LEO Satellite Beam Management Algorithms

Oren Markovitz and Michael Segal
School of Electrical and Computer Engineering
Ben-Gurion University, Beer-Sheva, Israel

***Abstract*—A global service LEO constellation aims to provide service to any terminal covered by the constellation planes. To reduce the satellite cost, which is directly related to its power usage and weight, each satellite should be able to service (cover) all the terminals within its Field-of-View (FoV) using the satellites beams in the most efficient way. LEO satellite network vendors use different approaches to handle the user terminals diverse locations. A stepping (or tracking) beam constellation provides service to predefined areas instead of a full coverage. As a result, satellites using stepping beams are more efficient, as power is used only for populated areas. When the constellation shifts, beams are allocated such that each area is serviced by one of the satellites that has the area in its FoV. Stepping beams constellations raise unique and complicated tasks. When admitting a new service area, we should verify it can be covered by a satellite beam at any coverage combination of the moving constellation. The algorithms should take into account the satellite limitations (power, number of beams).**

Previous works analyzed the overall efficiency of each constellation architecture and the technologies of the on-board beams, while little attention was paid to the algorithm that validates a consistent coverage of new service areas. The major contribution of this paper is a novel algorithm for validating new service areas in a LEO stepping beam constellation.

***Index Terms*—LEO, stepping beam, tracking beam, satellite, user beams.**

I. Introduction

A LEO constellation must include several hundreds to several thousands satellites to achieve a reliable and continuous service of terminals. A satellite has two types of beams. 'GW beams' are used to communicate with the gateways (GWs) and 'user beams' are used to communicate with the terminals. The reminder of the paper refers only to user beams (or simply beams). LEO satellite network vendors use different approaches to meet the user terminals diverse locations. In tiles (or stripes) coverage, the footprint of the satellite's user beams covers the satellite's Field-of-View (FoV) at a fixed pattern. As the satellite moves, the beams are 'dragged'. The constellation is planned such that the satellites FoV covers the service area (e.g. for polar constellation – global coverage) and the entire constellation service area is covered with user beam tiles at any given time. Tile coverage is used by OneWeb.

In most (if not all) cases the satellites FoV is not fully populated with terminals. In case of a stepping beam satellite, the satellite points its beams to cover only the areas that need to be serviced (Figure 1). As the satellite moves, one of the beams is released (the service area is covered by other satellites) and is pointed to a new location. TeleSat claims to use a DRA (Direct Radio Array). Each satellite's DRA will be able to form at least 16 beams in the uplink and downlink

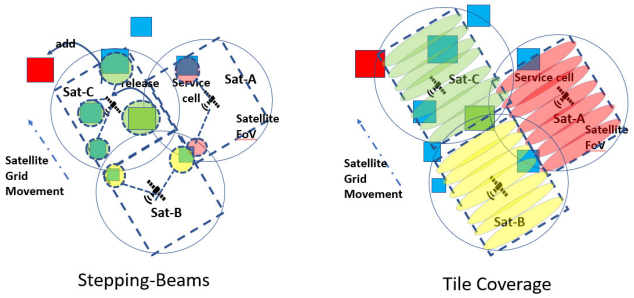

Fig. 1: Stepping Beams vs. Tile Coverage.

directions. Stepping beams are more efficient as the satellite's power is used only for populated service areas; however, managing the beams is more complicated. When adding a new service area we should verify it can be covered and provided with sufficient power by a user beam at any coverage combination generated by the movement of the constellation.

Previous works analyzed overall efficiency of each approach and the technologies of the on-board beams. Unfortunately, the algorithms for managing the stepping beams and validating a new service area (given a constraint on maximum number of user beams and the total power of each satellite) were not investigated. In this paper we propose a novel low complexity ($O(m^3)$) stepping beam validation algorithm for guaranteeing that a new service area can be covered at any given time using up to m user beams and total power p for each satellite.

The paper is organized as follows: Section II reviews the related work. A model of the constellation is presented in section III. A novel service cell validation algorithm is shown and analyzed in section IV. A simulation of the algorithm and analysis of the results is detailed in Section V. Section VI summarizes the benefits of the algorithm introduced in this paper and suggests directions for future research.

II. Related Work

Current work focuses on different services and coverage aspects of LEO constellations starting from the satellite design [9], link budget and beam characteristics [2], geographical and population coverage [4], [6], [7], total capacity [2], [3] and the number of devices that can be served in a LEO constellation given the handover period and log on time in case of TDMA return channel [8].

In [1], the paper analyzes the rationality of two different beam coverage designs of oneWeb and SpaceX. Papers [2] and

978-1-6654-2854-5/21/$31.00 ©2021 IEEE

[3] compare Cyber Star, Spaceway and Celesti, by evaluating the capacity and signal reduction. The authors in [4] and [6] compare the capacity provided by OneWeb, SpaceX, and TeleSat using a statistical framework and provide a performance estimation of the system throughput. A geometric subdivision method for obtaining the coverage target area and capacity for each satellite of a constellation is presented in [7]. Zhou et al. [8] evaluate the coverage by latitude and how many IoT devices can be accessed simultaneously when there is a change in coverage. In addition, the number of devices in a TDMA return channel constellation is analyzed. Katona et al. [9] provide a reference design for developing a Ka-band satellite.

The problem of validating a new service cell can be easily reduced to a graph clique problem. The service cells are mapped to graph nodes, edges between nodes are added when the distance between them is smaller than FoV edge size. When a clique size is larger than the number of satellite user beams (m), there exists a scenario in which a FoV rectangle includes more than m service cells, i.e a satellite is required to serve more than m service cells. The clique problem is considered NP-hard. Max clique size algorithms in geometrical graphs [14] provide a complexity of $O(|E| \cdot |V| + |V|^2)$ where $|E|$ and $|V|$ are the cardinalities of a geometrical graph's edge set and vertex set, respectively. In order to validate each service addition, we need to run the algorithm $|V|$ times. Our approach takes advantage of the geographical properties of the problem (i.e. the maximum clique size is m and the maximum distance between clique members which is known in advance).

III. Constellation Modeling

In our model, the terminals are mapped into service cells rectangles, and the vertices of these rectangles are defined by geographical altitude and longitude degrees. Each service cell is served by one or more user beams from one or more satellites at a given time. When a service cell is partially or fully covered by the satellites FoV, it is assigned to a user beam.

The satellite FoV circle outlines the ground area as seen from the satellite. In order to provide a continuous service, the FoVs of adjacent satellites must overlap. The Continuous Field-of-View (CFoV) of each satellite is defined by either a hexagon or a tile that is contained in the satellite's FoV (see Figure 2).

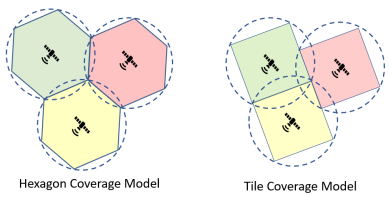

Fig. 2: Satellite Continuous Field-of-View Models.

For simplicity, we use the tile model throughout this paper. However, the algorithms can be easily adjusted to the hexagon model. In addition, we rotate the axes for simpler representation.

When we move away from the equator, there is an increased overlap between the satellites physical FoV, which in turn provides additional bandwidth for terminals in the overlapped sections. A different approach is to define the horizontal and vertical length of the CFoV in degrees, therefore the size of the CFoV remains fixed as we move away from the equator, while the power density (and rate) that each satellite can provide to the terminals in its CFoV is increased. In this paper, we take the latter approach, which is simpler to manage and provides the same capacity benefits.

A. Geo-Cells Array

We split the earth into 'geo-squares', each square size is equal to the CFoV size (Figure 3). When the CFoV vertical and horizontal lengths are measured in degrees, the number of squares is the same as the number of satellites, i.e. for a constellation with n planes there are n^2 geo-squares of equal size. A two-dimensional array of geo-cells is used to store the service cells in each geo-square. The array's indexes are calculated based on the geo-cells (geo-square) location (longitude and latitude). The service cells located in the geo-cell are stored in a linked-list ordered by unique ID. When a service cell is split between two geo-cells, it will be stored twice. The array is easily accessed in $O(1)$ time by translating a given longitude and latitude coordinates to two dimensional array indexes $(x, y) \implies$ geo-cell(x', y').

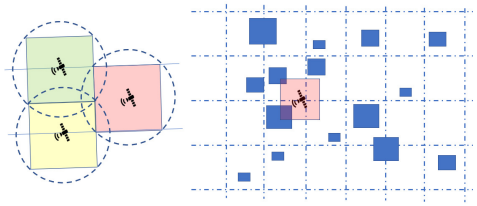

Fig. 3: Service Cells and Geo-Cells.

IV. Service Cell Validation

A terminal can be added to an existing service cell or to a new one. We assume a service cell rectangle is defined and validated before being used by the satellites (i.e. offline). When adding a new service cell, we must verify that the number of service cells covered by any satellite at any constellation coverage scenario, does not exceed the number of satellite user beams (m). We assume the service cell rectangle maximum edge sizes are equal to those of the satellites CFoV rectangle ($CFoV_D$).

Algorithm outline: The algorithm builds a subset of the service cells that should be considered (the service cells located inside the 'reference-zone'). The algorithm locates all the relevant satellite positions within the 'reference-zone', for which the satellite CFoV contains the new service cell (or part of which). For each position of the satellite CFoV rectangle, we verify that it does not include more than m members and

that the total power consumed by the service cells does not exceed p. The new service cell can be added if there are no relevant CFoV rectangles for which the number of beams or power constraints are violated. We prove that the reference zone is contained in a rectangle of 16 geo-cells were the new service cell is located at the center (Figure 4).

Lemma 1. *If the number of members in all the CFoV rectangles is $\leq m$, then there is no scenario in which a satellite has more than m service cells in its CFoV.*

A. Service Cell Reference-Zone

The service cell reference-zone outlines the area that needs to be considered when validating a service cell.

Lemma 2. *The maximum edge size of the reference-zone is $3 \cdot CFoVD$ and the number of service cells in the reference-zone is smaller than $9 \cdot m$. See Figure 4.*

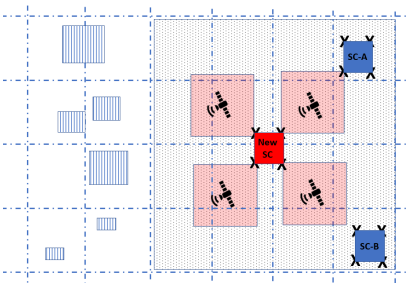

Fig. 4: Service cell Reference-Zone.

B. Reference-Zone Algorithm

The algorithm returns the list that contains the relevant service cells in the reference-zone of a new service cell in $O(m)$.

The algorithm models the service reference-zone as a set of squares and builds the list of the service cells in the service reference-zone. The service cells locations are stored in a two-dimensional geo-cells array $SCA[1 \ldots n, 1 \ldots n]$ (as described in section III) . Each entry stores up to m service cells (Lemma 3). The service cells are stored in a linked list. The array is stored on each satellite. The distribution algorithm is out of the scope of this paper. However, we assume the distribution algorithm provides coherency.

Let:

- k be the number of service cells.
- m be the number of user beams in each satellite.
- $CFoV_D$ be the edge size of the satellites CFoV.
- $SC_{i,(x,y,r)}$ be a service cell rectangle with a center location of x, y and an edge size of $2 \cdot r$.
- $SCA[1 \ldots n, 1 \ldots n]$ be an array of service cells sets initialized to $\{\}$.

Algorithm outline:

1) The service cells are stored in a two-dimensional array such that each cell covers an area of size $CFoV_D \times CFoV_D$. Each record stores all the service cells that have a center in the same geo-cell.
2) The algorithm uses the array to build a list of service cells that are stored in the same record as the new service cell and the adjacent records (i.e. reference-zone records). The algorithm returns all the service cells located in one of the reference-zone records.

---

getReferenceZone(x, y, r) Input: Reference-zone center x, y with edge size of $2 \cdot r$.
Return: $\{SC_1, ..., SC_{16 \cdot m}\}$.

1) $referencezoneRecords = \{\}$.
2) $CFoV_D = \frac{360}{n}$.
3) x'=$\lfloor \frac{x - r}{CFoV_D} \rfloor$, y'=$\lfloor \frac{y - r}{CFoV_D} \rfloor$.
4) Append to $referencezoneRecords$: $SCA[x', y'], SCA[x' - 1, y'], SCA[x' - 1, y' - 1], SCA[x', y' - 1]$.
5) x'=$\lfloor \frac{x - r}{CFoV_D} \rfloor$,$y'$=$\lfloor \frac{y + r}{CFoV_D} \rfloor$.
6) Append to $referencezoneRecords$: $SCA[x', y'], SCA[x' - 1, y'], SCA[x' - 1, y' + 1], SCA[x', y' + 1]$.
7) x'=$\lfloor \frac{x + r}{CFoV_D} \rfloor$, y'=$\lfloor \frac{y - r}{CFoV_D} \rfloor$.
8) Append to $referencezoneRecords$: $SCA[x', y'], SCA[x' + 1, y'], SCA[x' + 1, y' - 1], SCA[x', y' - 1]$.
9) x'=$\lfloor \frac{x + r}{CFoV_D} \rfloor$, y'=$\lfloor \frac{y + r}{CFoV_D} \rfloor$.
10) Append to $referencezoneRecords$: $SCA[x', y'], SCA[x' + 1, y'], SCA[x' + 1, y' + 1], SCA[x', y' + 1]$.
11) Remove* duplicates from $referencezoneRecords$
12) For each $SCA[i, j] \in referencezoneRecords$, add the service cells to $referencezone$.
13) Remove* duplicates from $referencezone$
14) Return $referencezone$.

* Removing duplicates is done by setting a flag for every record that is added for the first time, and checking the flag when adding records.

Lemma 3. *The number of service cells stored in each cell of the array is smaller than the number of user beams (m).*

Proof. A service cell is approved only if it can be served by the constellation at any given positioning of the constellation. In particular, when the constellation is aligned with the geo-cells, the satellite CFoV is contained within the geo-cell. Assuming the number of service cells in a geo-cell is larger than m, then in the latter scenario, the number of service cells in the satellite CFoV is bigger than m. □

Lemma 4. *$referencezoneRecords$ returns all the records that are at maximum index difference (± 1) from the new service cell record indexes.*

Lemma 5. *A service cell that is located in a CFoV rectangle with any vertex of the new service cell is returned by the algorithm.*

Proof. We prove the lemma for the upper right vertex of the new service cell but the same proof applies for all vertices. Let $F1_{(x,y)}, ..., F4_{(x,y)}$ be the vertices of the satellite CFoV rectangle. Let $S1_{(x,y)}, ..., S4_{(x,y)}$ be the vertices of the service cell that is served by the satellite. Let c be the upper right vertex of the new service cell. If two service cells belong to the same CFoV, then there are at least two vertices (each from a different service cell) for which the horizontal and vertical distances are smaller than $CFoV_D$. Let $S_i \in \{S_1, S_2, S_3, S_4\}$ be the vertex at a (horizontal and vertical) distance smaller than $CFoV_D$ from c. Let c_x, c_y and s_x, s_y be the locations of c,s_i respectively such that c is stored in array cell $[\frac{c_x}{CFoV_D}, \frac{c_y}{CFoV_D}]$, and s_i is stored in array cell $[\frac{s_x}{CFoV_D}, \frac{s_y}{CFoV_D}]$. The distance between x indexes of the record that stores c and the one that stores s_i is $\frac{|c_x - s_x|}{CFoV_D}$, and the distance between y indexes is $\frac{|c_y - s_y|}{CFoV_D}$. Since the maximum distance between the coordinates is $CFoV_D$, then the absolute value of the difference between x (y) indexes is at most 1. By Lemma 4, $referencezoneRecords$ returns all the records that are at maximum index difference plus/minus one of the new service cell record indexes. The algorithm returns all the service cells that are stored in records of the $referencezoneRecords$, therefore, all the service cells that are located at a distance (horizontal and vertical) equal to or smaller than $CFoV_D$ are returned by the algorithm. □

Lemma 6. *$getReferenceZone$ computation complexity is bounded by $O(m)$.*

C. Service Cell Validation Algorithm

The algorithm calculates all the relevant positions of the CFoV rectangle inside the reference-zone, that include at least one of the new service cell vertices. For each of these locations, we verify that it does not include more than m service cells and that the total power required by the service cells does not exceed p.

Algorithm 1 List of Relevant CFoV Positions Algorithm

$getRelevantCFoVList(SCList)$
Input: List of service cells.
Return: List of relevant CFoV positions $\{(x_1, y_1), ..., (x_{m^2}, y_{m^2})\}$.

1) $xList$ =[], $yList$ = []
2) For each SC in $SCList$: $xList$.append($SC_x \pm r$), $yList$.append($SC_y \pm r$).
3) Order $xList$, order $yList$.
4) Remove duplicates from $xList$, $yList$
5) $list$ = [].
6) For x in $xList$:
   For y in $yList$: $list$.append((x,y)).
7) Return $list$.

Lemma 7. *Given a set of points (Figure 5), there exists a rectangle that contains all the points, such that the vertices of the rectangle are a combination of the x and y coordinates of two of these points.*

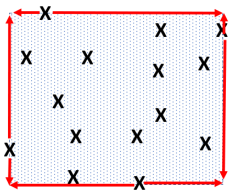

Fig. 5: Vertex of a Group of Points.

Proof. Let $p_1, ..., p_n$ be a group of points with coordinates $(x_1, y_1), ..., (x_n, y_n)$ respectively.
Let:
$x_{min} = min\{x_1, ..., x_n\}$, $x_{max} = max\{x_1, ..., x_n\}$.
$y_{min} = min\{y_1, ..., y_n\}$, $y_{max} = max\{y_1, ..., y_n\}$.
The lower left and right upper vertices of the rectangle that contain all points are (x_{min}, y_{min}),(x_{max}, y_{max}) respectively. □

Lemma 8. *$getRelevantCFoVList$ returns all possible rectangle vertices in the reference-zone in $O(m^2)$.*

Lemma 9. *$getRelevantCFoVList$ contains at most $O(m^2)$ vertices.*

$insideCFoV$(**sc**, $v(x, y)$)
Input: Service cell, Vertex of CFoV candidate.
Return: If the service cell is (partially or fully) contained in the CFoV rectangle return True.

1) $v'_x = v_x + CFoV_D$, $v'_y = v_y + CFoV_D$
2) For sc_{vertex} of sc:
   If $v_x \leq sc_{vertex_x} \leq v'_x$ and If $v_y \leq sc_{vertex_y} \leq v'_y$ then return True.
3) $v'_x = v_x + CFoV_D$, $v'_y = v_y - CFoV_D$
4) For sc_{vertex} of sc:
   If $v_x \leq sc_{vertex_x} \leq v'_x$ and If $v_y \leq sc_{vertex_y} \leq v'_y$ then return True.
5) $v'_x = v_x - CFoV_D$, $v'_y = v_y + CFoV_D$
6) For sc_{vertex} of sc:
   If $v_x \leq sc_{vertex_x} \leq v'_x$ and If $v_y \leq sc_{vertex_y} \leq v'_y$ then return True.
7) $v'_x = v_x - CFoV_D$, $v'_y = v_y - CFoV_D$
8) For sc_{vertex} of sc:
   If $v_x \leq sc_{vertex_x} \leq v'_x$ and If $v_y \leq sc_{vertex_y} \leq v'_y$ then return True.
9) return False.

Let:

- m be the number of user beams in each satellite.
- p be the satellite power.
- $CFoV_D$ be the satellite Field of View edge size.
- $SC_{(x,y,r)}$ be a service cell with a center at x, y and an edge size of $2 \cdot r$.

- SC_p be the power consumed by SC.

Algorithm 2 Service Cell Validation Algorithm

validatateServiceCell($SC_{(x,y,r)}$)

1) $referencezone$ = $getReferenceZone(SC_{(x,y,r)})$
2) $relevantCFoVVertices$ = $getCFoVLocations(referencezone)$
3) For each $CFoVRC$ in $relevantCFoVLocations$:
   $SCCounter$ = 0, $Pcounter$ = 0
   If $insideCFoV(CFoVRC,\ SC_{(x,y,r)})$ == False then continue.
   For each SC in $referencezone$:
   a) If $insideCFoV(CFoVRC,\ SC)$ == True
      Increase $SCCounter$
      Increase $PCounter$ by SC_p
      If $SCCounter > m$ return False.
      If $PCounter > p$ return False.
4) Return True

AddServiceCell($SC_{(x,y,r)}$)

1) $CFoV_D = \frac{360}{n}$.
2) x'=$\lfloor \frac{x}{CFoV_D} \rfloor$, y'=$\lfloor \frac{y}{CFoV_D} \rfloor$.
3) Append $SC_{(x,y,r)}$ to $SCA[x'][y']$.

Lemma 10. *The algorithm returns True if there are no rectangles that include the service cell with more than m service cells or consume more than p power.*

Proof. Assuming there is such a rectangle called FRC that contains the new service cell and more than m service cells or the total power in the rectangle exceeds p then one of the following must be true:

- Some of the FRC service cells were not considered by the algorithm. Contradicts Lemma 5.
- The algorithm considered all the service cells in FRC, but did not check FRC as a rectangle, i.e. the service cells in FRC were not verified as a group. Contradicts Lemma 8.
- FRC service cells were analyzed as a group, but the algorithm returned True, although the number of service cells is bigger than m or the total power of the service cells is bigger than p. Contradicts the algorithm (see Algorithm 2, specifically, Operation 3).

□

Lemma 11. *The algorithm returns False if there is a FRC rectangle that includes the service cell with more than m service cells or consumes more than p power.*

Proof. Assuming there is such a rectangle FRC that contains the new service cell and more than m service cells or the total power in the rectangle exceeds p and the algorithm returns True, then one of the following must be true:

- Some of the FRC service cells were not considered by the algorithm. Contradicts Lemma 5.
- The algorithm considered all the service cells in RFC, but did not check RFC as a rectangle, i.e. the service cells in FRC were not verified as a group. Contradicts Lemma 8.
- RFC service cells were analyzed as a group, but the algorithm returned False, although the number of service cells is smaller than m or the total power of the service cells is smaller than p. Contradicts the algorithm (see Algorithm 2, specifically, Operation 3).

□

Lemma 12. *The algorithm complexity is $O(m^3)$ where m is the maximum number of user beams in a satellite.*

Lemma 13. *The Algorithm memory requirement is $O(n^2+k)$ where n^2 is the number of satellites and k is the number of valid service cells.*

V. Simulation

This section details the simulations of the service cells validation algorithm. The simulations include multiple configurations of service cells randomly allocated (with different CFoV rectangle sizes ranging from 2% to 32% of CFoV rectangle size) and served by a grid of 8x8 satellites. The service area is a rectangle of size 5x5 CFoV. The number of user beams ranges from 1 to 16 beams per satellite, and we assume uniform satellites (i.e. all satellites use the same number of beams).

The satellite grid is moving vertically and horizontally such that service cells (and parts of them) are dynamically assigned to the proper satellites.

The simulation demonstrates the service cell validation algorithm complexity and correctness. The simulation is coded in Python and was executed on a dual core i5 Intel PC running Windows10 OS. The simulated scenarios include:

- The service cell validation algorithm is used to validate the service cells. The algorithm adds the service cells that were approved to a two dimensional geo-cells array.
- The service cell validation algorithm correctness is evaluated by checking beam pointing failures (a satellite has reached its maximum number of beams).

A. Maximum Number of Service Cells

In this section we present the impact of the number of user beams and service cell size on the number of valid service cells.

As expected, the number of valid service cells decreases as we increase the size of service cells and increases as we add more beams. The impact of the increase in the service cell size is logarithmic and the impact of increasing the number of user beams is linear (see Figure 7). Figure 6 demonstrates the combined impact of changes in number of beams and service cell size when trying to add 1000 service cells at random locations.

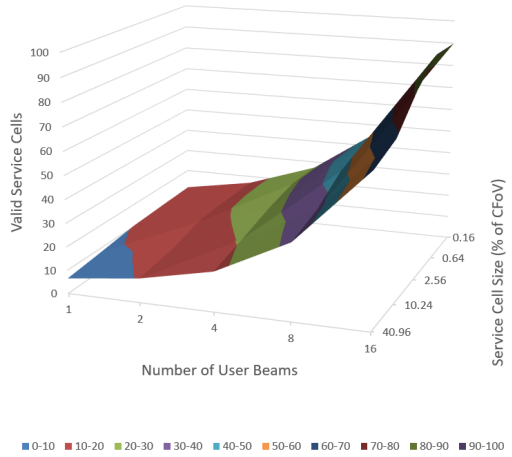

Fig. 6: Combined Impact Analysis of Number of User Beams and Service Cell Size on the Maximum Number of Service Cells.

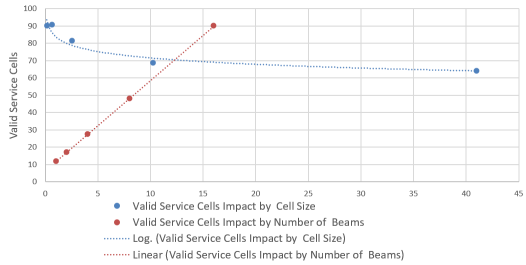

Fig. 7: Maximum Number of Service Cells.

B. Service Cell Validation Algorithm Complexity

In order to evaluate the complexity of the service cell validation algorithm, the following telemetries are collected in each validation cycle:

- Reference-zone size: How many service cells are in the reference zone.
- Number of CFoV rectangles: The number of CFoV rectangles in the reference zone, that are evaluated by the algorithm.
- Number of relevant CFoV rectangles: The number of CFoV rectangles in the reference zone, that include the new service cell and are checked for maximal number of service beams and power limit.

The results (Figure 8) support the expected theoretical complexity. In addition, the correctness of the algorithm was evaluated by running beam pointing algorithm to cover the service cells.

VI. Discussion

Stepping beams constellations provide a more efficient coverage, yet require complex algorithms for managing the beams, and validating new service areas. We presented a novel stepping beam validation algorithm for validating a new service area at a complexity of $O(m^3)$ (where m is the number of beams in a satellite). Future research can prove a tighter computational and memory complexity bounds.

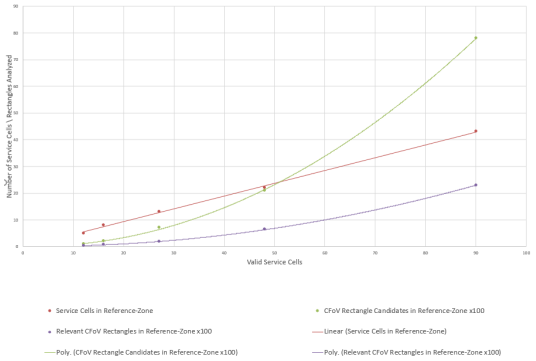

Fig. 8: Service Cell Validation Complexity Analysis.

References

[1] Shiyi X, Quanjiang J, Cheng Z, Guotong L, "Beam Coverage Comparison of LEO Satellite Systems Based on User Diversification," *IEEE ACCESS*, 2019.

[2] J. V. Evans, "Proposed U.S. Global satellite systems operating at Ka-Band," *IEEE Aerosp. Conf*, Vol. 4, Mar. 1998, pp. 525-537.

[3] G. B. Shaw, D. W. Miller, and D. E. Hastings, "The generalized information network analysis methodology for distributed satellite systems,", textifPh.D. dissertation, Dept. Aeronaut. , Massachusetts Inst. Technol., Cambridge, MA, USA, 1999.

[4] I. del Portillo, B. G. Cameron, and E. F. Crawley, "A technical comparison of three low Earth orbit satellite constellation systems to provide global broadband,", *Acta Astronautica*, vol. 159, pp. 123-135, Jun. 2019.

[5] "Propagation Data and Prediction Methods Required for the Design of Earth-Space Telecommunication Systems, document Recommendation", *ITU-R618íC13*, 2017.

[6] Inigo del Portillo, Bruce G. Cameron, Edward F. Crawley, "A technical comparison of three low earth orbit satellite constellation systems to provide global broadband,", *Acta Astronautica*, Volume 159,2019, Pages 123-135, ISSN 0094-5765.

[7] Guangming DaiXiaoyu, ChenXiaoyu ChenMaoca, WangMaocai Wang-Show, "Analysis of Satellite Constellations for the Continuous Coverage of Ground Regions", *Journal of Spacecraft and Rockets*,54(3):1-10, Aug 2017.

[8] Haotian Zhou, Liang Liu, Huadong Ma, "Coverage and Capacity Analysis of LEO Satellite Network Supporting Internet of Things,", *IEEE ICC*, May 2019, DOI: 10.1109/ICC.2019.8761682.

[9] Zoltán Katona, Michael Gräßlin, Anton Donner, Norman Kranich, Hartmut Brandt, Hermann Bischl, Martin Brück, "A flexible LEO satellite modem with Ka-band RF frontend for a data relay satellite system", *Journal of Satellite Comm. and Networking*, 2018, 00:1-18, DOI: 10.1002/sat.

[10] LeoSat Non-Geostationary Satellite System Attachment,, A: Technical Annex to Supplement Schedule S. Accessed: Mar. 20, 2018. [Online]. Available: http://licensing.fcc.gov/myibfs/download. do attachment key=115822"

[11] M. A. Sturza, "The teledesic satellitesystem: Overview and design trades,", *in Proc. Annu. Wireless Symp*, Feb. 1995, pp. 402-409.

[12] WorldVu Satellites Limited. OneWeb Ka-Band NGSO Constellation FCC Filing SAT-LOI- 20160428-00041. [Online]. Available: http://licensing.fcc.gov/myibfs/forwardtopublictabaction.o?,le number=SATLOI2016042800041

[13] Telesat Canada. Telesat Ka-Band NGSO Constellation FCC Filing SAT-PDR-20161115-00108.

[14] I.M. Derici, M. T. Panu, "Determining the maximum clique size in large random geometric graphs,",*Conf. on High Performance Computing and Simulation*, 2011, pp. 143-154, doi: 10.1109/HPCSim.2011.5999818.