \equiv 0 \ (mod\ 4)$: $q \geq m/4$.

➢ *Algorithm for $\{pC_4, qS_4\}$ −Decomposition of $K_{m.n}$*

We describe an algorithm that determines whether a complete bipartite graph $K_{m,n}$ admits a decomposition into p cycles C_4 and q stars S_4, and constructs such a decomposition whenever possible.

- *Algorithm 1: $\{pC_4, qS_4\}$ −Decomposition of $K_{m.n}$*

	REQUIRE: Integers m, n (part sizes), integer q (number of stars)
	ENSURE: Lists of cycles and stars forming a decomposition, or report impossible
1.	Compute total number of edges: $total\_edges = m \times n$
2.	Check divisibility: if $(total\_edges - 4q) \bmod 4 \neq 0$, **return** "Decomposition impossible"
3.	Set $p = \frac{tota\_edges - 4q}{4}$
4.	Check structural conditions (small cases, parity, forbidden values):
5.	if conditions fail, **return** "Decomposition impossible"
6.	Initialize vertex sets:
7.	$X = \{x_1, \ldots, x_m\}, \quad Y = \{y_1, \ldots, y_n\}$
8.	Initialize empty sets: $C4\_list, S4\_list$, used_edges
9.	**Construc**t p **copies** of **C_4:**
10.	**for** each of the p cycles **do**
11.	Select two vertices $x_i, x_{i+1} \in X$ and two vertices $y_j, y_{j+1} \in Y$
12.	Form 4-cycle: $(x_i, y_j), (x_{i+1}, y_j), (x_{i+1}, y_{j+1}), (x_i, y_{j+1})$
13.	Add cycle to $C4\_list$ and mark edges as used
14.	Update indices i, j with wrap-around as needed
15.	**end for**
16.	**Construct q copies of** S_4:
17.	Let remaining_edges = edges not in $C4\_list$
18.	**while** remaining_edges is not empty **do**
19.	Find a vertex in $X \cup Y$ with degree ≥ 4 in remaining edges
20.	Form a star with 4 incident edges
21.	Add star to S4_list and remove edges from remaining edges
22.	**if** no such vertex exists **then**
23.	Handle fallback (group remaining 4 edges if possible)
24.	**end if**
25.	**end while**
26.	**return** C4_list, S4_list

➢ *Remarks on Correctness*

- The algorithm first checks the necessary edge-count condition $mn = 4p + 4q$. If this fails, decomposition is impossible.
- Structural feasibility conditions include parity checks, forbidden small cases ($e.g., p = 1$ or $q = 1$ in small graphs), and minimum size requirements for m and n.
- Each C_4 is constructed using disjoint pairs of vertices, ensuring edge-disjoint cycles.
- Remaining edges are grouped into S_4 stars by selecting vertices with degree at least 4 . This ensures all edges are covered.
- The algorithm terminates in $O(mn)$ time.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1812

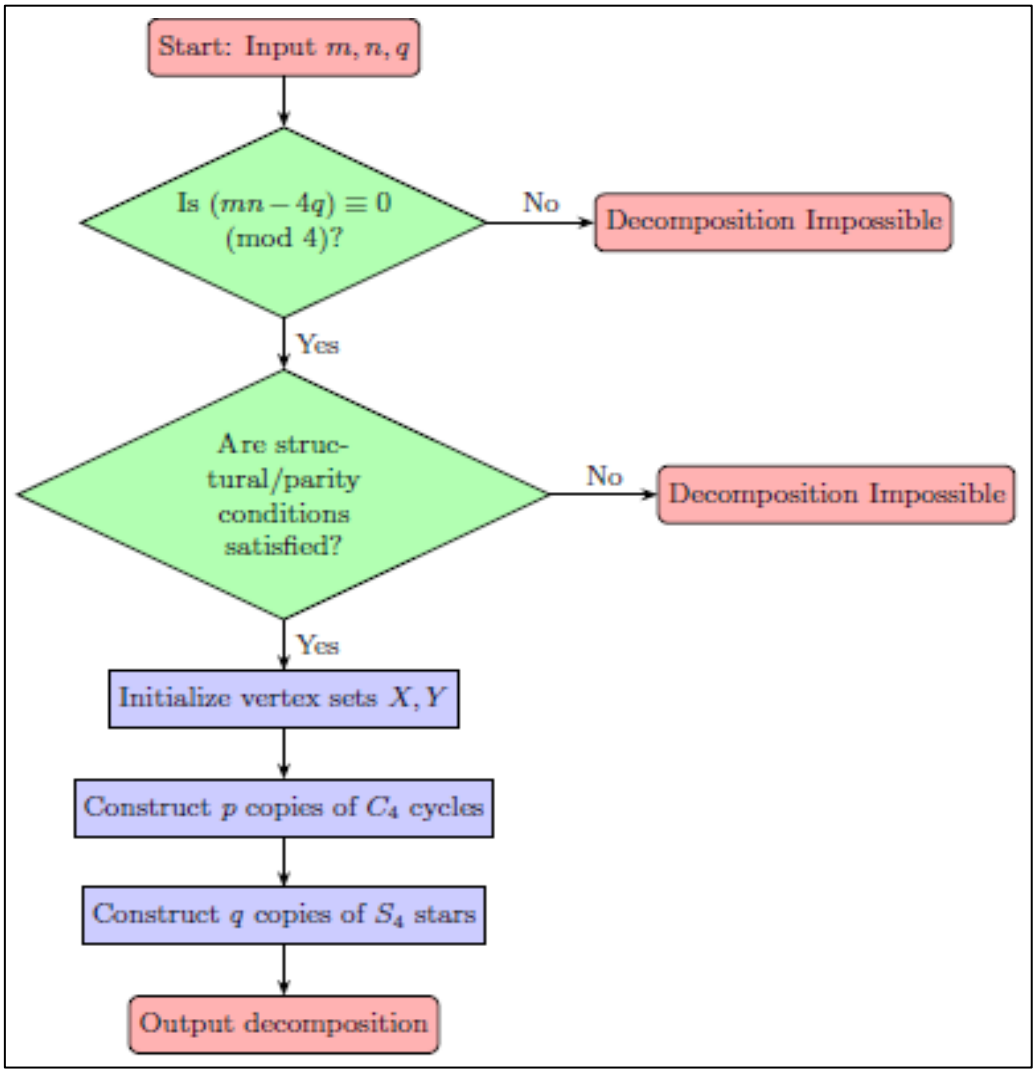

Fig 1 Flowchart for the Decomposition of a Bipartite Graph into C_4 Cycles and S_4 Stars

III. PYTHON PROGRAM FOR DECOMPOSITION

In this section, we present a Python program that verifies the existence of a $C_4 - S_4$ decomposition of the complete bipartite graph $K_{m,n}$ and constructs such a decomposition whenever possible. The program first checks the necessary conditions and then explicitly generates the cycles and stars.

```
def is_decomposition_possible(m, n, q):
total_edges = m * n

# Check edge count condition
if (total_edges - 4*q) % 4 != 0:
return False, None

p = (total_edges - 4*q) // 4

# Special cases
if m == 2 and n % 2 == 0 and q % 2 == 0:
p = n // 2
return True, p

if m == 4 and n >= 4 and q != 1 and p != 1:
return True, p

if m >= 6 and n >= 6 and q != 1:
return True, p

if (m % 2 == 1 and n % 4 == 0 and q >= n // 4) or \
(n % 2 == 1 and m % 4 == 0 and q >= m // 4):
return True, p

return False, None


def decompose_Kmn_C4_S4(m, n, q):
possible, p = is_decomposition_possible(m, n, q)
if not possible:
raise ValueError("Decomposition not possible for given parameters")

# Vertex sets
X = [f"x{i+1}" for i in range(m)]
Y = [f"y{i+1}" for i in range(n)]

# All edges of K_{m,n}
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1812

```
total_edges = {(x, y) for x in X for y in Y}
used_edges = set()

C4_list = []
S4_list = []

# Construct p copies of C4
i = j = 0
for _ in range(p):
x1, x2 = X[i], X[i+1]
y1, y2 = Y[j], Y[j+1]
cycle = [(x1, y1), (x2, y1), (x2, y2), (x1, y2)]
C4_list.append(cycle)
used_edges.update(cycle)

j += 2
if j >= n - 1:
j = 0
i += 2
if i >= m - 1:
i = 0

# Remaining edges for S4 stars
remaining_edges = list(total_edges - used_edges)

from collections import defaultdict
adj_X = defaultdict(list)
adj_Y = defaultdict(list)

for x, y in remaining_edges:
adj_X[x].append(y)
adj_Y[y].append(x)

# Construct S4 stars
while remaining_edges:
made_star = False

for x in X:
if len(adj_X[x]) >= 4:
leaves = adj_X[x][:4]
star = [(x, leaf) for leaf in leaves]
S4_list.append(star)
for leaf in leaves:
adj_X[x].remove(leaf)
adj_Y[leaf].remove(x)
remaining_edges.remove((x, leaf))
made_star = True
break

if made_star:
continue

for y in Y:
if len(adj_Y[y]) >= 4:
leaves = adj_Y[y][:4]
star = [(leaf, y) for leaf in leaves]
S4_list.append(star)
for leaf in leaves:
adj_Y[y].remove(leaf)
adj_X[leaf].remove(y)
remaining_edges.remove((leaf, y))
made_star = True
break

if not made_star:
# Fallback grouping (theoretical cases should avoid this)
S4_list.append(remaining_edges[:4])
for edge in remaining_edges[:4]:
remaining_edges.remove(edge)
break

return C4_list, S4_list
```

The function is_decomposition_possible implements the theoretical conditions established in Section 3. The function decompose_Kmn_C4_S4 then constructs the corresponding decomposition by first generating C_4 cycles and subsequently partitioning the remaining edges into S_4 stars.

IV. EXAMPLES

In this section, we illustrate the implementation of the algorithm on several instances of complete bipartite graphs. The examples demonstrate both feasible and infeasible cases of $\{pC_4, qS_4\}$ −decompositions.

- *Example 1:* $K_{2,6}$ *with* $q = 0$

```
m, n, q = 2, 6, 0
C4s, S4s = decompose_Kmn_C4_S4(m, n, q)
print("C4 cycles:", C4s)
print("S4 stars:", S4s)
```

- *Output.* The graph $K_{2,6}$ admits a decomposition into three C_4 cycles and no S_4 stars.

- *Example 2:* $K_{4,4}$ *with* $q = 1$

```
m, n, q = 4, 4, 1
try:
C4s, S4s = decompose_Kmn_C4_S4(m, n, q)
except ValueError as e:
print(e)
```

- *Output.* No decomposition exists, as the parameters violate the necessary condition $p, q \neq 1$.

- *Example 3:* $K_{6,6}$ *with* $q = 2$

```
m, n, q = 6, 6, 2
C4s, S4s = decompose_Kmn_C4_S4(m, n, q)
print("C4 cycles:", C4s)
print("S4 stars:", S4s)
```

- *Output.* The graph $K_{6,6}$ admits a decomposition into eight C_4 cycles and two S_4 stars.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1812

➢ *Example 4:* $K_{3,8}$ *with* $q = 2$

```
m, n, q = 3, 8, 2
C4s, S4s = decompose_Kmn_C4_S4(m, n, q)
print("C4 cycles:", C4s)
print("S4 stars:", S4s)
```

- *Output.* Since one partite set has odd cardinality, the algorithm verifies the parity condition and produces a valid decomposition consisting of C_4 cycles and two S_4 stars.

V. CONCLUSION

In this paper, we established necessary and sufficient conditions for the existence of $\{pC_4, qS_4\}$-decompositions of complete bipartite graphs in several special cases. These conditions were translated into an explicit algorithm implemented in Python.

The program verifies all edge-count and parity constraints and constructs the required decomposition whenever it exists. The examples confirm the correctness of the theoretical results and demonstrate the effectiveness of the algorithm. Future work may explore extensions to larger cycle lengths, other classes of trees, or more general bipartite graphs.

REFERENCES

[1]. Alspach, B., Gavlas, H. (2001). Cycle decompositions of K_n and $K_n - I$. *Journal of Combinatorial Theory, Series B*, 81(1), 77–99.
[2]. Bondy, J. A., Murty, U. R. S. (1976). *Graph Theory with Applications*. Macmillan Press, New York.
[3]. Jeevadoss, S., Muthusamy, A. (2014). Decomposition of complete bipartite graphs into paths and cycles. *Discrete Mathematics*, 331, 98–108.
[4]. M. Ilayaraja and A. Muthusamy. Decomposition of Complete Graphs. *Indian Journal of Discrete Mathematics*, 4 (2018), 13–33.
[5]. M. Ilayaraja and A. Muthusamy. Decomposition of Complete Bipartite Graphs into Cycles and Stars with Four Edges. *AKCE International Journal of Graphs and Combinatorics*, 17 (2020), 697–702.