Scalable Resource Allocation Methods

J. Biro

Department of Telecommunications and Media Informatics

Budapest University of Technology and Economics

2012 szeptember

1 Introduction

Several efforts, such as [1–3] have been studying changes to the internetworking architecture. Common to these efforts is the emphasis on disseminating information at possibly large-scale between a set of transient communication entities. In addition to these explorative efforts, recent developments in the Internet, such as increased usage of online video services like YouTube, have been demanding more efficient solutions for the distribution of content to a growing number of users. This makes efficient multicast a core requirement for any successful solution in this space.

Due to the growing number of information streams to be distributed (ranging from online video over news distribution to large-scale sensor data distribution), it is obvious that a stateless solution to large-scale multicast is desired to make any headway in the adoption of such new services. An appealing approach to minimize, even avoid, state in the network is that of source routing which is therefore in its second blossom [4–6] after decades of ignorance in the IP world. Here, instead of encoding next hop information of the multicast tree at the intermediate nodes, as done in traditional approaches, efficiently encoding link information of the graph in a compact header allows for avoiding any state at these intermediary elements.

Such design issues manifest in recent efforts towards using in-packet Bloom filters [4–7] for encoding the edges of the multicast tree into the packet header. Bloom filters are originally designed for membership queries i.e. for determining whether an element/edge belongs to a set/tree or not. Placed in packet headers, the *in-packet* Bloom filters can effectively address a set of nodes or links. In this paper we argue that this flat type of tree representation (i.e. the structure of the whole tree is represented in an implicit manner as a set of edges) can require needlessly large amount of bits, only allowing to encode multicast trees with limited number of receivers. As our main contribution we also show that the problem can be more efficiently tackled by using some topology-related information when composing the filter. We propose the so called *multi-stage filter* which consists of consecutive Bloom filters encoding only the membership of the edges residing at a given hop-distance (*stages*) from the source. Our analytical investigations and simulations prove that the solution not only provides excellent space efficiency but it also remedies several anomalies arising when Bloom filters are at use, like forwarding loops and packet storms.

From an overall network design perspective, an important consideration is that of the scheme being used for encoding the link information. Favored here is the usage of fixed size identifiers, enabling line-speed execution of the forwarding operation (see [4] for an example). Such fixed size, however, limits the ability to appropriately encode any given size tree within the limited size of the identifier. While variable size header approaches have been clearly at a disadvantage compared to fixed size approaches, it is our contribution in this paper to provide a solution that closes this gap in possible forwarding speed while providing the design advantages outlined above. For this, we contrast our work against existing source routing approaches and provide insight into the performance and implementation issues of our scheme.

Crucial to our approach is the knowledge of the overall topology over which the delivery graph is formed. Such

topology knowledge is not unreasonable to assume, as can be seen in technologies like MPLS as well as in proposals envisioning future information-centric solutions [3]. Common here is the existence of a topology manager with knowledge of link information between forwarding elements under the management of this entity. It is the role of this topology manager to compute an appropriate forwarding header that can be used within its topology. We argue that the existence of such topology management in today's networks points towards the possibility to improve information delivery in existing IP networks as well.

The rest of the paper is organized as follows. Section 2 expands on the related work in this area. As the main contribution of this paper, our new multicast architecture is proposed in Section 3. We discuss the performance measures and present our simulation results on the proposed false positive free stateless multicast approach in Section 4 and Section 5, respectively. Finally, Section 6 concludes the paper.

2 Related Work

2.1 Multicast Routing

Standard IP multicasting can provide a way of transmitting packets from a source to many recipients in a bandwidth economical way. Multicast routing protocols like DVMRP [8, 9] and PIM-SM [10] for performing the transmission maintain Multicast Forwarding Tables (MFT) in the routers along the multicast trees. This stateful approach is not only in full contradiction of the stateless design of unicast IP, but is also the Achilles' heel of IP multicast. Due to the potentially high number of multicast groups in the network according to massive multi-party applications like video conferencing or networked games, the states to be maintained in a router could be untenable due to the unaggregability of the MFT's.

For better scalability explicit multicast solutions have been suggested in the literature, in which forwarding information (e.g. list of destination IP addresses) on the targeted group is encoded in the packet headers. This flat type of stateless multicast routing protocols (like Xcast, Xcast+ [11]) undoubtly has the drawback that every intermediate node along the multicast tree should process the header, even in the case the node is not a branching point of the tree. To avoid mandatory packet header processing, tree encoding schemes have been proposed to integrate into multicast protocols, like in ERM and Linkcast [12], [13]. In these methods, the entire tree structure is encoded into the headers, moreover, in the latter case (and its successors [11]) forwarding at intermediate nodes is based on interpreting the tree code, unicast lookups are completely eliminated. Nevertheless, the tree encoding scheme in Linkcast uses fixed length link codes by the assumption on the maximum size of multicast groups and imposes large computational overhead when interpreting the encoded tree.

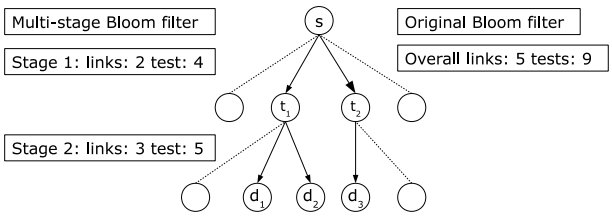


Figure 1: Testing the traditional and multi-stage in-packet Bloom filters at the forwarding phase

2.2 In-packet Bloom Filters

The Bloom filter [14] is a simple yet efficient data structure to answer membership queries. Let $\mathbb{S}$ be a set of elements, assigned with m-bit long binary codes, in which a maximum of k bits are set in positions indicated by k different *hash functions*. Each of the k hash functions maps the given element onto one of the m bit positions. The hash functions are assumed to be independent and each position is selected with equal probability. The Bloom filter representing $\mathbb{S}$ is an m bit long binary array consisting of the bitwise OR of the codes of the elements in $\mathbb{S}$. As the main feature a membership test can simply be performed by checking if all bit positions that are set in the code of the underlying element are also set in the filter. The performance of the filter is measured by the false positive rate that is the probability that an element appears to be included (by test) but actually it is never added.

The application of Bloom filters is becoming increasingly popular in future Internet architectures [4, 5]. Placed in packet headers, the *in-packet* Bloom filters can effectively address a set of nodes or links, hereby qualifying themselves as a strong candidate solution for efficient stateless multicast addressing. When a packet with in-packet Bloom filter arrives to a router, membership testing is performed on the outgoing link identifiers. The bitwise AND operation on the Bloom filter placed in the header and on the address of the outgoing link is extremely fast when implemented on a digital signal processor, which leads to a simpler router architecture compared to current IP routers. Moreover, Bloom filters are favored for their space efficiency since the filter requires much less space than listing the identifiers for each links/hosts in the multicast tree. Furthermore, this allows routers to stay quasi-stateless, because the routers only need to know the address of the neighboring nodes and links.

A clear weakness of the in-packet bloom filter concept is the total ignorance of (i) the topology information and (ii) the tree structure in addressing. In [6] the first issue was touched by generating the bloom filter addresses of each link according to the number of adjacent links, which is intuitively beneficial because the links with fewer adjacent links has a smaller chance to give false positive.

2.3 Our Contribution

In this paper, we propose a solution for arbitrary size tree and multicast groups which lies somewhat between the two extreme approaches of explicit routing (i.e. the completely flat solution[1]) and the complete tree encoding based ones. In our solution link identifiers at equal hop-distances from the source of the multicast tree (referred to as *stages* hereafter) are to be confined into Bloom filters, but without any exact encoding of the tree topology hereby providing an appealing mixture of space and forwarding efficiency.

Furthermore, we address both weaknesses (i) and (ii) of in-packet Bloom filters by introducing multi-stage filters. As opposed to the traditional in-packet Bloom filter concept [4] which represent the trees flatly as sets, we build our filter to enclose limited information about the structure of the tree (its stage decomposition). We will show that using this information we obtain more succinct tree representation while still maintaining forwarding efficiency and getting rid of typical Bloom filter illnesses.

3 Multicast architecture with multi-stage in-packet Bloom filters

For simplicity reasons, similarly to previous works on Bloom filters, we assume in the following a central entity called a *membership manager*. However, it is important to note that this function can be easily implemented in a distributed fashion, e.g. target nodes subscribe to source nodes directly. Similarly to other source routing approaches we assume that the up-to-date topology of the network is available at the *topology manager*. It is also the task of this topology manager to compute the multicast trees (or DAGs) for each multicast group after consulting with the membership manager. Following the architecture of [4] the links are assigned by binary addresses and the topology manager computes the multicast packet headers by combining the addresses of the links residing in the corresponding multicast tree. For a better understanding of the specifics of the proposed multi-stage filter let us first recall how this computation is done using the traditional in-packet Bloom filters.

- Each link is assigned by a binary link address (consisting of m bits of which at most k are set)
- The topology manager computes the multicast header by bitwise OR-ing the addresses of the links in the corresponding multicast tree.

At each router outgoing link addresses are tested against the Bloom filter (i.e. testing at each bit-position where the link address is set if the filter is also set) and forwarded if positive. Note that the this approach is totally topology unaware since it simply traces back the forwarding decision to a classical membership testing problem. As an example in the

[1] We also consider that case as completely flat approach in which all the link identifiers are collected and used somehow, without any knowledge on the tree topology.

multicast tree of Fig. 1 the in-packet Bloom filter contains the bitwise OR of 5 link addresses. The filter is tested 9 times (against every outgoing interface) during the forwarding process.

To reduce filter size, our idea is to use a sequence of smaller so called "stage" filters instead of a large one, each containing the edges with the same hop-distance from the source node in the multicast tree. When forwarding in the relay nodes these filters are removed at each stage one-by-one. That is, a tree of h hops is represented by h bloom filters, where the i-th one contains only the links residing at i hop-distance from the source. When leaving the source the multicast packet header consists of h filters, which then shrinks as the packet travels along the tree. As it is shown in the example of Fig. 1, at stage 1 the first filter is tested against four links, of which two is expected to be chosen, while at stage 2 the second filter is tested five times with three expected positive outcomes.

For ensuring space efficiency the length of the filters at each stage are optimized (discussed in Section 4) to the number of elements (i.e. links) it contains, which clearly results in varying sized stage filters. For identifying filter boundaries we propose to store the length of each filter in the header e.g. by applying the commonly used Elias gamma universal code [15], where an m bit long filter is coded as follows. First $\lceil \log_2(m) \rceil - 1$ zero bits are written, followed by the binary representation of m on $\lceil \log_2(m) \rceil$ bits.

3.1 Implementation Issues in the Routers

The forwarding process of this multi-stage filter run in each relay node along the multicast tree consists the following steps.

(1) The header of the incoming multicast packet is loaded into a fast access type of memory.

(2) The starting number of 0's is counted, and the *length* field is loaded accordingly.

(3) The following *length* number of bits is identified as the current stage filter and prepared for bloom filter membership testing.

(4) The bloom filter membership testing is executed in parallel at each link interface card.

(5) If the test is positive then at the current bit position the header is truncated, that is the current stage filter with the length field is removed from the header.

(6) If the header size is greater than zero the packet is forwarded using the new header.

The membership testing requires a bitwise logical AND operation of the bloom filter and the link address. The result is then tested if equals to the link address itself. Recall, that the filters at different stages can have different sizes, which requires link addresses of variable sizes. To store the addresses of all possible lengths for each link can be overwhelmingly memory consuming, prompt generation can provide a better solution instead. For this purpose [16] proposes a lightweight

mechanism, where two uniformly distributed random hash functions (even two can be enough) is used to generate variable size hash codes to implement Bloom filters without any loss in the asymptotic false positive probability. In this case two hash functions $h_1(x)$ and $h_2(x)$ are stored at line cards, where x is the link address. The i-th hash function $g_i(x)$ is generated by the formula $g_i(x) = h_1(x) + i \cdot h_2(x) \mod m$, where m is the length of the hash code of the link to be established, and $i = 1, \ldots, k$. Finally, it is tested whether the $g_i(x)$-th bit in the stage filter is 1 for every i. Note that in such implementation membership testing depends mainly on the number of hash functions and less on the size of bloom filters, allowing fast forwarding even for large bloom filters.

Finally, we sketch how can it be implement on the current routers which were designed to handle fixed size headers. Recall that, the header is abbreviated at each forwarding by erasing the processed bloom filter; thus the actual stage is always the first Bloom filter. Therefore by performing Step (1)-(4) it is enough to load the first stage, which is typically much smaller than the current IPv4 headers, and was always smaller than the current IPv6 headers in our simulation experiments, not to mention the comparison with the traditional Bloom filter approach, where the complete tree encoding should be loaded. In addition truncating the header at forwarding is also a simple task. This presumes that the proposed multi-stage Bloom filter approach with minor limitations can be implemented on the current router hardware.

Forwarding an IPv4 packet takes $6 - 8$ memory access (and up to 16 for IPv6); thus, in the current IP routers the memory access is still the dominant time at forwarding. Note that a 3GHz processor has CPU cycle of 0.3 nanoseconds, while a DDR SDRAM memory cycle is ~ 20 nanoseconds which leads to 60 CPU cycles for each memory access, and $240 - 320$ CPU cycles for the IPv4 address lookup.

In the proposed method $h_1(x)$ and $h_2(x)$ should be stored for each port, which can be stored in the local Level 1 (L1) cache of the CPU. L1 cache can be accessed in just a few CPU cycles and its typical size is tens of kilobytes. Recall that in the proposed method instead of address lookup we need to decode the length of the first stage, perform $2 - 3$ modulo division with the two hash functions of $h_1(x)$ and $h_2(x)$, and test the related bit-positions in the header. These operations should not require more than ~ 100 CPU cycles according to our rough estimation.

4 Performance Measures

4.1 Filter Compactness

The standard performance metric for the evaluation of Bloom filters is the false positive probability, which is 0 in today's IP based routing. Remember that this metric gives the probability that a packet is forwarded on a link which is not contained in the multicast tree (it was never added to the filter). Thus, for the sake of appropriate positioning of our results among currently used technologies, in the following analysis we will use filter size related metrics, i.e. in every scenario we determine the filter length which ensures false positive free operation. Note that such filters can be easily generated by

the topology manager for any given multicast tree $\mathbb{L}$.

Remember that the filter length m_l at link l in the proposed architecture stands for:

$$m_l = \gamma_l + b_l, \tag{1}$$

where γ_l is the total length of the Elias gamma codes of the stage filters, while b_l is the total bits consumed by (stage) Bloom filters in the header at link l.

Definition 1 *The* filter compactness *of a multicast tree $\mathbb{L}$ is denoted by $\eta(\mathbb{L})$, which is the sum of the header overhead along each link in the multicast tree divided by the square of the number of links $|\mathbb{L}|$ in the multicast tree, formally*

$$\eta(\mathbb{L}) = \frac{\sum_{l \in \mathbb{L}} m_l}{|\mathbb{L}|^2} = \frac{m_{avg}}{|\mathbb{L}|}.$$

Obviously, a lower $\eta(\mathbb{L})$ refers to a more compact filter in bit per link (*bpl*). To further explain, filter compactness refers to the average filter length divided by the number of tree links stored in the filter. Using this definition the performance of an architecture does not depend on the tree size, and the average filter lenght can be obtained by $\eta(\mathbb{L}) \cdot |L|$.

To compare the overall performance of different filters, in the simulations a set of multicast tree is generated for a given network topology $G = (V, E)$, denoted by $\mathcal{D}$, and the *average filter compactness*

$$\eta_{avg} = \frac{\sum_{\mathbb{L} \in D} \eta(\mathbb{L})}{|D|}$$

of these trees is evaluated in the investigated architecture.

Recall that in our multicast architecture the headers are truncted at forwarding (i.e. the route information already traversed by the packet is erased), which leads to a better filter compactness η_{avg}. Therefore as a reference average filter compactness is also evaluated without abbreviating the headers at forwarding, denoted by μ_{avg}.

4.2 False-Positive-Free Bloom Filter Length

Now we show that breaking down the Bloom filter into stages according to the levels of the multicast tree (called *multi-stage (MS)* in-packet Bloom filter) can significantly reduce the overall space requirement b_l of the traditional Bloom filter (called *Single Stage (1S)* in-packet Bloom filter). In order to obtain a fair comparison of the two approaches, similarly to the filter compactness we define *Bloom filter compactness*, formally

$$\lambda(\mathbb{L}) = \frac{\sum_{l \in \mathbb{L}} b_l}{|\mathbb{L}|^2} = \frac{b_{avg}}{|\mathbb{L}|},$$

and *average Bloom filter compactness* without abbreviating the headers as λ_{avg} for a set of $\mathcal{D}$ multicast trees.

Suppose that we want to encode a set of links $\mathbb{L}$ with cardinality n of a given multicast tree. First we recall how to design an optimally sized Bloom filter to meet a prescribed false positive requirement [17]. Without loss of generality, the

Table 1: Comparison of traditional and multi-stage Bloom filter sizes b_l when $\alpha = 2$, $\beta = 3$

No. of hops	b_{avg}^{1S}	b_{avg}^{MS}	No. of hops	b_{avg}^{1S}	b_{avg}^{MS}
1	5.15	5.15	4	30.70	20.60
2	12.42	10.3	5	41.07	25.75
3	21.08	15.45			

false positive probability for a Bloom filter can be calculated as:

$$P_f = \left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^k . \tag{2}$$

Introduce the notation $p = \left(1 - \frac{1}{m}\right)^{kn}$ and minimizing P_f using the first derivative transforms to

$$\frac{\partial P_f}{\partial k} = (1-p)^{k-1} \cdot [(1-p) \cdot \ln(1-p) - p \ln p] = 0. \tag{3}$$

It is easy to see that the optimal k, that is the optimal number of hash functions is attained when $p = 0.5$:

$$k^{OPT} = \frac{1}{n} \frac{\ln(0.5)}{\ln(1 - \frac{1}{m})} \text{ , and } P_f^{OPT} = 0.5^{\frac{\ln(0.5)}{n \ln\left(1 - \frac{1}{m}\right)}}, \tag{4}$$

or using

$$\left(1 - \frac{k}{m}\right)^n \approx e^{-\frac{kn}{m}} \tag{5}$$

we get the approximation

$$P_f^{OPT} \approx 0.5^{\frac{-m \ln 0.5}{n}} \approx 0.618503^{\frac{m}{n}}. \tag{6}$$

Let α and β denote the number of in-tree (which are contained in the multicast tree, denoted by arrows in Fig. 1) and out-tree (which are not contained in the multicast tree, dashed lines in Fig. 1) links in a given stage, respectively, and we assume that there are h stages. For the sake of simplicity we conduct our analysis for identical α and β values in each stage. However, in Section 5 general settings are used.

In the following the length of false-positive-free Bloom filters is discussed, i.e. a large enough filter is created for which the expected number of false positives will be zero. For this let us define the following probability:

$$p(m, \alpha, \beta) := \left(1 - 0.618503^{\frac{m}{\alpha}}\right)^{\beta} . \tag{7}$$

This expresses the probability, that in an m-length filter containing α elements none of the β out-tree links are included in the filter. Now the length of the false-positive-free Bloom filter comes from the process of trying with increasing size filters and stop when false positive cannot be found. That is this expected length in case of the traditional filter is

$$b_{avg}^{1S} = \sum_{m=1}^{\infty} m \cdot p(m, h\alpha, h\beta) \prod_{i=1}^{m-1} (1 - p(i, h\alpha, h\beta)), \tag{8}$$

Table 2: Topologies used for the numerical evaluations with number of nodes $|V|$ and links $|E|$, and bpl value for the Multi-Stage (MS) and traditional in-packet Bloom filters (1S).

	Xcast	η_{avg}		μ_{avg}	λ_{avg}	
Topology ($\|V\|, \|E\|$)	IPv4	MS	1S	MS	MS	1S
Cost266 [18] (37,57)	14.4	4.63	6.85	7.25	4.26	5.83
Germany [18] (50,88)	15.6	4.99	7.47	7.55	4.60	6.40
Deltacom [19] (113,161)	10.3	4.17	7.10	6.89	3.97	6.34
random E-R (1000,1997)	9.94	7.41	11.9	11.01	8.39	11.1
AS level (34305,71448)	10.0	12.5	24.0	18.81	15.8	23.0

because there are $h\alpha$ links to be included and $h\beta$ links are to be excluded in the filter. The expected length of the false-positive-free multi-stage filter is

$$b_{avg}^{MS} = h \sum_{m=1}^{\infty} m \cdot p(m, \alpha, \beta) \prod_{i=1}^{m-1} (1 - p(i, \alpha, \beta)), \tag{9}$$

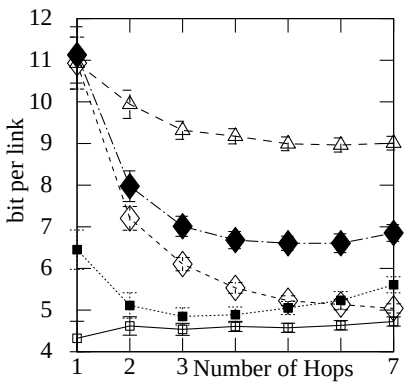
because there are h identical false-positive-free filters (each containing α links and not containing β links) confined in the stage filter.

Using the above formulae Table 1 shows a comparison between the length of the traditional and multi-stage Bloom filters, when $\alpha = 2$, $\beta = 3$ and the number of hops increases from 1 to 5. Notice that the improvement can be stunning even if the number of stages is relatively small, for trees with larger depth 30-40% improvement can be reached.

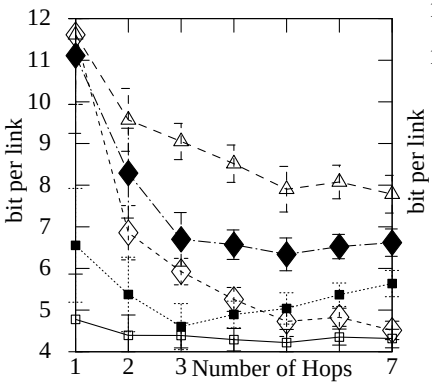
5 Simulation Results

In the simulation we compare the two Bloom filter based forwarding approaches, namely the traditional 1S in-packet Bloom filters which was modified to handle variable size headers (drawn with filled marks on the charts), and the proposed MS in-packet Bloom filters (drawn with empty marks on the charts). We compare their performance in terms of the metrics proposed in Section 4, i.e. η_{avg} and μ_{avg} to investigate the effect of abbreviating the header, and μ_{avg} and λ_{avg} to investigate the effect of variable size headers.

Fig. 2 shows the result on the COST 266 pan-European backbone network with 37 nodes and 57 links. The demands were classified according to the maximum hop distance in the multicast tree. 2000 random demands were generated with unicast traffic only on Fig. 2(a), and multicast traffic with up to 10 terminal nodes on Fig. 2(b). Coinciding with our analytical evaluation in Section 4 the multi-stage Bloom filter has significantly shorter filter sizes. Surprisingly this is true even when the multi-stage Bloom filter consists of one Bloom filter in one hop trees. This is because in multi-stage Bloom filters the number of links on which false positive can occurr is smaller, since none of the links, which are two hops away from the source node can generate a false positive, due to the header truncation mechanism, i.e. zero sized header prevents the forwarding of a packet in the multi stage case. Note that the size of the multicast trees increases as the number of hops



(a) Unicast demands



(b) Multicast demands

Figure 2: The header length versus the number of hops in the multicast tree for the 37-node European reference network.

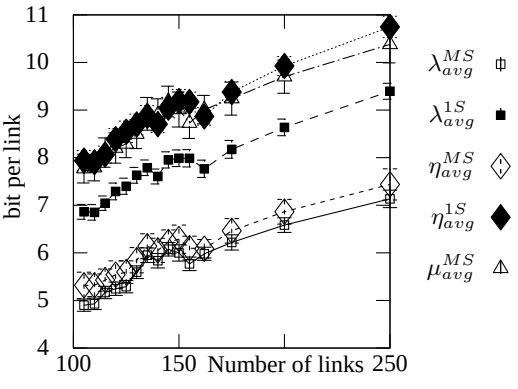


Figure 3: Results of 2-connected 50-node networks with different number of links.

increases and shows the great scalability multi-stage Bloom filter can achieve.

In the case of traditional in-packet Bloom filters encoding the size adds small overhead, especially for large multicast trees. Since multi-stage Bloom filters consists of several consecutive Bloom filters, one may argue that encoding these boundaries may end up in a larger overhead. Our results support that this is not the case, since due to the abbreviation of the header at forwarding and the space efficiency of the multi-stage filter, in average header length the multi-stage Bloom filters remarkably outperform the original approach. The advantages of multi-stage in-packet Bloom filters is even more significant for unicast demands.

Next we investigate the performance respect the network density. 20 random 50-node, two-connected networks are generated with different network density. Fig. 3 shows the results of 500 demands where the horizontal axis corresponds to the number of links in the graph. Charts shows that the performance gain of multi-stage Bloom filter is always $3-4$ *bpl* and does not depend on the density of the topology.

Finally, we investigate the scalability of the approaches as the network growth. Table 2 shows the results on topologies with different sizes and with multicast demands of terminal nodes at most 10. As a reference Xcast based solution is also added to the table, in which the header consists of a series of IPv4 addresses with 32 bit for each destination. However, note that IPv4 and Xcast are not a source routing solution and require large routing tables at the routers. We were surprised to see that, although the forwarding decision for an in-packet Bloom filter is significantly simpler and faster compared to traditional IPv4 forwarding, using in-packet Bloom filters even at the AS-level has similar performance than Xcast.

5.1 Forwarding loops and packet storms

A well-known problem of Bloom filters is that through a chain of false positives a packet can loop back to a previously visited node where generates a false positive again and stuck in an infinite loop. In extreme situations such a behavior may cause even packets storms. In [6] a bit permutation technique was proposed to prevents such anomalies with very high probability. The proposed multi-stage filters remedy these illnesses in a fairly natural way, since due to the shrinking of the filter after every stage, the packets cannot go farther than a few hops. In such a way multi-stage Bloom filters certainly prevent the formation of infinite loops by encoding the stage decomposition into the header.

6 Discussions

In this paper the concept of in-packet Bloom filters is revisited for achieving stateless multicast addressing. Although the current Internet supports mainly unicast traffic with IPv4 forwarding and addressing mechanisms, there is an ever increasing demand for a scalable multicast addressing and forwarding architecture future information centric networks can rely on. We believe that in such networks true stateless multicast addressing is the desired solution. The proposed novel approach, called in-packet multi-stage Bloom filters, with variable sized headers, can efficiently provide true stateless multicast. Moreover, since all of the prerequisites (e.g. topology manager, membership manager) of our scheme already exists in today's networks, this points towards the possibility to improve information delivery in existing IP networks as well. For this, we envision that the establishment of (multi-point) communication relationships is handled through existing IP approaches, such as search functions, discovery through repositories or explicit signaling frameworks, while the data delivery is off-loaded to an efficient source routing forwarding mechanism. This leads to a hybrid system in which slower operations are IP-based while utilizing an efficient multicast scheme for the actual delivery.

7 Fair Allocation of Scarce Network Resources

In this paper we address the problem of allocating scarce resources in a network so that every user gets a fair share, for some reasonable definition of fairness. For example, a fair allocation would be such that every user gets the same share, and the allocation is maximal in the sense that there does not exist any larger, even and feasible allocation. In the context of networking, the fairness problem manifests itself most prominently in the task to compute a fair rate at which users can send data in a telecommunications network, whose links are of limited capacity. In this paper we concentrate on this very setting, while also realizing that there are many other important areas of networking to which our results do not immediately apply.

Perhaps the most practical way to understand the context of this paper is through an example. Consider the simple directed network of Fig. 4, and suppose that there are 3 source-destination pairs (or commodities or users): (1,5), (2,5) and

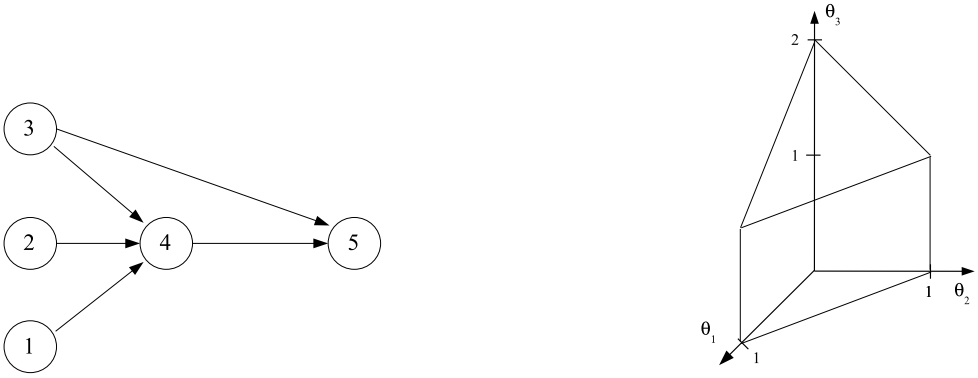


Figure 4: A sample network and the set of throughputs realizable in it. All edge capacities equal to 1.

(3,5). All the edge capacities are uniformly 1. Now, the task is to compute a transmission rate (or throughput, for short) for each of the commodities, that is on the one hand feasible (so it can be routed in the network without violating the edge capacities) and on the other hand satisfies some fairness criteria. For example, according to the above naive interpretation of fairness, we would need to allocate $\frac{1}{2}$ amount of throughput for each user. This allocation unarguably gives even share to each user, it is certainly feasible and it is also maximal in the sense that there does not exist any larger feasible and even allocation.

Amongst the many different definitions of fairness perhaps the most prevailing one is max-min fairness. A max-min fair allocation is, roughly speaking, such that we cannot increase the throughput of any one of the users without decreasing some other user's throughput, which is already smaller [20]. Max-min fairness is a simple, yet very powerful fairness criterion, and consequently it has grown to be a central and essential ingredient in diverse fields of telecommunications, like flow control protocols [21], bandwidth sharing in ATM networks [22], etc. For further analysis of the related extensive literature, the reader is referred to [23], [24]. For some related economical aspects, see [25].

Max-min fairness is most easily described in a network model, where a single path is assigned to each of the users, and this path remains fixed during the lifetime of the communication. Here, the task is to compute a rate at which users can send data to their path, so that the allocation is max-min fair and neither of the edges gets overloaded. A very useful tool to solve this problem is the notion of bottlenecks [23]. A bottleneck edge, with respect to a certain commodity, is an edge with the properties that *(i)* it is filled to capacity and *(ii)* the commodity has the maximum throughput amongst the commodities whose path traverses the edge. Bottlenecks are very tightly coupled with max-min fairness, for it can be shown that an allocation of throughputs is max-min fair over some fixed single-path routing, if and only if all the commodities have a bottleneck edge.

From the practical standpoint, the importance of this *bottleneck argumentation* is multi-faceted. First, as the name suggests, bottlenecks point to certain shortages of resources in the network that, given the selected set of paths, constrain the fair allocation. Additionally, bottlenecks substantiate a fast algorithm to find a max-min fair allocation: we increase

the throughputs of the commodities at the same pace, until an edge is saturated. Then we fix the throughputs of the commodities whose path passes through the saturated edge and keep on increasing others. The procedure is repeated until eventually a bottleneck is found for each commodity. This algorithm is known as the water-filling algorithm and it provably yields the max-min fair allocation [23].

Assume that, in the sample network of Fig. 4, path $1 \to 4 \to 5$ is assigned to commodity (1,5), path $2 \to 4 \to 5$ to commodity (2,5), and the direct path $3 \to 5$ to commodity (3,5), respectively. Then, the first edge that becomes saturated as the the water-filling algorithm proceeds is edge (4,5), which becomes the bottleneck edge for commodities (1,5) and (2,5). So the throughput of both of these commodities is fixed at $\frac{1}{2}$, and only the throughput of commodity (3,5) is increased any longer. This, in turn, gets saturated at a throughput of 1 unit. The final max-min fair allocation is represented by the vector $[\frac{1}{2}, \frac{1}{2}, 1]$, using the order of commodities set out above.

Highlighting its usefulness, several extensions and ramifications of max-min fairness have come to existence throughout the years (min-max fairness [26], weighted max-min fairness [23], max-min utility fairness [27], max-min fairness with guaranteed minimum rate [28] and various combinations of these). Since all of these concepts can be traced back to the unweighted case [26] and a respective bottleneck argumentation, analogous to the one above, can always be made, we shall not address these concepts any more. Furthermore, the limitation that each commodity uses one single, fixed path can also be weakened somewhat without invalidating the bottleneck argumentation: it is possible to assign more paths to each of the commodities, provided that the splitting ratios at branching nodes, and the paths themselves, remain fixed.

It is important to observe that the actual selection of paths influences to a great extent the emergent max-min fair allocation. For example, if the path of commodity (3,5) is changed to $3 \to 4 \to 5$, then the max-min fair throughput vector turns to $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$. If we allow both paths for the third commodity, but we require that traffic is split evenly amongst them, then the max-min fair allocation becomes $[\frac{2}{5}, \frac{2}{5}, \frac{2}{5}]$. If the splitting ratio is changed, the max-min fair allocation also changes. Apparently, different routings give rise to different max-min fair allocations, which is somewhat unnatural since, after all, it is the network – or, more precisely, the capacity of the edges – that constrains feasible allocations. Accordingly, we should first compute a max-min fair allocation that is only dependent on the parameters of the network itself, and only after this we should pick a routing that realizes it.

So far, very little has been done to understand the relationship between the specifics of a network, the distribution of the source-destination pairs and the emergent max-min fair allocation (see [29] for a treatment in the context of single-source unsplittable routing and [30] for an algorithm based on successive linear programming). For this intriguing problem to solve, we need to somehow describe the entire set of throughput values that can be realized in the network, and examine this set whether or not it gives rise to a max-min fair allocation. Prior to this paper it has not even been clear, if the max-min fairness problem in this setting makes sense, albeit its importance has been very clearly pointed out [26, Section "When bottleneck and water-filling become less obvious"]. Below, we shall refer to this problem as *the general max-min fair allocation problem*, and all the former incarnations will be called *fixed-path max-min fairness problems*.

The first part of the paper will be devoted to solve the general max-min fair allocation problem. Our main tool in doing so is the so called throughput polytope: a concise polyhedral description of the feasible throughput vectors. After a quick roundup on the notation in Section 8, we shall investigate this polytope and its applications to capacitated network fairness in Section 9. We study non-dominated throughput allocations, a basic type of fairness concepts of which more complex concepts, like Pareto-optimality and max-min fairness, can be constructed. Regarding the latter, we extend the scope of the bottleneck argumentation from the fixed-path network model to the generic case. Closing the first part of paper, we propose an algorithm to compute the throughput polytope. The viability of the algorithm will be demonstrated in real network scenarios.

In the second part of the paper, Section 10, we shall argue that the polyhedron of the feasible throughputs, our main working horse in studying max-min fairness, is a remarkably interesting beast even in its very own right. We shall point out some intriguing potential applications ranging from inter-domain traffic engineering to admission control. Most importantly, a deficiency in the concept of max-min fairness will be addressed. Namely, realizing the max-min fair allocation in a network might drive it to a highly suboptimal region of operation, where it carries much less traffic than it could bear. Therefore, we formulate the concept of optimally max-min fair allocations, which assures that the network realizes its maximum achievable net-throughput, yet the users get as fair service as possible. Finally, in Section 11 we conclude our work.

Reading this paper requires a solid understanding of polyhedral mathematics, for which we give a basic introduction in the Appendix. To further help the comprehension of the results, the theory will always be accompanied by illustrative examples and the proofs will be deferred to the Appendix.

8 Preliminaries

In this section, we present the most important notations and conventions we shall use throughout this paper. A vector will be denoted by a lowercase letter. Most of the time, the ith coordinate of a vector v will be referred to as v_i, but in some cases, to stress that we are dealing with a specific coordinate, we shall use the notation $(v)_i$. What now follows is a list of the notation we shall use in the sequel:

- $G(V, E)$: a directed graph, with the set of nodes V ($|V| = n$) and the set of directed edges E ($|E| = m$).
- u: the column m-vector of (finite) edge capacities.
- $(s_k, d_k) : k \in \mathcal{K} = \{1, \ldots, K\}$: the set of source-destination pairs (commodities).

Note that the graph $G(V, E)$, the edge capacities u and the set of commodities $(s_k, d_k) : k \in \mathcal{K}$ together describe a *network*, which will be referred to as G for brevity.

- e_i: the canonical unit vector (of proper size implied by the context) with 1 in the position corresponding to the ith coordinate and all zero otherwise.
- $\mathcal{P}_k$: the set of *all* directed paths from s_k to d_k in $G(V, E)$ for some commodity $k \in \mathcal{K}$.
- f_P^k: the amount of flow routed to some path $P \in \mathcal{P}_k$ for some commodity $k \in \mathcal{K}$. Note that the collection $[f_P^k] : k \in \mathcal{K}, P \in \mathcal{P}_k$ gives rise to a *routing* in G.
- θ_k: the throughput of some commodity $k \in \mathcal{K}$, that is, the aggregate flow that flows from s_k to d_k:

$$\theta_k = \sum_{P \in \mathcal{P}_k} f_P^k \,. \tag{10}$$

  The vector of throughputs is a column K-vector θ.
- $T(G)$: the set of throughputs realizable in the network G. Here, realizability means that there exists a routing in G (represented by the variables $[f_P^k] : k \in \mathcal{K}, P \in \mathcal{P}_k$), so that (10) holds for each commodity and edge capacities are respected:

$$\sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{P}_k : (i,j) \in P} f_P^k \leq u_{ij} \qquad \forall (i,j) \in E$$

- w: a weighing of the edges of $G(V, E)$. In fact, w is a row m-vector, so that the coordinate w_{ij} corresponding to some edge $(i, j) \in E$ denotes the weight of that edge. We usually require that $w \geq 0$.
- β: a row K-vector. We shall mostly use β in a very special sense: the coordinate $(\beta)_k$ corresponding to commodity k will denote the length of shortest s_k to d_k path over some weighing w. This will be made more explicit later on.
- $\mathcal{S} \subseteq E$: a separating edge set, that is, a set of edges whose removal from the network would destroy all the directed s_k to d_k paths for at least one commodity $k \in \mathcal{K}$.
- $\mathcal{K}_\mathcal{S}$: the set of commodities disconnected by some separating edge set $\mathcal{S}$.

In order to simplify the foregoing discussions, hereafter we shall only deal with networks that satisfy certain (rather mild) regularity conditions:

Definition 2 *A network G is regular, if*

- *a directed path exists in G from s_k to d_k for each commodity $k \in \mathcal{K}$ and*
- *all the edge capacities are finite and strictly positive.*

It is easy to see that any network can be reduced to a collection of regular networks (by eliminating edges with zero capacity, and fixing the throughput of the un-connected commodities to zero), so we shall mostly stick to regular networks henceforward.

9 The Throughput Polytope and Fairness

As promised, we intend to illustrate the ideas of this paper with examples. First, we show how to describe the set of throughput values that can be realized in the network of Fig. 4. Let θ_1 denote the throughput of commodity (1,5), θ_2 of commodity (2,5) and θ_3 of commodity (3,5), respectively. Since we can not push more flow than 1 via the edge (4,5), which is traversed by all the potential paths of commodity (1,5) and (2,5), we have that $\theta_1 + \theta_2 \leq 1$. Additionally, after routing 1 unit of flow of commodity (3,5) along the edge (3,5), every ϵ units of additional flow of this commodity has to traverse edge (4,5), decreasing the aggregate throughput remaining for commodity (1,5) and (2,5) by exactly ϵ units. So, $\theta_1 + \theta_2 + \theta_3 \leq 2$. It can be shown that these inequalities, together with the restriction that the throughputs are non-negative, give rise to a complete and irredundant description of the set of throughputs realizable in the network:

$$T(G) = \{[\theta_1, \theta_2, \theta_3] : \quad \theta_1 + \theta_2 \leq 1 \tag{11}$$

$$\theta_1 + \theta_2 + \theta_3 \leq 2 \tag{12}$$

$$\theta_1, \theta_2, \theta_3 \geq 0 \quad \} \tag{13}$$

Observe how all the constraints turned out to be linear. Sets of similar kind are called polyhedra, which might be familiar as these are exactly the geometric objects that underlie linear programming. It is tempting to take a closer look at the boundaries of this polyhedron (called faces in the terminology relevant to polyhedra), since these are exactly the points that represent allocations, which are maximal in some sense. In our example, $T(G)$ is characterized by 2 constraints. We have already seen that constraint (11) comes from the property of edge (4,5) that all paths of commodity (1,5) and (2,5) traverse it. In other words, edge (4,5) separates away the source nodes from the respective destination nodes. Hence, its capacity provides an upper bound on the achievable throughput for these commodities. Interestingly, a set of separating edges can be associated with constraint (12) too: observe that the edges (1,5) and (4,5) separate away *all* the commodities, and the aggregate amount of flow that can be pushed through this separating edge set is 2 units. We shall point out later on the strong relation between the constraints that describe $T(G)$ and certain separating edge sets, which then will be used to analyze max-min fair allocations.

Taking again our example, it turns out that the max-min fair allocation over the set $T(G)$ is represented by the vector $[\frac{1}{2}, \frac{1}{2}, 1]$. Observe that neither throughput values can be increased without decreasing an already smaller throughput. Now, to commodity (1,5) and (2,5) assign the separating edge set made up by the edge (4,5), and to commodity (3,5) the edge set $\{(1,5),(4,5)\}$, respectively (these separating edge sets just happen to be identical to the ones we associated with constraints (11) and (12), but this is only coincidental). It is intriguing to observe, how nicely all the properties of bottleneck edges (given for the fixed-path model previously) are reflected and generalized by the above selection of (bottleneck) separating edge sets. First, in our example, all the edges of both of the separating edge sets are saturated when we realize the max-min fair allocation. Second, for any commodity the throughput is maximal amongst the commodities

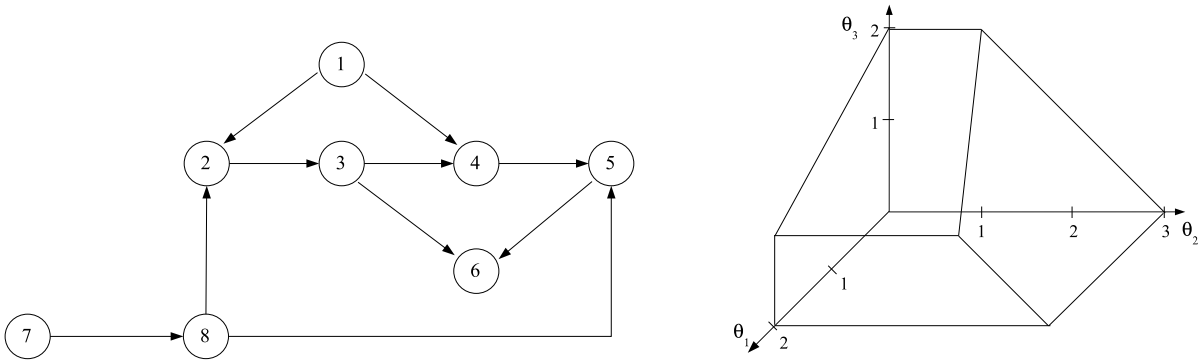


Figure 5: Another sample network and the associated set of feasible throughputs. All edge capacities equal to 1 except for edge (7,8), whose capacity is 3.

whose source node is separated away from the respective destination node by the corresponding separating edge set. Later on we shall confirm this analogy more formally. We shall also extend it to the most important property of bottlenecks: we shall show that a throughput allocation is max-min fair in the general sense, if and only if all commodities have a bottleneck – at least for some liberal interpretation of the notion of bottlenecks.

Unfortunately, the network of Fig. 4 is too simple to demonstrate any more than the most basic ideas. Thus, in Fig. 5 we give another, more complex network for illustrative purposes. The commodities are as follows: let $(s_1, d_1) = (1, 6)$, $(s_2, d_2) = (7, 8)$ and $(s_3, d_3) = (7, 5)$. The corresponding set of throughputs, realizable in the network of Fig. 5, is as follows:

$$T(G) = \{[\theta_1, \theta_2, \theta_3] : \quad \theta_1 \leq 2 \tag{14}$$

$$\theta_2 + \theta_3 \leq 3 \tag{15}$$

$$\theta_1 + 2\theta_3 \leq 4 \tag{16}$$

$$\theta_1, \theta_2, \theta_3 \geq 0 \quad \} \tag{17}$$

In order to explain how the constraints of $T(G)$ arise and to find the max-min fair allocation, we need to invoke polyhedral mathematics (see the Appendix, [31] and [32]).

9.1 The Throughput Polytope

Below, we concentrate on characterizing and describing the set of feasible throughputs in terms of the topological properties of the network, but independently of a selection of fixed paths whatsoever. In doing so, our most important aid is the so called Japanese Theorem, which dates back to 1971 when it was found independently by Iri [33] and Onaga and Kakusho [34].

Proposition 1 (Japanese Theorem) *Some $0 \leq \theta \in T(G)$, if and only if for each non-negative weighing w of the edges of G it holds that $\beta\theta \leq wu$, where β is the vector of shortest path lengths, whose kth coordinate $(\beta)_k$ denotes the length of the shortest s_k to d_k path over w.*

As shall be seen, the Japanese Theorem gives a very useful characterization for $T(G)$. It is only an additional benefit that this characterization comes in the familiar form of edge weights. We kindly encourage the reader to experiment with different weight settings in the toy-networks of Fig. 4 and Fig. 5 to make herself comfortable with the claims of the proposition.

The most important consequence of the Japanese Theorem is that $T(G)$ is constrained by linear inequalities $\beta\theta \leq wu$ generated by different weighings of the edges, and *all* the relevant inequalities can be derived this way. Obviously, if one takes every possible non-negative weighing w, computes the corresponding vector of shortest path lengths β and composes the inequality $\beta\theta \leq wu$, then eventually a complete description of $T(G)$ is obtained. Additionally, Iri showed that it is enough to take integer-valued weight sets, upper-bounded by $w_{ij} \leq K^m$. Hence, $T(G)$ is basically an intersection of finitely many halfspaces, which leads us to the following important result:

Theorem 1 *$T(G)$ is a polyhedron. For a regular network, $T(G)$ is bounded, so it is a polytope.*

The latter claim is simply a consequence of the boundedness of the capacity vector u. Since regularity infers some very useful properties of $T(G)$ (for example, $T(G)$ is of full dimension for a regular G), we shall limit ourselves to regular networks in the sequel. Accordingly, $T(G)$ will be referred to as the *throughput polytope* henceforth.

So far, we have seen that $T(G)$ arises as the intersection of finitely many inequalities, generated by different non-negative, integer valued and upper-bounded weight settings. So, generate all these inequalities, eliminate redundancy, and multiple each remaining inequality with a positive number, such that the coordinates of β are integers and relative primes. What we obtained in this way is a unique, complete and non-redundant representation, which we shall call the *standard form* of the throughput polytope:

$$T(G) = \{\theta \geq 0 : \beta_i\theta \leq b_i,\ i \in \{1, \ldots, N\}\}\ ,$$

and a constructive proof to the following properties:

Property 1 *Every constraint $\beta_i\theta \leq b_i : i \in \{1, \ldots, N\}$ is generated by some weighing $w \geq 0$ of the edges of G, so that β_i is exactly the vector of the shortest path lengths with respect to w and $b_i = wu$.*

Property 2 *$\forall i \in \{1, \ldots, N\}$: $\beta_i \geq 0$ and $b > 0$.*

Below, we show how certain weighings of the edges reproduce the throughput polytope corresponding to the network of Fig. 5. The first constraint $\theta_1 \leq 2$ comes from the Maximum Flow–Minimum Cut Theorem of Ford and Fulkerson [35]

applied to the first commodity. The actual minimum cut is formed by, for instance, edges $(1,2)$ and $(1,4)$. The weight set that generates the constraint might be chosen as follows: associate a weight of 1 unit for both of the edges of the minimum cut, and zero to all the others. Observe that the source node of commodity $(1,6)$ is separated away from the respective destination node by the edges of the minimum cut, so the length of the corresponding shortest path is 1 unit. Thus, $\beta_1 = 1$. Along the same lines one gets $\beta_2 = \beta_3 = 0$ and $wu = 2$, so both the left-hand-side and the right-hand-side are correctly reproduced by this selection of edge weights. In the case of $\theta_2 + \theta_3 \leq 3$, for example the selection $w_{(7,8)} = 1$ and all zero otherwise correctly reproduces the constraint. But choosing the right weights for the third constraint, $\theta_1 + 2\theta_3 \leq 4$, is a bit more involving. Set $w_{(2,3)} = w_{(4,5)} = 1$ and $w_{(8,5)} = 2$. We leave it as an exercise for the reader to confirm that all the claims of the Japanese Theorem apply.

9.2 Non-dominated and Pareto-optimal Allocations

The significance of the polyhedral description of $T(G)$ is manifold. First, it gives a concise characterization of the amount of traffic that can be pushed through a network, which only depends on the parameters of the network itself. When its dimension is low (that is, for networks with only a few commodities), the throughput polytope can be computed by hand and it can be visualized easily. However, even for networks with many source-destination pairs, $T(G)$ is a relatively low dimensional polytope, so once we have it in the standard form, it becomes painless to optimize over it. But perhaps most notably, the polyhedral description provides a tool to make some important observations regarding the fair throughput allocations arising in a network. In this section, we investigate the most basic types of fairness concepts, namely non-dominated and Pareto-optimal allocations.

If we say that some commodity k is non-dominated at some $\theta_0 \in T(G)$, we basically mean that either the throughput of k can not be increased at all from $(\theta_0)_k$, or otherwise increasing it is only possible at the expense of decreasing the throughput of some other commodity, say, l [25]. Note that l is also non-dominated at θ_0 in this case.

Definition 3 *Let a feasible allocation of throughputs be θ_0. We say that some commodity $k \in K$ is dominated at θ_0, if $\exists \epsilon > 0$ so that $\theta_0 + \epsilon e_k \in T(G)$. Otherwise, k is non-dominated at θ_0.*

An allocation of throughputs θ_0 is non-dominated, if at least one commodity is non-dominated at θ_0. Otherwise, θ_0 is dominated.

In the network of Fig. 5, the point $\theta_0 = [2,0,1]$ for instance is non-dominated, since commodity $(1,6)$ is at its maxflow. In another interpretation, as we try to increase the throughput of $(1,6)$ from θ_0, a constraint in $T(G)$ becomes active (in this case this constraint is $\theta_1 \leq 2$), which inhibits any further increase. Besides commodity $(1,6)$, $(8,5)$ is also non-dominated at θ_0, since it can only be increased at the cost of decreasing the throughput of $(1,6)$. Here, a constraint causing non-dominatedness is for example $\theta_1 + 2\theta_3 \leq 4$. Finally, we see that commodity $(7,8)$ is dominated at θ_0, since we can by no means find a valid inequality that would block it at θ_0.

This reasoning helps to identify where exactly dominated and non-dominated allocations are located in the throughput polytope. Since at least one constraint is binding at any non-dominated point, these allocations are exactly those residing at any one of the *facets* of $T(G)$ (not counting here the $\theta_k \geq 0$ constraints as facets). In contrast, dominated throughput vectors lie in the *interior* of $T(G)$.

The following important result, which we shall often put to use in the sequel, relates a non-dominated allocation to a very special valid inequality (and a dominated one to the lack thereof).

Theorem 2 *Let $\theta_0 \in T(G)$ be non-dominated. Now, some set of commodities $\mathcal{N} \subseteq \mathcal{K}$ is non-dominated, and $\mathcal{K} \setminus \mathcal{N}$ is dominated at θ_0, if and only if there exists an inequality $\beta\theta \leq b$ so that:*

- *$\beta\theta \leq b$ is valid for $T(G)$ and $\beta \geq 0$*
- *$\beta\theta_0 = b$*
- *$\beta_k > 0$ if and only if $k \in \mathcal{N}$*

The proof of the theorem is based on the observation that if one sums up all the constraints in the standard form of $T(G)$, which are active at θ_0, then a valid inequality is obtained that satisfies all the requirements of the theorem. In the case of the non-dominated point $\theta_0 = [2, 0, 1]$, this valid inequality would be the sum of the two binding constraints (14) and (15): $\theta_1 + \theta_3 \leq 3$. For a complete proof, the reader is referred to the Appendix.

In their own right, non-dominated allocations do not make really much sense. Why we still treat them is because non-dominatedness is exactly the essential building block of which we can construct more complex concepts of fairness. The first such concept we discuss here is Pareto-optimality.

A Pareto-optimal allocation is such that "there is no way to make any person better off without hurting anybody else" [25]. That is, for a Pareto-optimal allocation of throughputs it holds that any commodity is either at its maximum flow, so the throughput can not be increased at all, or otherwise increasing it is only possible at the expense of decreasing the throughput of some other commodity. It is then easy to dissect Pareto-optimality to non-dominatedness.

Definition 4 *An allocation of throughputs θ_0 is (strictly) Pareto-optimal, if it is feasible and all the commodities are non-dominated at θ_0.*

To identify the faces of $T(G)$ that contain the Pareto-optimal vectors, we reason as follows. Being non-dominated, a Pareto-optimal vector of throughputs must reside at some facet of $T(G)$. In fact, we expect it to reside at certain intersections of the facets, since Pareto-optimality is a stronger property than non-dominatedness. And indeed, applying directly Theorem 2 yields the following characterization:

Corollary 1 *An allocation of throughputs $\theta_0 \in T(G)$ is Pareto-optimal, if and only if there exists an inequality $\beta\theta \leq b$, so that:*

- $\beta\theta \leq b$ *is valid for* $T(G)$
- $\beta\theta_0 = b$
- $\forall k \in \mathcal{K} : (\beta)_k > 0$

Corollary 1 suggests a simple algorithm to search for a Pareto-optimal allocation in $T(G)$: taking some random ordering of the commodities, increase the throughput of the commodities one by one as long as it is possible. Eventually, all the commodities will be blocked, so a valid inequality $\beta\theta \leq b$ with all strictly positive $(\beta)_k$ coordinates must be binding at the resultant point. In the case of the network of Fig. 5, the two line segments between the points $[2, 3, 0]$ and $[2, 2, 1]$, respectively $[2, 2, 1]$ and $[0, 1, 2]$, contain all the Pareto-optimal points.

9.3 Max-min Fair Allocations

From the practical standpoint, Pareto-optimality is a somewhat weak, though clearly desirable fairness criterion. Desirable, because it avoids the wastage of resources non-dominatedness generally allows for. Weak, because Pareto-optimality permits allocations where one user gets everything, which is not really fair (observe for instance that the allocation $[0, 0, 2]$ is Pareto-optimal in network of Fig. 4). The concept of max-min fairness is based on the idea to pick the "fairest" Pareto-optimal allocation. Here, fairness means that any throughput increase must be at the cost of a decrease of some already smaller throughput. Formally:

Definition 5 *An allocation of throughputs* θ_0 *is max-min fair, if it is feasible and* $\forall\theta \in T(G) : (\theta)_k > (\theta_0)_k \Rightarrow \exists l \in \mathcal{K} \setminus \{k\}$*, so that* $(\theta)_l < (\theta_0)_l$ *and* $(\theta_0)_l \leq (\theta_0)_k$*.*

It is by far not evident whether or not this definition makes sense in the case of $T(G)$, or in fact how many max-min fair vectors it yields. Nevertheless, it was shown that compact and convex sets of vectors always give rise to a unique max-min fair allocation [26]. But, being a polytope, $T(G)$ is compact and convex, which substantiates the following result:

Theorem 3 *Let* G *be a regular network. Then there exists a max-min fair allocation over* $T(G)$*, and it is unique.*

In the case of non-dominated and Pareto-optimal allocations, it turned out to be really beneficial to characterize the conforming vectors in terms of valid inequalities. This characterization helped to localize non-dominated and Pareto-optimal allocations in $T(G)$, and also suggested a simple algorithm to compute one. Below we again take this route, however, our characterization of max-min fair allocations involves not just one, but exactly K valid inequalities (one for each commodity).

Corollary 2 *An allocation of throughputs* θ_0 *is max-min fair in* G*, if and only if for each* $k \in \mathcal{K}$ *there exists an inequality* $\beta\theta \leq b$*, so that:*

- $\beta\theta \leq b$ *is valid for* $T(G)$ *and* $\beta \geq 0$
- $\beta\theta_0 = b$
- $\forall l \in \mathcal{K} : (\beta)_l > 0$ *if and only if* $(\theta_0)_l \leq (\theta_0)_k$

Again, consult the Appendix for the proof.

The valid inequality $\beta\theta \leq b$ ordered to a commodity k will be referred to as the *bottleneck* of k. Using this convention we could rephrase Corollary 2 as follows: *an allocation of throughputs is max-min fair in the generic sense, if and only if all commodities have a bottleneck*. This formulation is exactly the same as the one given for fixed-path max-min fairness problem, only the definition of bottlenecks differs somewhat. Interestingly, the analogy goes even further, since not just bottlenecks, but the water-filling algorithm too extends to the general max-min fair allocation problem. Recall that the water-filling algorithm is based on the idea to create a bottleneck for at least one commodity in every iteration, no matter in which form bottlenecks are defined. Thus, the second important consequence of Corollary 2 is that *the water-filling algorithm is correct to search for a max-min fair allocation over* $T(G)$.

Consider the network of Fig. 5, and execute the water-filling algorithm. As the first step, increase the throughput of all the commodities at the same pace. This equals to, starting from the origin, moving along the direction $[1, 1, 1]$ as long as some of the commodities gets blocked. This occurs at the point $[\frac{4}{3}, \frac{4}{3}, \frac{4}{3}]$, where the constraint $\theta_1 + 2\theta_3 \leq 4$ becomes active. As a matter of fact, this constraint will be the bottleneck for commodities $(1, 6)$ and $(7, 5)$. The only commodity that remains dominated at this point is $(7, 8)$, whose throughput can be increased to $\frac{5}{3}$. The resultant allocation, $\theta_0 = [\frac{4}{3}, \frac{5}{3}, \frac{4}{3}]$ is max-min fair. To obtain the bottleneck for the last commodity, $(7, 8)$, sum up the two constraints binding at θ_0, which yields $\theta_1 + \theta_2 + 3\theta_3 \leq 7$.

The final question that remained to be answered in this section is that, once we computed the max-min fair allocation θ_0, how to actually obtain a routing that realizes it. Consider the following linear program:

$$\min \sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k} f_P^k \tag{18}$$

$$\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k:(i,j)\in P} f_P^k \leq u_{ij} \quad \forall (i,j) \in E \tag{19}$$

$$\sum_{P\in\mathcal{P}_k} f_P^k = (\theta_0)_k \qquad \forall k \in \mathcal{K} \tag{20}$$

$$f_P^k \geq 0 \tag{21}$$

It is easy to see that any optimal feasible solution delivers a suitable routing via the $[f_P^k]$ variables. Constraint (19) assures that the capacity of an edge is not exceeded by the aggregate flow routed to it, while (20) requires that the routing indeed realizes θ_0. Additionally, (21) ensures that path flows are non-negative. The objective function (18) assures that the resultant routing uses as few resources as possible, since all the emergent paths are required to be shortest paths. See [35] and [36] on how to solve multicommodity flow problems of similar kind in polynomial time.

9.4 A Bottleneck Argumentation

So far, we have shown how the concept of bottlenecks extend from the fixed-path max-min fairness problem to the generic case. Analogously to the traditional fixed-path model, we could obtain an "if and only if" relation between the existence of bottlenecks for each commodity and max-min fairness, which also guaranteed the correctness of the water-filling algorithm. Quite regrettably, however, our bottlenecks are currently defined in terms of valid inequalities, which – being more of a polyhedral concept than a network theoretical one – is not really descriptive. In this section, we translate this bottleneck argumentation to the more palpable concept of separating edge sets, whose properties show remarkable similarity to the properties of the conventional "bottleneck edges" of the fixed-path model.

In the heart of the fixed-path model there lies the notion of bottleneck edges. A bottleneck edge is one that blocks any increase in the throughput of the commodity it belongs to. This is because *(i)* it is filled up to capacity when we realize the max-min fair allocation, and *(ii)* the corresponding commodity has the maximum throughput amongst the commodities that might want to use that edge. This conventional interpretation fails in the generic model, since neither the set of paths, nor the users of a particular edge are fixed.

A bottleneck edge blocks one particular, fixed path of some commodity. A separating edge set, by definition, blocks *all* the paths, so it might not be a completely lost idea to search for the counterparts of bottleneck edges in the form of *bottleneck separating edge sets*. What remained to be done is to somehow translate the defining properties of bottleneck edges to separating edge sets and see whether or not they make sense.

Let θ_0 be max-min fair in a regular network G, let $\mathcal{S}_k$ be the bottleneck separating edge set of some commodity $k \in \mathcal{K}$ and let $\mathcal{K}_{\mathcal{S}_k} \subseteq \mathcal{K}$ denote the set of commodities, whose source node is separated away from the respective destination node by $\mathcal{S}_k$. First, we reformulate the property of bottleneck edges that a commodity's throughput is maximal on its bottleneck edge amongst the commodities that use that edge. But "users" of separating edge sets are exactly the commodities that are separated away by it, so the this generalization seems reasonable:

Property 3 $l \in \mathcal{K}_{\mathcal{S}_k} \Leftrightarrow (\theta_0)_l \leq (\theta_0)_k$.

The second defining property of bottleneck edges is that they are always filled to capacity and the traffic of the blocked commodities can not circumvent this bottleneck. We translate this property to separating edge sets as follows:

Property 4 *For any routing* $[f_P^l] : l \in \mathcal{K}, P \in \mathcal{P}_l$ *that realizes* θ_0*, it holds that*

$$\forall\, (i,j) \in \mathcal{S}_k : \sum_{l \in \mathcal{K}_{\mathcal{S}_k}} \sum_{P \in \mathcal{P}_l : (i,j) \in P} f_P^l = u_{ij}$$

In words, Property 3 and Property 4 insist that a bottleneck separating edge set is always saturated by the flow of the commodities separated away by it, no matter how we instantiate the max-min fair allocation in the network. Therefore, any increase in the throughput of some commodity would cause the decrease of some other commodity's throughput that is

already smaller, and this property is independent of the actual routing. Interestingly, these properties, which appear quite natural generalizations of the conventional properties of bottlenecks, give rise to a bottleneck argumentation completely analogous to the conventional one:

Theorem 4 *An allocation of throughputs θ_0 is max-min fair, if and only if for each commodity there exists a bottleneck separating edge set featuring both Property 3 and Property 4.*

See the Appendix for a full-fledged proof of the theorem. Roughly speaking, the idea is to – extending Property 1 to bottlenecks – find a weight set that generates the bottleneck and use that weight set to define the bottleneck separating edge set. So, provided that G is regular, if $\beta\theta \leq b$ is the bottleneck of some commodity $k \in \mathcal{K}$, then there exists a weight set $w \geq 0$ so that $b = wu$ and β is exactly the vector of the shortest path lengths with respect to w. The bottleneck separating edge set $\mathcal{S}_k$ ordered to commodity k is defined as:

$$\mathcal{S}_k = \{(i,j) \in E : w_{ij} > 0\} .$$

Using this definition, $\mathcal{S}_k$ has the following property:

$$\beta_l > 0 \Leftrightarrow l \in \mathcal{K}_{\mathcal{S}_k} .$$

In the Appendix we prove that this definition indeed makes sense and bottleneck separating edge sets defined this way satisfy all the requirements of Theorem 4. We kindly encourage the reader to find these bottleneck separating edge sets in the network of Fig. 5 and verify their properties.

As a final remark, we note that our bottleneck argumentation contains the conventional one as a special case. To see this, it is enough to restrict all the commodities to use one single path and observe that bottleneck separating edge sets degrade to the conventional bottleneck edges in this case.

9.5 An Algorithm to Compute the Throughput Polytope

In the previous sections some very important concepts of fairness, like Pareto-optimality and max-min fairness, were analyzed. Notably, we could always find a quick algorithm to calculate a fair allocation, provided that an explicit description of the throughput polytope is at our disposal. Unfortunately, more complex fairness criteria (like for instance proportional fairness, utility fairness, etc., see [24] for the respective definitions) are not so easy to handle, and one needs to invoke nonlinear optimization techniques to obtain the corresponding fair allocation [37]. Even in such cases an explicit description of the throughput polytope is quite useful since, given its low-dimensionality and "nice" properties, nonlinear optimization over the throughput polytope is much simpler than optimization over the high-dimensional space of all feasible path-flows and throughputs. Therefore, it is crucial to find a viable algorithm which is, given some network G, able to compute the standard form of the corresponding throughput polytope $T(G)$.

Unfortunately, no algorithm to solve this problem is known for the authors at the moment. Moreover, we are not aware of any reasonable upper bound on the number of constraints that define $T(G)$ in the generic case. Therefore, the problem appears difficult, since there is no real hope for a polynomially sized description of the output. The following algorithm implements the most plausible way to attack this problem: generate *all* the valid inequalities one by one until we finally obtain $T(G)$.

1. Initialize the algorithm with an empty set $\mathcal{X}$ that will hold the valid inequalities we find. Let $j = 1$ and $\mathcal{D} = \{0, 1\}^K$.

2. Generate a valid inequality for each $\beta \in \mathcal{D}$, whose elements are relative primes. For this, solve the following linear program:

$$\begin{aligned} &\min wu \\ &\textstyle\sum_{(i,j)\in P} w_{ij} \geq (\beta)_k \quad \forall k \in \mathcal{K}, \forall P \in \mathcal{P}_k \\ &w \geq 0 \end{aligned}$$

   This linear program yields the weight set w, and so the left-hand-side wu, corresponding to the current β vector. Add the resultant inequality $\beta\theta \leq wu$ to $\mathcal{X}$.

3. After eliminating redundancy form $\mathcal{X}$, our current estimation of the throughput polytope is

$$T_j(G) = \{\theta \geq 0 : \beta_i\theta \leq b_i \quad \{\beta_i, b_i\} \in \mathcal{X}\} .$$

   Now, we check, whether all the vertices of $T_j(G)$ are feasible in G, so $T_j(G) \subseteq T(G)$. This immediately implies that $T_j(G) = T(G)$, since $T_j(G) \subset T(G)$ is impossible because all the inequalities we generate are actually valid. Checking for $T_j(G) \subseteq T(G)$ can be done by using the double-description method [38] to find all the vertices of $T_j(G)$ and solve the system (18)–(21) for each vertex. If every vertex turns out to be feasible, then terminate the algorithm since $T_j(G) = T(G)$.

4. Otherwise, the current set of β vectors was not sufficiently large to obtain all the constraints, so we have to extend $\mathcal{D}$. Let $j \leftarrow j + 1$ and $\mathcal{D} \leftarrow \{0, \ldots, j\}^K \setminus \mathcal{D}$ and proceed with Step 2.

Recall that there is a trivial upper bound on the link weights: $w_{ij} \leq K^m$. This imposes the upper bound mK^m on the relevant β vectors, and therefore on the number of iterations the algorithm has to perform. This confirms that the above algorithm terminates in finite, albeit usually exponentially many, steps. Quite amazingly, however, according to the simulation results presented below the algorithm generally terminates in much less iterations than its theoretical properties would suggest.

In the first round of the simulations, we systematically constructed networks for which the corresponding throughput polytope has many facets. Thus, we generated a sequence of increasing sized, directed, complete graphs. For a complete

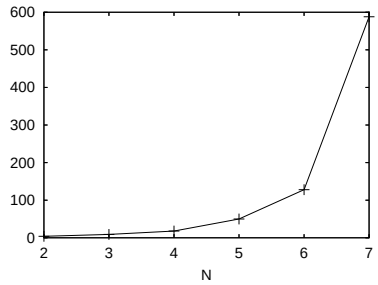


Figure 6: Number of facets of throughput polytope for complete graphs.

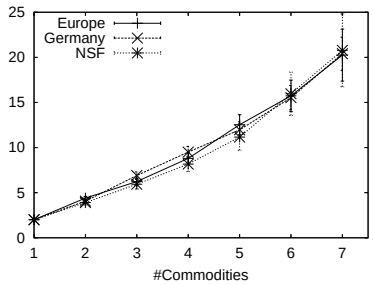


Figure 7: Average number of facets of the throughput polytope for realistic topologies (with 95% significance).

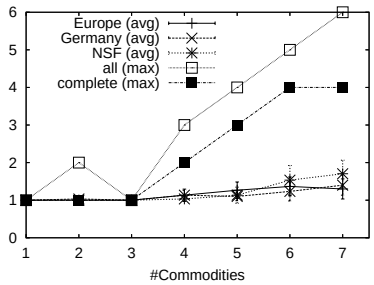


Figure 8: The average and the maximal number of iterations (i.e., the value of the maximal β_k coefficient in $T(G)$).

graph G_N of size N, we set exactly N commodities: $(1,2),(2,3),\ldots,(N,1)$. The number of facets of the resultant throughput polytope $T(G_N)$ as the function of N is given in Fig. 6. The consequence seems to be that one can easily construct graphs, for which the size of the throughput polytope quickly grows intractable (for $N = 8$ we could not even run the algorithm until completion). However, in real network scenarios the situation seems a bit more promising. To see this, we conducted a second round of simulations. We took some real-life network topology, placed an increasing number of commodities in it selecting the source and destination nodes randomly, fed the resultant network to the algorithm to compute the corresponding throughput polytope, repeated this process 30 times, and averaged the results. The network topologies were the 28 node European and the German reference networks used extensively in recent EU projects [39] and the ubiquitous US NSF network topology [40]. Fig. 7 shows the average number of facets of the throughput polytope as the function of the number of commodities. This graph suggests that in real networks a reasonable sized description of the throughput polytope is quite expectable.

Throughout both rounds of the simulations, we found that the algorithm performed only a few iterations (see Fig. 8). Interestingly, it almost always terminated in no more than 2 iterations for real networks and it never performed more than K iterations. As a rule of thumb, therefore, K appears to be a pessimistic, yet reasonable upper bound to estimate the number of iterations, and hence the memory and CPU requirements of the algorithm. This upper bound, albeit theoretically not correct, turned out to be considerably accurate in real-life networks, at least as long as the number of commodities was not prohibitively large.

10 Further Applications

The questions "can we increase the throughput of some of our users without hurting the traffic already in place?" and "how modifying up or down the traffic of some user affects the service we can deliver to some other user?" arise quite naturally in the context of networking. For example, similar questions have to be answered when a service provider

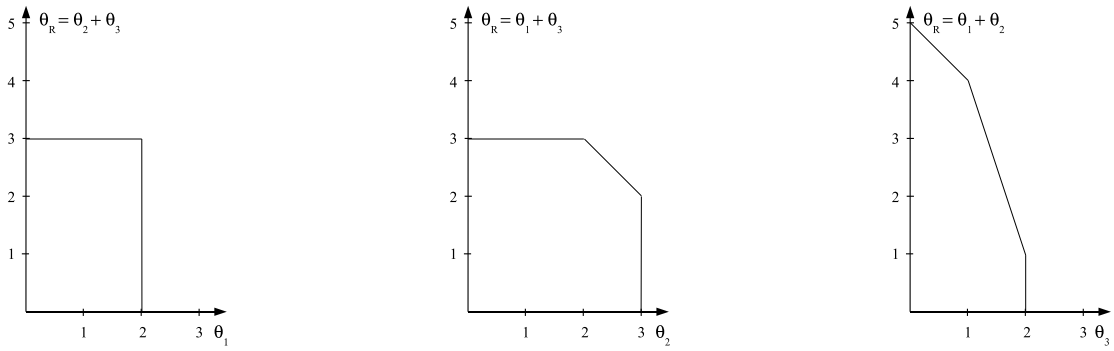
needs to decide, whether or not a particular combination of service level agreements can be safely met simultaneously, or whether to admit additional transit traffic into the network as a consequence of creating peering relationships. In this section, we study how the clever use of the throughput polytope answers these fundamental questions as well as several other compelling problems in diverse areas of telecommunications.

Network operators have long been struggling with the challenge of aggregating the state of their network in a compact way for the purposes of advertising it to the outside world. Here, the intent is to provide such an aggregate state information, which still contains enough data to make global traffic engineering decisions, yet not unduly detailed to reveal sensitive and confidential internal information to the competitors. The challenge of *inter-domain traffic engineering* rises most profoundly in the context of internets constituted by multiple Autonomous Systems (ASs), which communicate through long-haul data pipes provisioned between a small number of dedicated border routers. A possible solution to the state aggregation problem in this setting comes from ATM PNNI: roughly speaking the idea is to represent the topology of the AS (or *peer group* in the relevant terminology) by a *nucleus* and add a virtual link emanating from it to all the border routers. Though, the exact mapping of the internal structure of the AS to the virtual links is by far not self-evident and as such, subject to substantial inconsistency and inaccuracy. This makes it really hard for anyone outside of the AS to decide, whether or not enough transit capacity at a reasonable price is available in the AS between two border routers, or the AS needs to be circumvented upon establishing a connection spanning multiple domains.

A more precise method for state aggregation would be instead to generate the throughput polytope that describes the topology of the AS with the border routers as the source-destination pairs. This throughput polytope purports just enough information to facilitate inter-domain traffic engineering decisions, while it adequately hides all sensitive topological information. Provided that the number of border routers is not particularly large and consequently a compact description of the throughput polytope with not prohibitively many facets exists, it suits perfectly to be announced and interchanged between the ASs of an internet. The reconstruction of the network-global throughput polytope from those of the ASs is a startling perspective, however, it is well beyond the scope of this paper.

Apart from aggregating the characteristics of a network, throughput polytopes have yet another striking feature. Namely, a throughput polytope tells us many interesting little details about the subtle interplays and interrelations between the commodities in the network. After all, the traffic of different commodities has to share the same physical transport medium, which might lead to adverse interference phenomena amongst them. In other cases, the commodities might turn out to be completely independent from each other, and all this follows from the structural properties, and hence from the throughput polytope, of the network. In this paper we concentrate on one particular question that is of crucial relevance to the network operator: “if I admit an additional 1 unit of traffic of some commodity in the network, how does then the throughput realizable by the rest of the commodities change?”. Obviously, if admitting additional traffic costs too much[2], that is, if it supplants too much potential traffic of the rest of the commodities, then it is not worth admitting that

[2]For the sake of simplicity, in this paper we treat all commodities to be of equal utility to the operator. Though, our model clearly admits an arbitrary



$\theta_1 \leq 2$ $\quad$ $\theta_2 \leq 3$ $\quad$ $\theta_3 \leq 2$

$\theta_R \leq 3$ $\quad$ $\theta_R \leq 3$ $\quad$ $\theta_3 + \theta_R \leq 5$

$\theta_1, \theta_R \geq 0$ $\quad$ $\theta_2 + \theta_R \leq 5$ $\quad$ $3\theta_3 + \theta_R \leq 7$

$\theta_2, \theta_R \geq 0$ $\quad$ $\theta_3, \theta_R \geq 0$

Figure 9: Reductions of the throughput polytope with respect to the first, second and third commodity, respectively.

traffic into the network. To alleviate the difficulties of this *admission control* decision, the throughput polytope, or more precisely certain affine mappings of it, turn out to be exceptionally useful.

Take for example the sample network of Fig. 5. We ask how increasing the throughput of commodity $(7, 8)$ affects the aggregate throughput that remains realizable to the rest of the commodities. Accordingly, we need to represent $\theta_1 + \theta_3$ as the function of θ_2. The second subfigure in Fig. 9 depicts precisely this diagram. To obtain it, we add a new variable $\theta_R = \theta_1 + \theta_3$ to $T(G)$, extending its dimension to 4, and then project out the superfluous variables (namely, θ_1 and θ_3), reducing the dimension to 2. We call such 2-dimensional diagrams as *reductions* of the throughput polytope:

$$R_k(G) = \{[\theta_k, \theta_R] : \exists \theta \in T(G) \text{ with } \theta_R = \sum_{l \in \mathcal{K} \setminus \{k\}} \theta_l)\}$$

Hence, $R_k(G)$ is in effect an *affine mapping* of $T(G)$, which immediately gives a viable method to obtain it [31]. Additionally, this assures that a $R_k(G)$ is again a polytope, provided that G is regular.

Observe how simply the concept of reductions answers the fundamental question of admission control. For example, in Fig. 9 we see that one can safely increase θ_2 from zero to 2 units without affecting the aggregate throughput realizable by the rest of the commodities in any regards. Here we reach the maximum achievable net-throughput of the network (5 units of traffic). Increasing θ_2 from 2 to 3 units does not make too much harm either: the total throughput we lose in this process is also one unit, so we made a fair deal. At $\theta_2 = 3$ we reach the maxflow of commodity $(7, 8)$, which blocks any further increase. From this we conclude that no special care needs to be taken when we admit the traffic of

weighing of the commodities to be considered.

commodity $(7, 8)$ in the network. Notably, as can be seen on the third subfigure of Fig. 9, the situation is just the contrary with commodity $(7, 5)$: as θ_3 is increased from 1 unit, we basically trade-off every ϵ unit of increase in θ_3 for exactly 3ϵ units of aggregate traffic of commodity $(1, 6)$ and $(7, 8)$. So the net loss is 2ϵ units, which makes it really unappealing to increase θ_3 beyond 1 unit.

A close examination of reductions reveals interesting phase-transitions of networks: from an initial state, where increasing the throughput of some commodity k unequivocally increases the net-throughput, we arrive through a phase-transition to the state, where the network realizes its maximum achievable net-throughput. In this state, we trade-off one unit of increase of θ_k for exactly one unit of decrease of θ_R. From here, the following phase transition leads to a highly sub-optimal state, where we lose net-throughput by increasing θ_k any further. This last one is precisely the phase transition that admission control must preclude, and reductions of the throughput polytope make this decision clearly feasible.

It is noteworthy to study the region of optimal operation of the network a bit more thoroughly. We said that the network operates in the optimal region, if it attains its maximum achievable net-throughput (again, an arbitrary weighing of the commodities is possible here). For the network of Fig. 5, this region is a polytope:

$$\begin{aligned} O(G) = \{[\theta_1, \theta_2, \theta_3] : \quad & \theta_1 = 2 \\ & \theta_2 + \theta_3 = 3 \\ & 2 \leq \theta_2 \leq 3 \quad \} \end{aligned}$$

Observe that *the max-min fair allocation is not contained in the region of optimal operation* of our toy-network. Unfortunately, this is not a particularly rare occurrence. In fact, it was pointed out numerous times that a max-min fair allocation often leads to a highly sub-optimal use of the network resources, which in the worst case might lead to the loss of the half of the total achievable net-throughput [24]. This basically means that the interests of the network operator, who wants to optimize the utilization of her valuable network resources, are sacrificed for the sake of the users' fair service. A remedy to this shortcoming would be to restrict the set of permitted throughput allocations to $O(G)$ and select one particular fair allocation within this domain. This would assure that the network always works at its optimal region of operation, yet the users get a fair service. But, since one easily shows that the entire region $O(G)$ is strictly Pareto-optimal, a stronger and more sophisticated fairness concept must be invoked. As usual, max-min fairness comes to mind.

Definition 6 *Let $\Theta = \max\{c\theta : \theta \in T(G)\}$ be the maximum achievable net-throughput of a network G with respect to some weighing c of the commodities. Then, the max-min fair allocation over the set $O(G) = \{\theta : c\theta = \Theta\} \cap T(G)$ is called the optimally max-min far allocation.*

An immediate observation is that $O(G)$ is a non-empty face of $T(G)$ provided that G is regular. But any face of a polytope is again a polytope, which leads to the following observation:

Theorem 5 *Let G be a regular network. Then there exists an optimally max-min fair allocation in G, and it is unique.*

This result can be proved along the same lines as we proved the existence of a max-min fair allocation over $T(G)$. But this by and large ends all the similarities with traditional max-min fairness. While $T(G)$ has some very nice polyhedral properties (recall Property 1 and Property 2), which substantiated an entire line of interesting consequences – most importantly an algorithm and a bottleneck argumentation – unfortunately $O(G)$ does not. So we can not use the water-filling algorithm nor we can give a bottleneck argumentation for optimally max-min fair allocations. Instead, to compute it one needs to invoke successive linear programming methods like for instance Max-min Programming [26].

Fortunately, in the case of our sample network of Fig. 5, we do not need to deploy linear programming methods to find the optimally max-min fair allocation. It is easy to see that the allocation $\theta_0 = [2, 2, 1]$ is optimally max-min fair. On the one hand, this allocation clearly realizes the maximum achievable net-throughput of the network, and on the other hand it is max-min fair over the set of such throughput allocations.

11 Conclusions

In this paper, we have revisited the problem of fair allocation of resources in capacitated networks. The central line of our argumentation was that it must not be some random selection of fixed paths, but rather the specifics of the network itself that determines the emergent fair allocation. It was our intent to choose an unconventional approach to attack this problem, a polyhedral approach, which led to important new insights into the nature of several different fairness criteria in capacitated networks.

In the first part of the paper, we defined the notion of the throughput polytope and we investigated different fair allocations it gives rise to. Most notably, we showed how to compute a max-min fair allocation in the generic, routing-independent model and we gave an illustrative bottleneck argumentation. In the second part of the paper, we reviewed some important areas of telecommunications where throughput polytopes might find their use. Finally, we proposed a new type of fairness criterion, which simultaneously takes into account both the interests of the network operator and the users, letting the former to utilize the network to the extreme and arbitrating the resources fairly amongst the latter, all staying within the limitations imposed by the physical layout of the network.

Most of the work in this paper has been backed by the notion of the throughput polytope, for which we always assumed that an explicit description is at our disposal. Our simulation results show that this assumption is reasonable in generic network scenarios, though, it fails for certain artificial networks, above all if the number of commodities grows prohibitively large. Admittedly, our simulation studies need to be deepened to further justify this finding. Our algorithm would also benefit from some future improvements and optimizations. For instance, it would be interesting to see how much we cheat if we terminate the algorithm prematurely, basically trading-off running time for the precision, and what impacts using the resultant “low-order” approximation of the throughput polytope has on practical applications.

It is almost startling to realize how much we could achieve in such a well-researched area of telecommunications

and economics like fairness – and particularly max-min fairness – just by taking a polyhedral approach. Polyhedral mathematics is an established and rich discipline, and it is our best hope that this paper will help it to eventually find its well deserved place in the toolset of the networking community.

Appendix A: A Short Introduction to Polyhedra

A *polyhedron* $P \subset \mathbb{R}^n$ is the intersection of finitely many closed halfspaces. Here, the members of P are column n-vectors x and a *halfspace* $ax \leq b_0$ is the subset of $\mathbb{R}^n$ that is cut out from it by some *hyperplane* $ax = b_0$ (where a is a row n-vector and b_0 is a scalar). If P is the intersection of exactly m halfspaces, we write $P = \{x : Ax \leq b\}$, where A is a $m \times n$ matrix and b is a column m-vector. A polyhedron is *bounded* if it does not contain a ray $\{x + \lambda y : \lambda \geq 0\}$ for any $y \neq 0$. Bounded polyhedra are called *polytopes*. A *valid inequality* $ax \leq b_0$ is one for which it holds that $\forall y \in P : ay \leq b_0$. We say that a valid inequality $ax \leq b_0$ is binding (or active) at some $x_0 \in P$ if $ax_0 = b_0$. The *dimension* of a polyhedron is the dimension of its affine hull.

The boundaries of a polyhedron are called *faces*. Let $ax \leq b_0$ be a valid inequality for some polyhedron P. Now the intersection of the hyperplane $ax = b_0$ and P is called a face: $F = \{x : ax = b_0\} \cap P$. Non-empty faces of polyhedra are again polyhedra. For a polyhedron of dimension d, the faces of dimension $d - 1$ are called *facets* and of dimension 0 are *vertices*. An inequality that gives rise to a facet of the polyhedron is called a *facet-defining* inequality. A valid inequality is *redundant* in the description $P = \{x : Ax \leq b\}$, if it is already implied by the other inequalities in $Ax \leq b$. It can be proven that the set of facet-defining inequalities define an irredundant description for P, which is unique up to scalar multiplication of the rows.

Appendix B

[Proof of Theorem 2] Let the standard form of $T(G)$ be $T(G) = \{\theta \geq 0 : \beta_i \theta \leq b_i,\ i \in \{1, \ldots, N\}\}$ and let $\mathcal{B}$ be the set of constraints binding at θ_0: $\mathcal{B} = \{i \in \{1, \ldots, N\} : \beta_i \theta_0 = b_i\}$. If $\mathcal{B}$ is empty, then any coordinates of θ_0 can be increased freely, and we still remain within $T(G)$. So θ_0 is dominated, which contradicts the assumptions of the theorem. Let $\beta\theta \leq b$ be the inequality obtained by summing up all the inequalities in $\mathcal{B}$:

$$\beta = \sum_{i \in \mathcal{B}} \beta_i, \qquad b = \sum_{i \in \mathcal{B}} b_i \ .$$

First we prove that $\beta_k = 0$ if and only if k is dominated at θ_0. From Property 2: all $\beta_i \geq 0$, so $(\beta)_k = 0$ if and only if $(\beta_i)_k = 0$ for all constraints binding at θ_0. This means that $\exists \epsilon > 0$ and small enough, so that $\theta_0 + \epsilon e_k \in T(G)$, so k is dominated.

To prove that in turn $\beta_k > 0$ if and only if $k \in \mathcal{N}$, let $\theta = \theta_0 + \epsilon e_k$ for some $\epsilon > 0$ and some $k \in \mathcal{K}$ for which $(\beta)_k > 0$:

$$\beta\theta \leq \beta\theta_0 = b$$

$$(\theta)_k = (\theta_0)_k + \epsilon$$

This yields:

$$\sum_{l \in \mathcal{K} \setminus \{k\}} (\beta)_l \big[(\theta)_l - (\theta_0)_l\big] \leq -\epsilon\,(\beta)_k < 0\,.$$

Now, there are basically two choices. Either $(\theta_0)_k$ can not be increased at all (that is, it is at its single-commodity maximum flow), or otherwise, increasing it causes the decrease of the throughput of some other commodity l for which $(\beta)_l > 0$. This completes the proof of the theorem.

[Proof of Corollary 2] For each $k \in \mathcal{K}$ construct the vector θ', whose coordinates are defined as

$$(\theta')_l = \begin{cases} (\theta_0)_k & \text{if } (\theta_0)_l > (\theta_0)_k \\ (\theta_0)_l & \text{otherwise} \end{cases}$$

Observe that, by definition, exactly those commodities l are non-dominated at θ' for which $(\theta_0)_l \leq (\theta_0)_k$. All the other commodities dominated. Now, simply apply Theorem 2 to θ' to prove the Corollary.

[Proof of Theorem 4] We do not prove the Theorem directly. Instead, we prove a bit more, namely, that Theorem 2 can equivalently be posed over separating edge sets for which Property 4 holds. From this the claims of Theorem 4 follow.

Lemma 1 *Let $\theta_0 \in T(G)$ be non-dominated. Now, some set of commodities $\mathcal{N} \subseteq \mathcal{K}$ is non-dominated, and $\mathcal{K} \setminus \mathcal{N}$ is dominated at θ_0, if and only if there exists a separating edge set $\mathcal{S}$, so that*

i) $\mathcal{K}_{\mathcal{S}} = \mathcal{N}$

ii) all the routings that realize any θ', for which $\forall k \in \mathcal{N} : (\theta')_k = (\theta_0)_k$, saturate all the edges in $\mathcal{S}$.

Item (i) and (ii) only hold true, if and only if the there exists an optimal feasible solution $[f^k, \lambda, \mu]$ to the LP below, so that the optimal objective function value is zero:

$$\begin{aligned} 0 = \max & \textstyle\sum_{(i,j)\in\mathcal{S}} \lambda_{ij} + \sum_{k\in\mathcal{K}_{\mathcal{S}}} \mu_k & \\ & \textstyle\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k:(i,j)\in P} f_P^k + \lambda_{ij} = u_{ij} & \forall (i,j) \in E \\ & \textstyle\sum_{P\in\mathcal{P}_k} f_P^k - \mu_k = (\theta_0)_k & \forall k \in \mathcal{K} \\ & \mu_k \geq 0 & \forall k \in \mathcal{K}_{\mathcal{S}} \\ & f_P^k \geq 0, \lambda_{ij} \geq 0 & \end{aligned}$$

The objective function value is zero, if and only if $\forall k \in \mathcal{K}_{\mathcal{S}} : \mu_k = 0$ and $\forall (i,j) \in \mathcal{S} : \lambda_{ij} = 0$, which exactly formulate (i) and (ii). By strong duality, the dual is also soluble and the optimal objective function value is zero:

$$0 = \min wu - \beta\theta_0 \tag{22}$$

$$\textstyle\sum_{(i,j)\in P} w_{ij} \geq \beta_k \quad \forall k \in \mathcal{K}, \forall P \in \mathcal{P}_k \tag{23}$$

$$\beta_k \geq 1 \qquad \forall k \in \mathcal{K}_{\mathcal{S}} \tag{24}$$

$$\beta_k = 0 \qquad \forall k \in \mathcal{K} \setminus \mathcal{K}_{\mathcal{S}} \tag{25}$$

$$w_{ij} \geq 1 \qquad \forall (i,j) \in \mathcal{S} \tag{26}$$

$$w_{ij} \geq 0 \qquad \forall (i,j) \in E \setminus \mathcal{S} \tag{27}$$

Let $[w, \beta]$ be an optimal feasible solution to (22)–(27). Then, the inequality $\beta\theta \leq wu$ is valid for $T(G)$ by the Japanese Theorem, it is binding at θ_0 by (22) and $\mathcal{K}_{\mathcal{S}} = \mathcal{N} = \{k \in \mathcal{K} : \beta_k > 0\}$ by (24) and (25). Hence, applying Theorem 2 to this $\beta\theta \leq wu$ completes the proof.

As a final remark, we note that if for some $k \in \mathcal{K} : (\theta_0)_k > 0$, then β_k attains the length of the shortest path, since $(\theta_0)_k > 0 \Rightarrow \exists P \in \mathcal{P}_k : f_P^k > 0 \Rightarrow \sum_{(i,j)\in P} w_{ij} = \beta_k$ by complementary slackness. This assures that Property 1 extends to bottlenecks, at least for a regular G.

Appendix

Assuming $\left[\{d > r, d > 0, r > 0, \text{Cosh}[r+d] > t > \text{Cosh}[\text{Abs}[r-d]]\}, \int \text{ArcCos}\left[\frac{\text{Cosh}[r]t - \text{Cosh}[d]}{\sqrt{t^2-1}\text{Sinh}[r]}\right] dt\right]$

$t\text{ArcCos}\left[\frac{(-\text{Cosh}[d]+t\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-1+t^2}}\right] + \text{ArcTan}\left[\frac{\sqrt{2}(t-\text{Cosh}[d]\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-\text{Csch}[r]^2(2t^2+\text{Cosh}[d]^2-4t\text{Cosh}[d]\text{Cosh}[r]+\text{Cosh}[r]^2+\text{Sinh}[d]^2+\text{Sinh}[r]^2)}}\right]\text{Cosh}[d] -$

$\Big(i(\text{Cosh}[d]-\text{Cosh}[r])\text{Log}\Big[-\Big(2i\Big(-2t-\text{Cosh}[d]^2+2(1+t)\text{Cosh}[d]\text{Cosh}[r]-\text{Cosh}[r]^2-\text{Sinh}[d]^2-i\sqrt{2}\sqrt{(\text{Cosh}[d]-\text{Cosh}[r]$

$\Big(i(\text{Cosh}[d]+\text{Cosh}[r])\text{Log}\Big[\Big(2\Big(2it-i\text{Cosh}[d]^2+2i(-1+t)\text{Cosh}[d]\text{Cosh}[r]-i\text{Cosh}[r]^2-i\text{Sinh}[d]^2+\sqrt{2}\sqrt{(\text{Cosh}[d]+\text{Cosh}[r]$

∂t%35

$\text{ArcCos}\left[\frac{(-\text{Cosh}[d]+t\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-1+t^2}}\right] - \frac{t\left(\frac{\text{Coth}[r]}{\sqrt{-1+t^2}} - \frac{t(-\text{Cosh}[d]+t\text{Cosh}[r])\text{Csch}[r]}{(-1+t^2)^{3/2}}\right)}{\sqrt{1-\frac{(-\text{Cosh}[d]+t\text{Cosh}[r])^2\text{Csch}[r]^2}{-1+t^2}}} + \Bigg((-1+t)(\text{Cosh}[d]-\text{Cosh}[r])^2\Bigg(-\frac{2i\Big(-2+2\text{Cosh}[d]\text{Cosh}[r]+\frac{}{\sqrt{}}}{(-1+t)(0}$

$\Bigg(i(1+t)(\text{Cosh}[d]+\text{Cosh}[r])^2\Bigg(\frac{2\Big(2i+2i\text{Cosh}[d]\text{Cosh}[r]-\frac{\sqrt{(\text{Cosh}[d]+\text{Cosh}[r])^2}(-4\text{Cosh}[d]\text{Coth}[r]+4t\text{Csch}[r])}{\sqrt{2}\sqrt{-\text{Csch}[r](-4t\text{Cosh}[d]\text{Coth}[r]+(2t^2+\text{Cosh}[2d]+\text{Cosh}[2r])\text{Csch}[r])}}\Big)}{(1+t)(\text{Cosh}[d]+\text{Cosh}[r])\sqrt{(\text{Cosh}[d]+\text{Cosh}[r])^2}} - \Big(2\Big(2it-i\text{Cosh}[d]^2+2$

$\Big(\text{Cosh}[d]\Big(((4t-4\text{Cosh}[d]\text{Cosh}[r])(t-\text{Cosh}[d]\text{Cosh}[r])\text{Csch}[r]^3)\,/\,\Big(\sqrt{2}\,\big(-\text{Csch}[r]^2\,\big(2t^2+\text{Cosh}[d]^2-4t\text{Cosh}[d]\text{Cosh}[r]+\text{Co}$

FullSimplify[%36]

$Aborted

auxf3[r_, d_, t_]:=*t***ArcCos**$\left[\frac{(-\text{Cosh}[d]+t\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-1+t^2}}\right] +$

$$\text{ArcTan}\left[\frac{\sqrt{2}(t-\text{Cosh}[d]\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-\text{Csch}[r]^2(2t^2+\text{Cosh}[d]^2-4t\text{Cosh}[d]\text{Cosh}[r]+\text{Cosh}[r]^2+\text{Sinh}[d]^2+\text{Sinh}[r]^2)}}\right]$$

$$\text{Cosh}[d]-$$

$$(i(\text{Cosh}[d]-\text{Cosh}[r])$$

$$\text{Log}[$$

$$-\left(2i\left(-2t-\text{Cosh}[d]^2+2(1+t)\text{Cosh}[d]\text{Cosh}[r]-\text{Cosh}[r]^2-\text{Sinh}[d]^2-\right.\right.$$

$$i\sqrt{2}\sqrt{(\text{Cosh}[d]-\text{Cosh}[r])^2}$$

$$\sqrt{-\text{Csch}[r]\left(-4t\text{Cosh}[d]\text{Coth}[r]+\left(2t^2+\text{Cosh}[2d]+\text{Cosh}[2r]\right)\text{Csch}[r]\right)}\text{Sinh}[r]-$$

$$\left.\left.\text{Sinh}[r]^2\right)\right)/\left((-1+t)(\text{Cosh}[d]-\text{Cosh}[r])\sqrt{(\text{Cosh}[d]-\text{Cosh}[r])^2}\right)\Big]\Big)\Big/$$

$$\left(2\sqrt{(\text{Cosh}[d]-\text{Cosh}[r])^2}\right)+$$

$$(i(\text{Cosh}[d]+\text{Cosh}[r])$$

$$\text{Log}[$$

$$\left(2\left(2it-i\text{Cosh}[d]^2+2i(-1+t)\text{Cosh}[d]\text{Cosh}[r]-i\text{Cosh}[r]^2-i\text{Sinh}[d]^2+\right.\right.$$

$$\sqrt{2}\sqrt{(\text{Cosh}[d]+\text{Cosh}[r])^2}$$

$$\sqrt{-\text{Csch}[r]\left(-4t\text{Cosh}[d]\text{Coth}[r]+\left(2t^2+\text{Cosh}[2d]+\text{Cosh}[2r]\right)\text{Csch}[r]\right)}\text{Sinh}[r]-$$

$$\left.\left.i\text{Sinh}[r]^2\right)\right)/\left((1+t)(\text{Cosh}[d]+\text{Cosh}[r])\sqrt{(\text{Cosh}[d]+\text{Cosh}[r])^2}\right)\Big]\Big)\Big/$$

$$\left(2\sqrt{(\text{Cosh}[d]+\text{Cosh}[r])^2}\right)$$

(*ennekazintegralnakduv − rvtolduv + rvigpontosan$a$duvsugarukorrelkellmegegyeznie*)

$$\partial_t\left(t\text{ArcCos}\left[\frac{(-\text{Cosh}[d]+t\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-1+t^2}}\right]+\right.$$

$$\left.\text{ArcTan}\left[\frac{\sqrt{2}(t-\text{Cosh}[d]\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-\text{Csch}[r]^2(2t^2+\text{Cosh}[d]^2-4t\text{Cosh}[d]\text{Cosh}[r]+\text{Cosh}[r]^2+\text{Sinh}[d]^2+\text{Sinh}[r]^2)}}\right]\text{Cosh}[d]\right)$$

$$\text{ArcCos}\left[\frac{(-\text{Cosh}[d]+t\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-1+t^2}}\right]-\frac{t\left(\frac{\text{Coth}[r]}{\sqrt{-1+t^2}}-\frac{t(-\text{Cosh}[d]+t\text{Cosh}[r])\text{Csch}[r]}{(-1+t^2)^{3/2}}\right)}{\sqrt{1-\frac{(-\text{Cosh}[d]+t\text{Cosh}[r])^2\text{Csch}[r]^2}{-1+t^2}}}+\left(\text{Cosh}[d]\left(\left((4t-4\text{Cosh}[d]\text{Cosh}[r])(t-\text{Cosh}[d]\text{Cosh}[r])\text{Cs}\right.\right.\right.$$

FullSimplify[%4]

$$\text{ArcCos}\left[\frac{(-\text{Cosh}[d]+t\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-1+t^2}}\right]+\frac{\sqrt{2}\text{Cosh}[d]\text{Csch}[r]}{\sqrt{-(\text{Cosh}[2d]+2t(t-2\text{Cosh}[d]\text{Cosh}[r])+\text{Cosh}[2r])\text{Csch}[r]^2}}+\frac{t(\text{Coth}[r]-t\text{Cosh}[d]\text{Csch}[r])}{(-1+t^2)^{3/2}\sqrt{1-\frac{(t\text{Coth}[r]-\text{Cosh}[d]\text{Csch}[r])^2}{-1+t^2}}}$$

(*==========*)

Clear[r, d, t]

Assuming$\left[\{d>0, r>0, t>\text{Cosh}[r-d], t<\text{Cosh}[r+d]\}, \int_{d-r}^{d+r}\text{ArcCos}\left[\frac{\text{Cosh}[r]t-\text{Cosh}[d]}{\sqrt{t^2-1}\text{Sinh}[r]}\right]\,dt\right]$

$Aborted

(*————————————— − *)

Cosh[5.]

74.2099

Cosh[15.]

1.63451×10^6

$2\pi(-1+\text{Cosh}[\text{Max}[0,5.]])+2\text{auxf3}[5.,10.,\text{Cosh}[15.]]-2\text{auxf3}[5.,10.,\text{Cosh}[5.]]$

$-12.5664-0.000384372i$

(*————————*)

$\text{f3}[d\_,r\_,t\_]:=t\text{ArcCos}\left[\frac{(-\text{Cosh}[d]+t\text{Cosh}[r])\text{Csch}[r]}{\sqrt{-1+t^2}}\right]+$

$\text{ArcTan}[$

$\left(\sqrt{\left(\frac{1}{-1+\text{Cosh}[2r]}\left(-2t^2-\text{Cosh}[2d]+2t\text{Cosh}[d-r]-\text{Cosh}[2r]+2t\text{Cosh}[d+r]\right)\right)}\right.$

$\left.(-\text{Sinh}[d-2r]-4t\text{Sinh}[r]+\text{Sinh}[d+2r])\right)/$

$\left(2\left(2t^2+\text{Cosh}[2d]-2t\text{Cosh}[d-r]+\text{Cosh}[2r]-2t\text{Cosh}[d+r]\right)\right)]\,\text{Cosh}[d]-$

$(i(\text{Cosh}[d]-\text{Cosh}[r])$

$\text{Log}[$

$-\left(2i\sqrt{2}(2t+\text{Cosh}[2d]-\text{Cosh}[d-r]-t\text{Cosh}[d-r]+\text{Cosh}[2r]-\text{Cosh}[d+r]-\right.$

$\left.t\text{Cosh}[d+r]\right))/$

$\left((-1+t)(-\text{Cosh}[d]+\text{Cosh}[r])\sqrt{2+\text{Cosh}[2d]-2\text{Cosh}[d-r]+\text{Cosh}[2r]-2\text{Cosh}[d+r]}\right)+$

$\left.\frac{4\sqrt{\frac{-2t^2-\text{Cosh}[2d]+2t\text{Cosh}[d-r]-\text{Cosh}[2r]+2t\text{Cosh}[d+r]}{-1+\text{Cosh}[2r]}}\text{Sinh}[r]}{(-1+t)(-\text{Cosh}[d]+\text{Cosh}[r])}\right]\Bigg)\Bigg/$

$\left(\sqrt{2}\sqrt{2+\text{Cosh}[2d]-2\text{Cosh}[d-r]+\text{Cosh}[2r]-2\text{Cosh}[d+r]}\right)-$

$(i(-\text{Cosh}[d]-\text{Cosh}[r])$

$\text{Log}[$

$-\left(2i\sqrt{2}(-2t+\text{Cosh}[2d]+\text{Cosh}[d-r]-t\text{Cosh}[d-r]+\text{Cosh}[2r]+\text{Cosh}[d+r]-\right.$

$\left.t\text{Cosh}[d+r]\right))/$

$\left((1+t)(\text{Cosh}[d]+\text{Cosh}[r])\sqrt{2+\text{Cosh}[2d]+2\text{Cosh}[d-r]+\text{Cosh}[2r]+2\text{Cosh}[d+r]}\right)+$

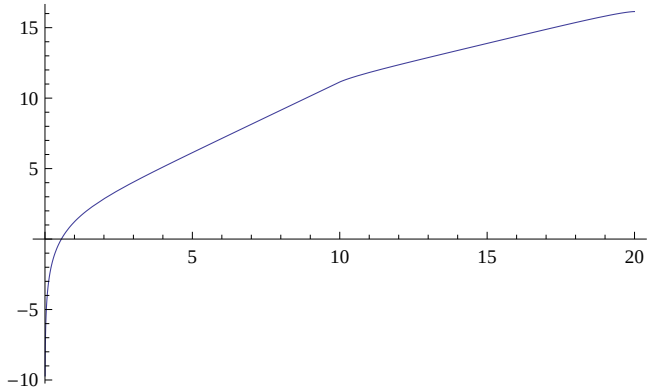
$\left.\frac{4\sqrt{\frac{-2t^2-\text{Cosh}[2d]+2t\text{Cosh}[d-r]-\text{Cosh}[2r]+2t\text{Cosh}[d+r]}{-1+\text{Cosh}[2r]}}\text{Sinh}[r]}{(1+t)(\text{Cosh}[d]+\text{Cosh}[r])}\right]\Bigg)\Bigg/$

$\left(\sqrt{2}\sqrt{2+\text{Cosh}[2d]+2\text{Cosh}[d-r]+\text{Cosh}[2r]+2\text{Cosh}[d+r]}\right)$

$\text{ter3}[r\_,d\_,rr\_]:=\text{If}[r+d<rr,2\pi(\text{Cosh}[d]-1),$

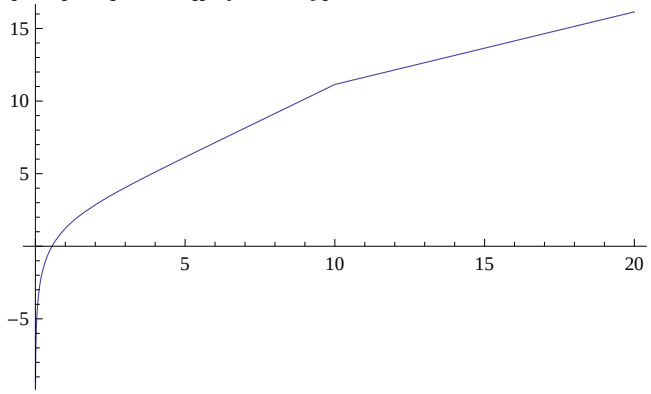
$2\pi(-1+\text{Cosh}[\text{Max}[0,d-r]])+2(\text{f3}[d,r,\text{Cosh}[.9999rr]]-\text{f3}[d,r,1.0001\text{Cosh}[\text{Abs}[d-r]]])]$

$\text{Plot}[\text{Log}[\text{ter3}[5.,d,15.]],\{d,0,20.\}]$

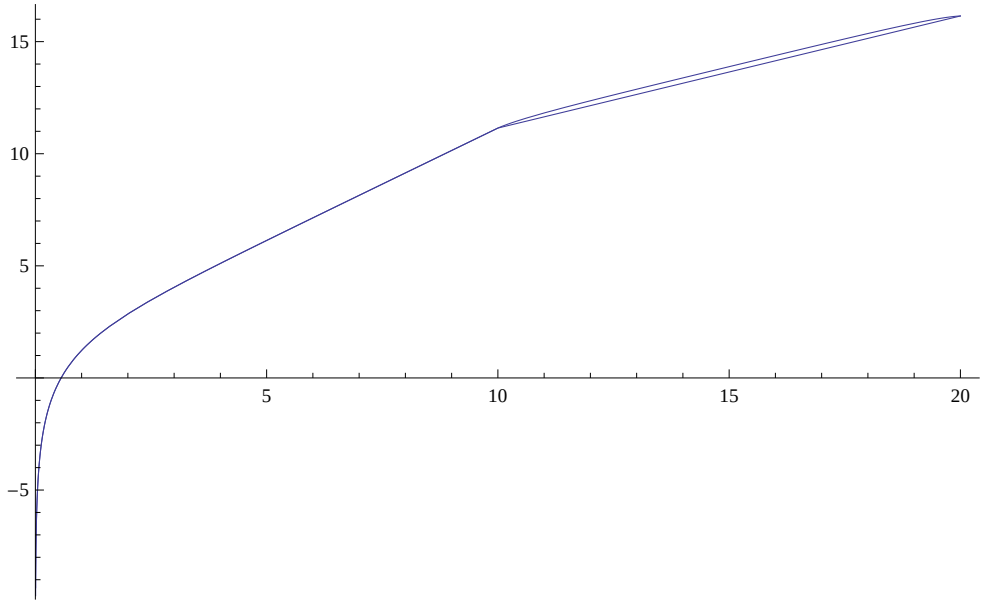


ter4[r_, d_, rr_]:=If $\left[r + d < \text{rr}, 2\pi(\text{Cosh}[d] - 1), \pi\text{Exp}\left[\frac{\text{rr}-r}{2}\right]\text{Exp}[d/2] + \pi\text{Exp}[-\text{rr} + r] - 2\pi\right]$

Plot[Log[ter4[5, d, 15]], {d, 0, 20}]



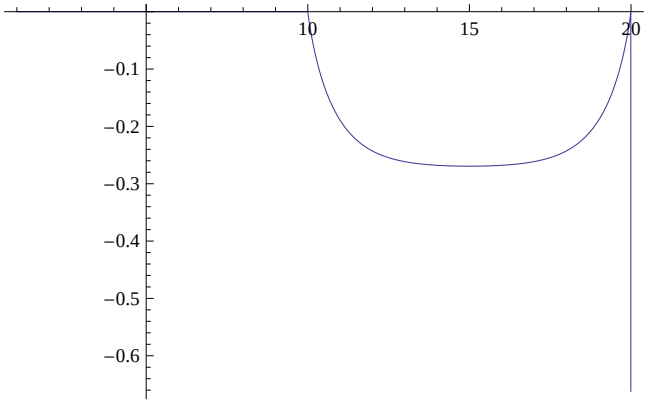
Show[%38, %14]



1 − Log[Re[ter3[5., 15, 15]]]/Log[ter4[5, 15, 15]]

-0.017484

Plot[1 − Re[ter3[5., d, 15]]/ter4[5, d, 15], {d, 1, 20}]



ru = 1

1

NIntegrate $\left[e^{-\text{ter4}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\text{ru}]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\text{ru}]],15]\cdot 001}, \{r, 0, 15\}, \{\phi, 0, 2\pi\}\right]

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

31.5438

ru = 9.5

9.5

.001NIntegrate $\left[e^{-\text{ter4}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\text{ru}]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\text{ru}]],15]\cdot 001}\text{Sinh}[r], \{r, 0, 15\},$

$\{\phi, 0, 2\pi\}\right]$

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

4.9934

$\{\#,$ .001NIntegrate $\left[e^{-\text{ter4}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]\cdot 001}\text{Sinh}[r],$

$\{r, 0, 15\}, \{\phi, 0, 2\pi\}\right]\}\&/@\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.⟩⟩

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

{{1, 65.2028}, {2, 113.374}, {3, 129.112}, {4, 110.646}, {5, 80.3911}, {6, 53.3646}, {7, 33.7975}, {8, 20.9112}, {9, 12.7949}, {1

Append[%61, {9.5, 4.994}]

list1 = {{1, "65.2028"}, {2, "113.374"}, {3, "129.112"},

{4, "110.646"}, {5, "80.3911"}, {6, "53.3646"},

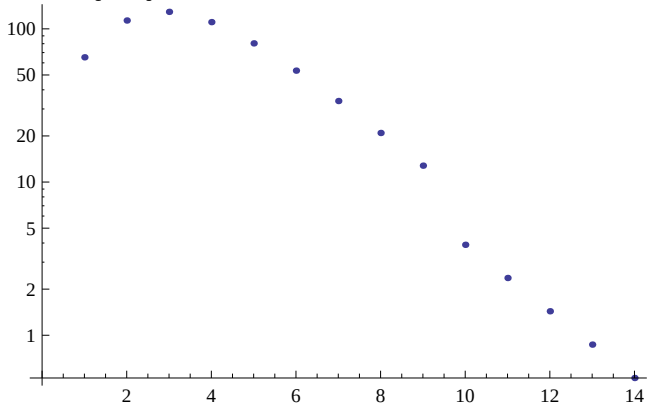
{7, "33.7975"}, {8, "20.9112"}, {9, "12.7949"},

{10, "3.8942"}, {11, "2.36468"}, {12, "1.43408"},

{13, "0.868946"}, {14, "0.526069"}, {15, "0."}}

$\{\{1, 65.2028\}, \{2, 113.374\}, \{3, 129.112\}, \{4, 110.646\}, \{5, 80.3911\}, \{6, 53.3646\}, \{7, 33.7975\}, \{8, 20.9112\}, \{9, 12.7949\}, \{1$

ListLogPlot[list1]



list2 = {}

$\{\}$

For[$i = 1, i < 15, i = i + 1$, list2 = Append[list2, {list1[[i]][[2]], Sinh[list1[[i]][[1]]]}]]

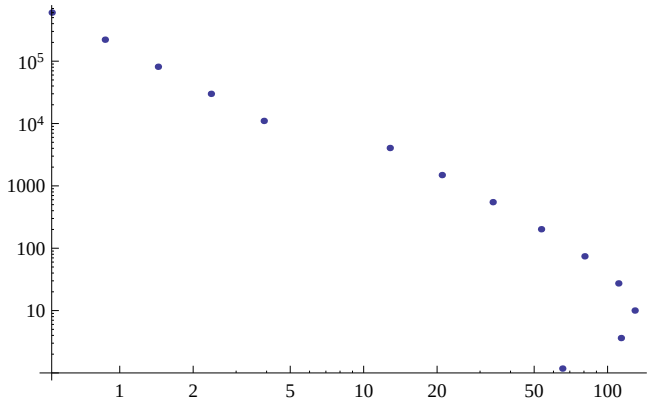
list1[[5]][[1]]

5

list2

$\{\{65.2028, \text{Sinh}[1]\}, \{113.374, \text{Sinh}[2]\}, \{129.112, \text{Sinh}[3]\}, \{110.646, \text{Sinh}[4]\}, \{80.3911, \text{Sinh}[5]\}, \{53.3646, \text{Sinh}[6]\}, \{33.7975$

ListLogLogPlot[list2]



(*10000 pont egy 15os sugaru koron*)

{#, .01NIntegrate [$e^{-\text{ter4}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]\cdot 01}\text{Sinh}[r]$,

{r, 0, 15}, {ϕ, 0, 2π}]}&/@{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.⟩⟩

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

General::stop : Further output of \!\(*StyleBox[NIntegrate::ncvb, "MT"]\) will be suppressed during this calculation.⟩⟩

$\{\{1, 1.\}, \{2, 1.\}, \{3, 1.00387\}, \{4, 0.841218\}, \{5, 0.594113\}, \{6, 0.3672\}, \{7, 0.207322\}, \{8, 0.106548\}, \{9, 0.0236808\}, \{10, 3.67$

(*5000 pont 15os koron*)

$\{\#, .005\text{NIntegrate}\left[e^{-\text{ter4}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]}.005\,\text{Sinh}[r],\right.$

$\left.\{r, 0, 15\}, \{\phi, 0, 2\pi\}\right]\}\&/@\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.⟩⟩

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

General::stop : Further output of \!\(*StyleBox[NIntegrate::ncvb, "MT"]\) will be suppressed during this calculation.⟩⟩

$\{\{1, 1.00014\}, \{2, 1.05318\}, \{3, 2.54931\}, \{4, 10.117\}, \{5, 21.4296\}, \{6, 0.270572\}, \{7, 0.155182\}, \{8, 0.0838291\}, \{9, 0.041244$

(*5000 pont 16.5os koron*)

$5000./(2\pi\text{Cosh}[16.5]16.5 - 2\pi)$

6.58380924797645*^-6

$\{\#, 6.58380924797645\text{*^-6}$

$\text{NIntegrate}\left[e^{-\text{ter4}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]6.58380924797645\text{*^-6}}\text{Sinh}[r],\right.$

$\left.\{r, 0, 16.5\}, \{\phi, 0, 2\pi\}\right]\}\&/@\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

$\{\{1, 289.88\}, \{2, 283.672\}, \{3, 272.577\}, \{4, 255.039\}, \{5, 229.014\}, \{6, 193.104\}, \{7, 148.641\}, \{8, 101.824\}, \{9, 62.3376\}, \{10$

$\{\#, 6.58380924797645\text{*^-6}$

$\text{NIntegrate}\left[e^{-\text{ter4}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]6.58380924797645\text{*^-6}}\text{Sinh}[r],\right.$

$\left.\{r, 0, 16.5\}, \{\phi, 0, 2\pi\}\right]\}\&/@\{16\}$

$\{\{16, 1.73062\}\}$

{{1, "289.88"}, {2, "283.672"}, {3, "272.577"},

{4, "255.039"}, {5, "229.014"}, {6, "193.104"},

{7, "148.641"}, {8, "101.824"}, {9, "62.3376"},

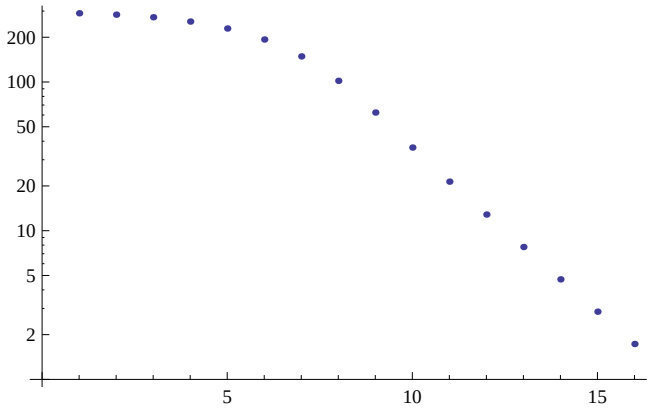
{10, "36.2403"}, {11, "21.3731"}, {12, "12.8455"},

{13, "7.7683"}, {14, "4.70679"}, {15, "2.85372"},

{16, "1.73062"}}

$\{\{1, 289.88\}, \{2, 283.672\}, \{3, 272.577\}, \{4, 255.039\}, \{5, 229.014\}, \{6, 193.104\}, \{7, 148.641\}, \{8, 101.824\}, \{9, 62.3376\}, \{10$

ListLogPlot[%143]



{#, For[$i = 1$; list2 = {}, $i < 17$, $i = i + 1$,

list2 = Append[list2, {#[[i]][[2]], Sinh[#[[i]][[1]]]}]]}&/@{%143}

$\{\{\{\{1, 289.88\}, \{2, 283.672\}, \{3, 272.577\}, \{4, 255.039\}, \{5, 229.014\}, \{6, 193.104\}, \{7, 148.641\}, \{8, 101.824\}, \{9, 62.3376\}, \{$

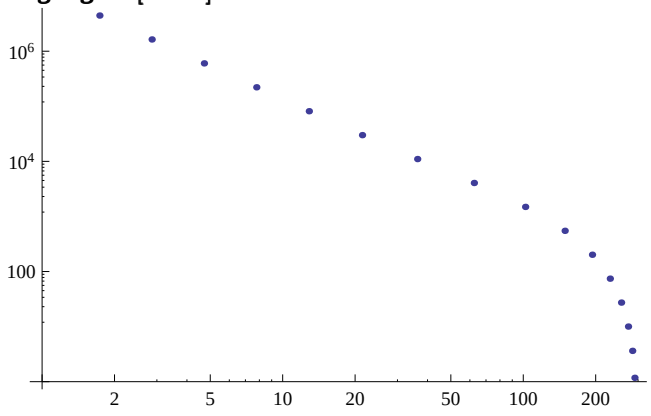
list2

$\{\{289.88, \text{Sinh}[1]\}, \{283.672, \text{Sinh}[2]\}, \{272.577, \text{Sinh}[3]\}, \{255.039, \text{Sinh}[4]\}, \{229.014, \text{Sinh}[5]\}, \{193.104, \text{Sinh}[6]\}, \{148.641,$

Length[list2]

16

ListLogLogPlot[%157]



(*A terulet elemi szamitasa*)

(*eloszor$a$koszinusztetelboldefinialjuk$a$szogfuggvenyt, haromkulonbozoszogszamitasaraisszuksegleszerre*)

angle[a_, b_, c_]:=ArcCos $\left[\frac{\text{Cosh}[a]\text{Cosh}[c]-\text{Cosh}[b]}{\text{Sinh}[a]\text{Sinh}[c]}\right]$

(*ezaboldallalszembenlevoszogetadjaki....*)

angle[5., 8., 10.]

0.0588819

angle[8., 5., 10.] + angle[5., 8., 10.] + angle[5., 10., 8.]

0.511662

angle[8., 2.0000000001, 10.] + angle[2.0000000001, 8., 10.] + angle[2.0000000001, 10., 8.]

3.14158

(*$a$fentinagyonlaposharomszogben$a$szogosszegtart$a\pi$hez...*)

angle[5., 15, 10]

$3.14159 - 2.1073424338879928$*^-8$i$

(*ennekπnekkelllennie, erremajdvigyaznikell*)

angle[15., 5., 10.]

$0.$

angle[5., 10., 15.]

$0.$

(*a kovetkezokben a terulet fuggvenyt fogjuk megadni*)

2angle[r, rr, d](Cosh[d] − 1) + 2angle[r, d, rr](Cosh[rr] − 1) − 2π+

2(angle[r, d, rr] + angle[r, rr, d] + angle[d, r, rr])

$-2\pi+2(\text{ArcCos}[(\text{Cosh}[d]\text{Cosh}[r]-\text{Cosh}[\text{rr}])\text{Csch}[d]\text{Csch}[r]]+\text{ArcCos}[(-\text{Cosh}[r]+\text{Cosh}[d]\text{Cosh}[\text{rr}])\text{Csch}[d]\text{Csch}[\text{rr}]]+$
$\text{ArcCos}[(-\text{Cosh}[d]+\text{Cosh}[r]\text{Cosh}[\text{rr}])\text{Csch}[r]\text{Csch}[\text{rr}]])+2\text{ArcCos}[(\text{Cosh}[d]\text{Cosh}[r]-\text{Cosh}[\text{rr}])\text{Csch}[d]\text{Csch}[r]](-1+$
$\text{Cosh}[d])+2\text{ArcCos}[(-\text{Cosh}[d]+\text{Cosh}[r]\text{Cosh}[\text{rr}])\text{Csch}[r]\text{Csch}[\text{rr}]](-1+\text{Cosh}[\text{rr}])$

ter[r_, d_, rr_]:=If[$r + d \leq$ rr, 2π(Cosh[d] − 1),

−2π+

2(ArcCos[(Cosh[d]Cosh[r] − Cosh[rr])Csch[d]Csch[r]]+

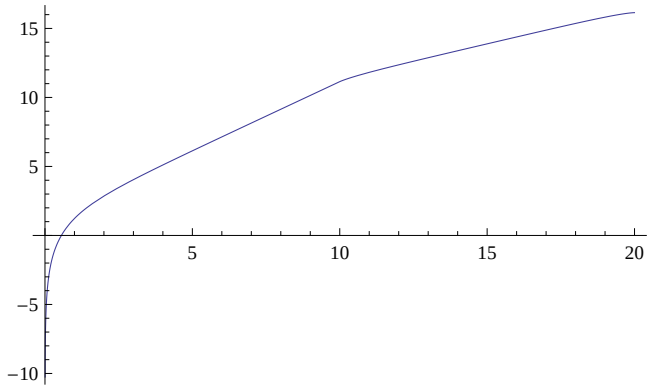
ArcCos[(−Cosh[r] + Cosh[d]Cosh[rr])Csch[d]Csch[rr]]+

ArcCos[(−Cosh[d] + Cosh[r]Cosh[rr])Csch[r]Csch[rr]])+

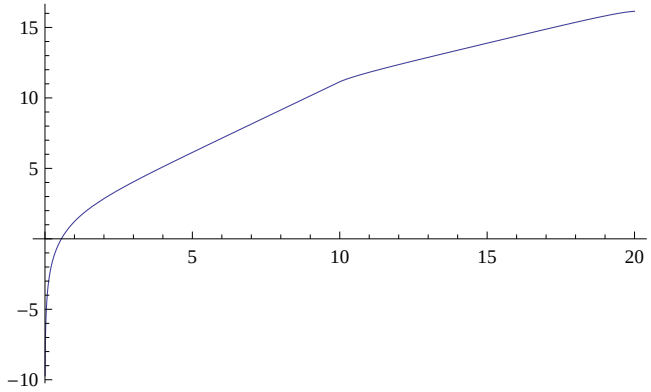
2ArcCos[(Cosh[d]Cosh[r] − Cosh[rr])Csch[d]Csch[r]](−1 + Cosh[d])+

2ArcCos[(−Cosh[d] + Cosh[r]Cosh[rr])Csch[r]Csch[rr]](−1 + Cosh[rr])]

Plot[Log[ter3[5, d, 15]], {d, 0, 20}]

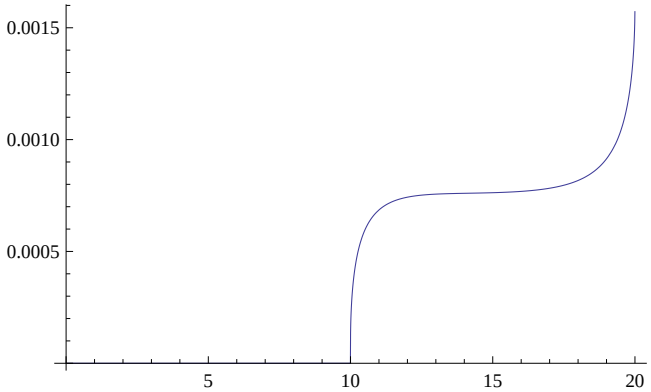


Plot[Log[ter[5, d, 15]], {d, 0, 20}]



(*a relativ kulonbseg*)

Plot[ter[5, d, 15]/ter3[5, d, 15] − 1, {d, 0, 20}]



ter[5, 10., 15]

69191.9

(*10000 pont 15 os sugaru koron*)

(*$\delta = N/T =$ kb.001*)

10000./(2π(Cosh[15] − 1))

0.000973718

{#, .001NIntegrate [$e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15].001}$ Sinh[r],

{r, 0, 15}, {ϕ, 0, 2π}]}&/@{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

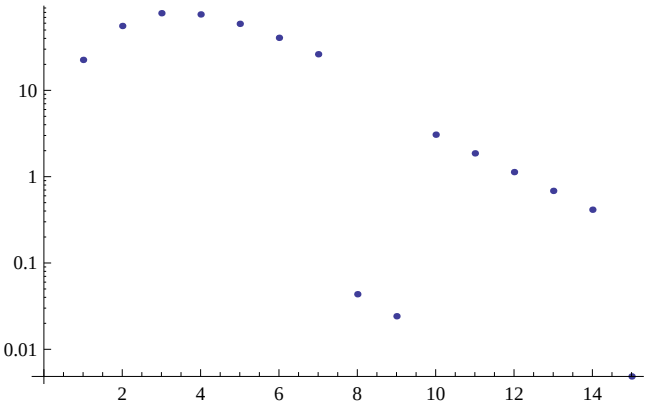
NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

{{1, 22.5355}, {2, 55.691}, {3, 78.2955}, {4, 75.8166}, {5, 58.9798}, {6, 40.6113}, {7, 26.2064}, {8, 0.0434689}, {9, 0.0241564}, 1.0992942418331244*^-17i}}

ListLogPlot[%54]



{#, .001NIntegrate [$e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15].001}$ Sinh[r],

{r, 0, 15}, {ϕ, 0, 2π}]}&/@{7.2, 7.3, 7.5, 7.7, 7.8, 7.9, 8.}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

{{7.2, 23.8966}, {7.3, 22.8095}, {7.5, 20.7659}, {7.7, 18.8886}, {7.8, 18.0093}, {7.9, 0.0459925}, {8., 0.0434689}}

{#, .001NIntegrate [$e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15].001}$ Sinh[r],

{r, 0, 15}, {ϕ, 0, 2π}]}&/@{8.2, 8.4, 8.6, 8.8, 9, 9.2, 9.4, 9.6}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

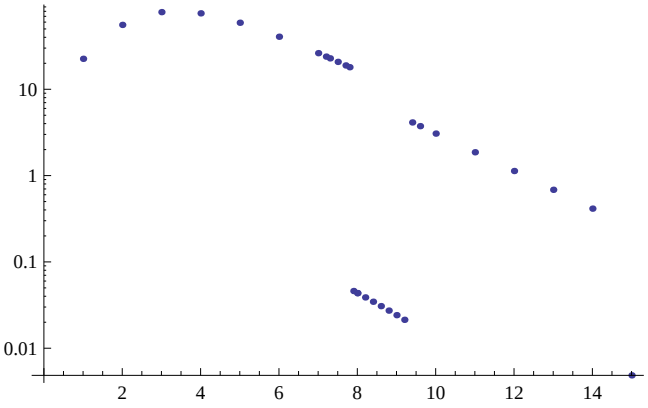
NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

{{8.2, 0.0387872}, {8.4, 0.034555}, {8.6, 0.0307305}, {8.8, 0.0272757}, {9, 0.0241564}, {9.2, 0.0213415}, {9.4, 4.12747}, {9.6,

ListLogPlot[Union[%54, %58, %59]]



(*talan megin komplex valahol a terulet*)

{#, .001NIntegrate [$e^{-\text{Re[ter[r,ArcCosh[-Sinh[r]Sinh[#]Cos[\phi]+Cosh[r]Cosh[#]],15]].001}}$Sinh[r],

{r, 0, 15}, {\phi, 0, 2\pi}]}&/@{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

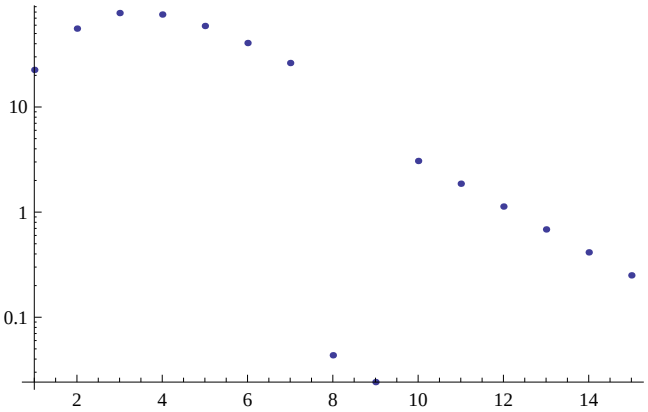
{{1, 22.5355}, {2, 55.691}, {3, 78.2955}, {4, 75.8166}, {5, 58.9798}, {6, 40.6113}, {7, 26.2064}, {8, 0.0434689}, {9, 0.0241564}

%54

{{1, 22.5355}, {2, 55.691}, {3, 78.2955}, {4, 75.8166}, {5, 58.9798}, {6, 40.6113}, {7, 26.2064}, {8, 0.0434689}, {9, 0.0241564}

1.0992942418331244*^-17 i}}

ListLogPlot[%63, PlotRange → All]



(*1000 pont egy 15 os koron*)

1000/(2\pi Cosh[15.] − 1)

0.0000973717

{#, .0001NIntegrate [$e^{-\text{ter[r,ArcCosh[-Sinh[r]Sinh[#]Cos[\phi]+Cosh[r]Cosh[#]],15].0001}}$Sinh[r],

{r, 0, 15}, {ϕ, 0, 2π}]}&/@{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 14.9}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

NIntegrate::singd : Singularity handling has failed at point \!\(*StyleBox[{r, ϕ}, "MT"]\) = \!\(*StyleBox[{14.9038, 6.28319

NIntegrate::inum : Integrand \!\(*StyleBox[$e^{-0.0001\text{If}[\langle\langle 1\rangle\rangle]}$Sinh[r], "MT"]\) is not numerical at \!\(*StyleBox[{r, ϕ}, "MT"]\

{{1, 450.438}, {2, 335.384}, {3, 213.836}, {4, 126.4}, {5, 74.9649}, {6, 45.1709}, {7, 27.3507}, {8, 16.5805}, {9, 10.0548}, {10,

{{1, "450.438"}, {2, "335.384"}, {3, "213.836"},

{4, "126.4"}, {5, "74.9649"}, {6, "45.1709"},

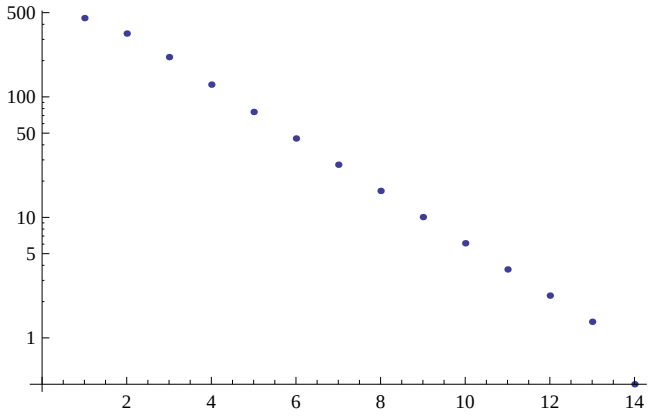
{7, "27.3507"}, {8, "16.5805"}, {9, "10.0548"},

{10, "6.09803"}, {11, "3.69836"}, {12, "2.24292"},

{13, "1.36017"}, {14, "0.412386"}}

{{1, 450.438}, {2, 335.384}, {3, 213.836}, {4, 126.4}, {5, 74.9649}, {6, 45.1709}, {7, 27.3507}, {8, 16.5805}, {9, 10.0548}, {10,

ListLogPlot[%69]



(*$N = 500, r = 10$*)

δ[n_, rr_]:=$n/(2\pi(\text{Cosh}[\text{rr}] - 1))$

δ[500, 10.]

0.00722628

{#, "0.00722628"

NIntegrate [$e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],10]\text{"0.00722628"}}$**Sinh[r],**

{r, 0, 10}, {ϕ, 0, 2π}]}&/@{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

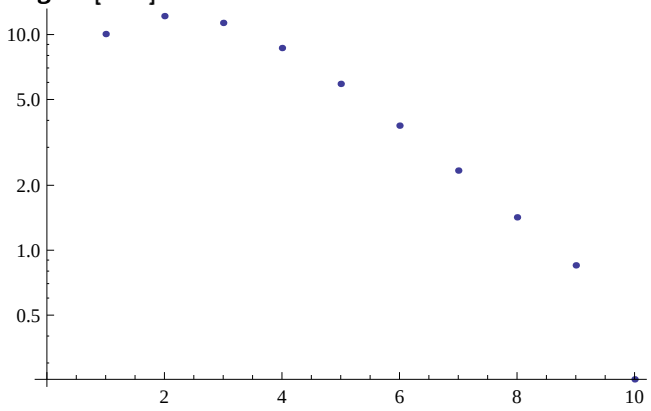
NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.⟩⟩

$\{\{1, 10.0549\}, \{2, 12.177\}, \{3, 11.3238\}, \{4, 8.66055\}, \{5, 5.90557\}, \{6, 3.78135\}, \{7, 2.34045\}, \{8, 1.4207\}, \{9, 0.850806\}, \{10$

$0.i\}\}$

ListLogPlot[%89]



Length[%73]

10

{#, For[i = 1; listaux = {}, $i \leq$ Length[#], $i = i + 1$,

listaux = Append[listaux, {#[[i]][[2]], Sinh[#[[i]][[1]]]/Sinh[Length[#]]}]]}&/@

{%89}

$\{\{\{1, 10.0549\}, \{2, 12.177\}, \{3, 11.3238\}, \{4, 8.66055\}, \{5, 5.90557\}, \{6, 3.78135\}, \{7, 2.34045\}, \{8, 1.4207\}, \{9, 0.850806\}, \{$

$0.i\}\}, \text{Null}\}\}$

listaux

{{"10.0549", Csch[10]Sinh[1]}, {"12.177", Csch[10]Sinh[2]},

{"11.3238", Csch[10]Sinh[3]}, {"8.66055", Csch[10]Sinh[4]},

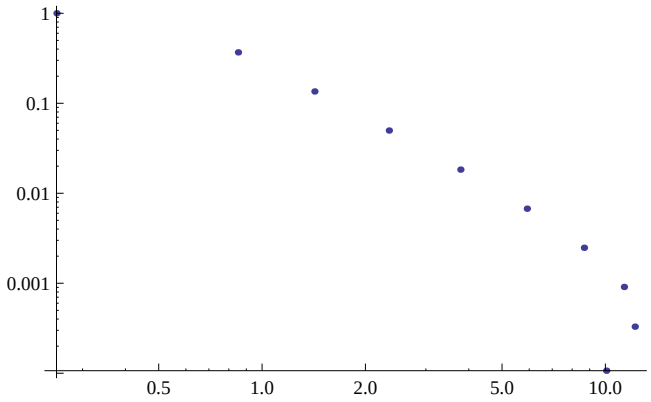
{"5.90557", Csch[10]Sinh[5]}, {"3.78135", Csch[10]Sinh[6]},

{"2.34045", Csch[10]Sinh[7]}, {"1.4207", Csch[10]Sinh[8]},

{"0.850806", Csch[10]Sinh[9]}, {"0.251909", 1}}

$\{\{10.0549, \text{Csch}[10]\text{Sinh}[1]\}, \{12.177, \text{Csch}[10]\text{Sinh}[2]\}, \{11.3238, \text{Csch}[10]\text{Sinh}[3]\}, \{8.66055, \text{Csch}[10]\text{Sinh}[4]\}, \{5.90557, \text{Csch}$

ListLogLogPlot[%106]



(*$n = 500r15$*)

δ[5000, 20.]

3.280427888351558*^-6

{#, 3.280427888351558*^-6

NIntegrate $\left[e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],20]3.280427888351558*^-6}\text{Sinh}[r]\right.$,

{r, 0, 20}, {ϕ, 0, 2π}]}&/@{1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

NIntegrate::singd : Singularity handling has failed at point \!\(*StyleBox[{r, ϕ}, "MT"]\) = \!\(*StyleBox[{18.125, 6.28319}

NIntegrate::inum : Integrand \!\(*StyleBox[$e^{-3.280427888351558*^-6\text{If}[\langle\langle 1\rangle\rangle]}\text{Sinh}[r]$, "MT"]\) is not numerical at \!\(*StyleBox[{r,

NIntegrate::singd : Singularity handling has failed at point \!\(*StyleBox[{r, ϕ}, "MT"]\) = \!\(*StyleBox[{19.4868, 6.28319

NIntegrate::inum : Integrand \!\(*StyleBox[$e^{-3.280427888351558*^-6\text{If}[\langle\langle 1\rangle\rangle]}\text{Sinh}[r]$, "MT"]\) is not numerical at \!\(*StyleBox[{r,

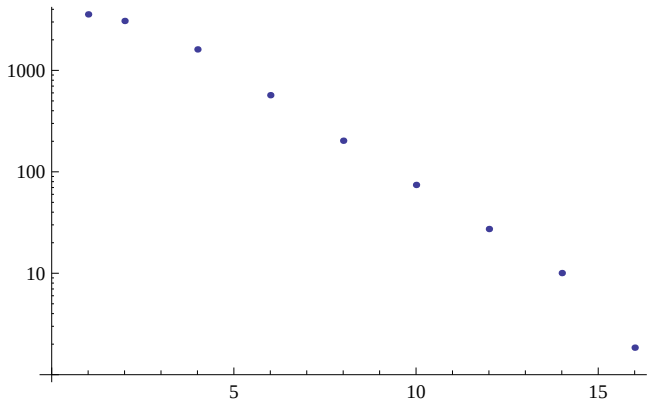
{{1, "3565.79"}, {2, "3070.88"}, {4, "1610.05"},

{6, "569.388"}, {8, "202.549"}, {10, "74.249"},

{12, "27.3021"}, {14, "10.0432"}, {16, "1.84733"}}

{{1, 3565.79}, {2, 3070.88}, {4, 1610.05}, {6, 569.388}, {8, 202.549}, {10, 74.249}, {12, 27.3021}, {14, 10.0432}, {16, 1.84733}

ListLogPlot[%111]



{#, For[$i = 1$; listaux = {}, $i \leq$ Length[#], $i = i + 1$,

listaux = Append[listaux, {#[[i]][[2]], Sinh[#[[i]][[1]]]/Sinh[20]}]]}&/@

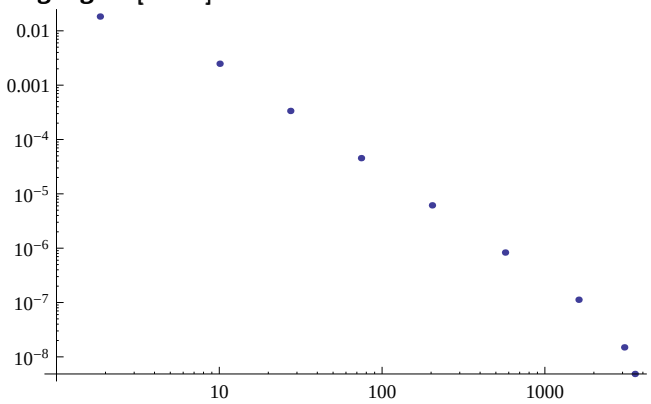
{%111}

$\{\{\{\{1, 3565.79\}, \{2, 3070.88\}, \{4, 1610.05\}, \{6, 569.388\}, \{8, 202.549\}, \{10, 74.249\}, \{12, 27.3021\}, \{14, 10.0432\}, \{16, 1.8473$

listaux

$\{\{3565.79, \text{Csch}[20]\text{Sinh}[1]\}, \{3070.88, \text{Csch}[20]\text{Sinh}[2]\}, \{1610.05, \text{Csch}[20]\text{Sinh}[4]\}, \{569.388, \text{Csch}[20]\text{Sinh}[6]\}, \{202.549, \text{Csc}$

ListLogLogPlot[%124]



(*$n = 500, r = 15$*)

δ[500., 15.]

$\delta[500., 15.]$

{#, "0.0000486859"

NIntegrate [$e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]"0.0000486859"}$ Sinh[r],

{$r, 0, 15$}, {$\phi, 0, 2\pi$}]}&/@{1, 2, 4, 6, 8, 10, 12, 14, 14.5}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

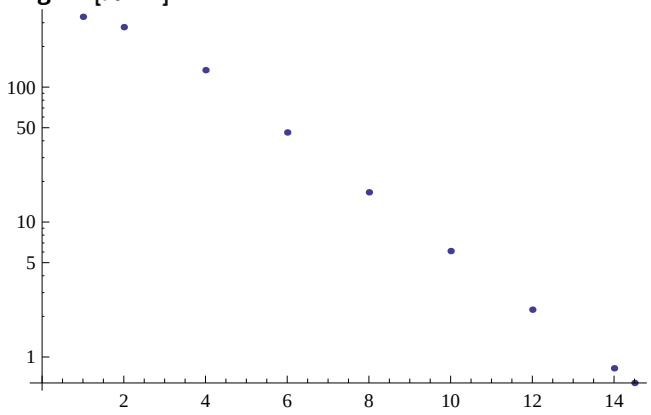
NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

$\{\{1, 331.849\}, \{2, 278.17\}, \{4, 133.567\}, \{6, 46.1349\}, \{8, 16.6112\}, \{10, 6.09664\}, \{12, 2.24197\}, \{14, 0.824568\}, \{14.5, 0.642$

ListLogPlot[%128]



{#, For[$i = 1$; listaux = {}, $i \leq$ Length[#], $i = i + 1$,

listaux = Append[listaux, {#[[i]][[2]], Sinh[#[[i]][[1]]]/Sinh[15]}]]}&/@

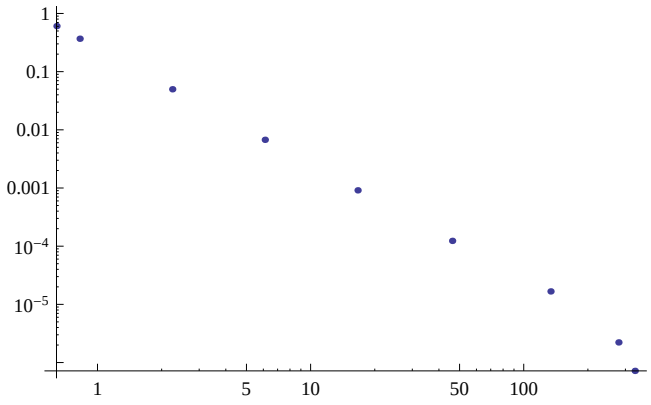
{%128}

$\{\{\{1, 331.849\}, \{2, 278.17\}, \{4, 133.567\}, \{6, 46.1349\}, \{8, 16.6112\}, \{10, 6.09664\}, \{12, 2.24197\}, \{14, 0.824568\}, \{14.5, 0.64$

listaux

$\{\{331.849, \text{Csch}[15]\text{Sinh}[1]\}, \{278.17, \text{Csch}[15]\text{Sinh}[2]\}, \{133.567, \text{Csch}[15]\text{Sinh}[4]\}, \{46.1349, \text{Csch}[15]\text{Sinh}[6]\}, \{16.6112, \text{Csch}$

ListLogLogPlot[listaux]



FullSimplify[ter[r, d, r]]

$$\begin{cases} 2\pi(-1 + \text{Cosh}[d]) & d \\ 2\left(-\pi + \text{ArcCos}\left[\text{Coth}[r]\text{Tanh}\left[\frac{d}{2}\right]\right] + \text{ArcCos}\left[\text{Coth}[r]\text{Tanh}\left[\frac{d}{2}\right]\right]\text{Cosh}[d] + \text{ArcCos}\left[\text{Coth}[r]^2 - \text{Cosh}[d]\text{Csch}[r]^2\right]\text{Cosh}[r]\right) & \text{Tr} \end{cases}$$

$$\bar{k}(r) = \frac{N}{2\pi(\cosh R-1)}\left\{2\pi(\cosh R-1)-2\cosh R\left(\arcsin\frac{\tanh(r/2)}{\tanh R}+\arctan\frac{\cosh R\sinh(r/2)}{\sqrt{\sinh(R+r/2)\sinh(R-r/2)}}\right)\right.$$
$$\left.+\arctan\frac{(\cosh R+\cosh r)\sqrt{\cosh 2R-\cosh r}}{\sqrt{2}(\sinh^2 R-\cosh R-\cosh r)\sinh(r/2)}-\arctan\frac{(\cosh R-\cosh r)\sqrt{\cosh 2R-\cosh r}}{\sqrt{2}(\sinh^2 R+\cosh R-\cosh r)\sinh(r/2)}\right\}, \quad (11)$$

2π(Cosh[R] − 1) − 2Cosh[R] (ArcSin [Tanh[r/2]/Tanh[R]] + ArcTan [Cosh[R]Sinh[r/2]/√(Sinh[R+r/2]Sinh[R−r/2])]) +
ArcTan [(Cosh[R]+Cosh[r])√(Cosh[2R]−Cosh[r])/(√2(Sinh[R]²−Cosh[R]−Cosh[r])Sinh[r/2])] −
ArcTan [(Cosh[R]−Cosh[r])√(Cosh[2R]−Cosh[r])/(√2(Sinh[R]²+Cosh[R]−Cosh[r])Sinh[r/2])] ==
−2π+
2(ArcCos[(Cosh[R]Cosh[r] − Cosh[R])Csch[R]Csch[r]]+
ArcCos[(−Cosh[r] + Cosh[R]Cosh[R])Csch[R]Csch[R]]+
ArcCos[(−Cosh[R] + Cosh[r]Cosh[R])Csch[r]Csch[R]])+
2ArcCos[(Cosh[R]Cosh[r] − Cosh[R])Csch[R]Csch[r]](−1 + Cosh[R])+
2ArcCos[(−Cosh[R] + Cosh[r]Cosh[R])Csch[r]Csch[R]](−1 + Cosh[R])

ArcTan [(Cosh[r]+Cosh[R])√(−Cosh[r]+Cosh[2R]Csch[r/2])/(√2(−Cosh[r]−Cosh[R]+Sinh[R]²))] − ArcTan [(−Cosh[r]+Cosh[R])√(−Cosh[r]+Cosh[2R]Csch[r/2])/(√2(−Cosh[r]+Cosh[R]+Sinh[R]²))] +
2π(−1 + Cosh[R]) − 2 (ArcSin [Coth[R]Tanh [r/2]] + ArcTan [Cosh[R]Sinh[r/2]/√(−Sinh[r/2−R]Sinh[r/2+R])]) Cosh[R] == −2π +
2 (2ArcCos[(−Cosh[R] + Cosh[r]Cosh[R])Csch[r]Csch[R]] + ArcCos [(−Cosh[r] + Cosh[R]²) Csch[R]²]) +
4ArcCos[(−Cosh[R] + Cosh[r]Cosh[R])Csch[r]Csch[R]](−1 + Cosh[R])

ter[r, R, R]

If[r + R ≤ R, 2π(Cosh[R] − 1), −2π + 2(ArcCos[(Cosh[R]Cosh[r] − Cosh[R])Csch[R]Csch[r]] +
ArcCos[(−Cosh[r] + Cosh[R]Cosh[R])Csch[R]Csch[R]] + ArcCos[(−Cosh[R] + Cosh[r]Cosh[R])Csch[r]Csch[R]]) +
2ArcCos[(Cosh[R]Cosh[r] − Cosh[R])Csch[R]Csch[r]](−1 + Cosh[R]) + 2ArcCos[(−Cosh[R] +
Cosh[r]Cosh[R])Csch[r]Csch[R]](−1 + Cosh[R])]

kulonbseg[r_, R_]:=
(2π(Cosh[R] − 1) − 2Cosh[R] (ArcSin [Tanh[r/2]/Tanh[R]] + ArcTan [Cosh[R]Sinh[r/2]/√(Sinh[R+r/2]Sinh[R−r/2])]) +
ArcTan [(Cosh[R]+Cosh[r])√(Cosh[2R]−Cosh[r])/(√2(Sinh[R]²−Cosh[R]−Cosh[r])Sinh[r/2])] −
ArcTan [(Cosh[R]−Cosh[r])√(Cosh[2R]−Cosh[r])/(√2(Sinh[R]²+Cosh[R]−Cosh[r])Sinh[r/2])] −
(−2π+
2(ArcCos[(Cosh[R]Cosh[r] − Cosh[R])Csch[R]Csch[r]]+
ArcCos[(−Cosh[r] + Cosh[R]Cosh[R])Csch[R]Csch[R]]+
ArcCos[(−Cosh[R] + Cosh[r]Cosh[R])Csch[r]Csch[R]])+
2ArcCos[(Cosh[R]Cosh[r] − Cosh[R])Csch[R]Csch[r]](−1 + Cosh[R])+

$$\mathbf{2ArcCos[(-Cosh[}R\mathbf{]+Cosh[}r\mathbf{]Cosh[}R\mathbf{])Csch[}r\mathbf{]Csch[}R\mathbf{]](-1+Cosh[}R\mathbf{]))}$$

$$\mathbf{Limit}\left[\left(2\pi(\mathbf{Cosh}[R]-1)-2\mathbf{Cosh}[R]\left(\mathbf{ArcSin}\left[\frac{\mathbf{Tanh}[r/2]}{\mathbf{Tanh}[R]}\right]+\mathbf{ArcTan}\left[\frac{\mathbf{Cosh}[R]\mathbf{Sinh}[r/2]}{\sqrt{\mathbf{Sinh}[R+r/2]\mathbf{Sinh}[R-r/2]}}\right]\right)+\right.\right.$$
$$\left.\mathbf{ArcTan}\left[\frac{(\mathbf{Cosh}[R]+\mathbf{Cosh}[r])\sqrt{\mathbf{Cosh}[2R]-\mathbf{Cosh}[r]}}{\sqrt{2}(\mathbf{Sinh}[R]^2-\mathbf{Cosh}[R]-\mathbf{Cosh}[r])\mathbf{Sinh}[r/2]}\right]-$$
$$\left.\left.\mathbf{ArcTan}\left[\frac{(\mathbf{Cosh}[R]-\mathbf{Cosh}[r])\sqrt{\mathbf{Cosh}[2R]-\mathbf{Cosh}[r]}}{\sqrt{2}(\mathbf{Sinh}[R]^2+\mathbf{Cosh}[R]-\mathbf{Cosh}[r])\mathbf{Sinh}[r/2]}\right]\right),r\to 0\right]$$

$$\frac{1}{2}\pi\left(-4+4\mathrm{Cosh}[R]+\frac{(-2+\mathrm{Cosh}[R])\sqrt{\frac{\mathrm{Sinh}[R]^2}{(-2+\mathrm{Cosh}[R])^2}}}{\sqrt{\mathrm{Sinh}[R]^2}}-\frac{(2+\mathrm{Cosh}[R])\sqrt{\frac{\mathrm{Sinh}[R]^2}{(2+\mathrm{Cosh}[R])^2}}}{\sqrt{\mathrm{Sinh}[R]^2}}\right)$$

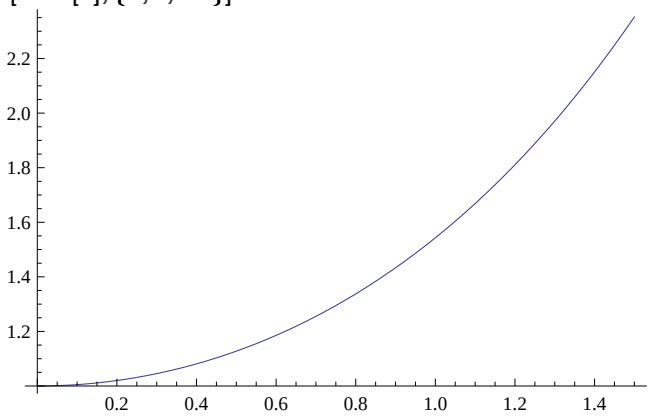
FullSimplify[%]

$$\frac{1}{2}\pi\left(-4+4\mathrm{Cosh}[R]+\frac{(-2+\mathrm{Cosh}[R])\sqrt{\frac{\mathrm{Sinh}[R]^2}{(-2+\mathrm{Cosh}[R])^2}}}{\sqrt{\mathrm{Sinh}[R]^2}}-\frac{(2+\mathrm{Cosh}[R])\sqrt{\frac{\mathrm{Sinh}[R]^2}{(2+\mathrm{Cosh}[R])^2}}}{\sqrt{\mathrm{Sinh}[R]^2}}\right)$$

$$\mathbf{Assuming}[\{R>0,\mathbf{Cosh}[R]>2\},$$
$$\mathbf{FullSimplify}\left[\frac{1}{2}\pi\left(-4+4\mathbf{Cosh}[R]+\frac{(-2+\mathbf{Cosh}[R])\sqrt{\frac{\mathbf{Sinh}[R]^2}{(-2+\mathbf{Cosh}[R])^2}}}{\sqrt{\mathbf{Sinh}[R]^2}}-\frac{(2+\mathbf{Cosh}[R])\sqrt{\frac{\mathbf{Sinh}[R]^2}{(2+\mathbf{Cosh}[R])^2}}}{\sqrt{\mathbf{Sinh}[R]^2}}\right)\right]]$$

$$2\pi(-1+\mathrm{Cosh}[R])$$

Plot[Cosh[x], {x, 0, 1.5}]



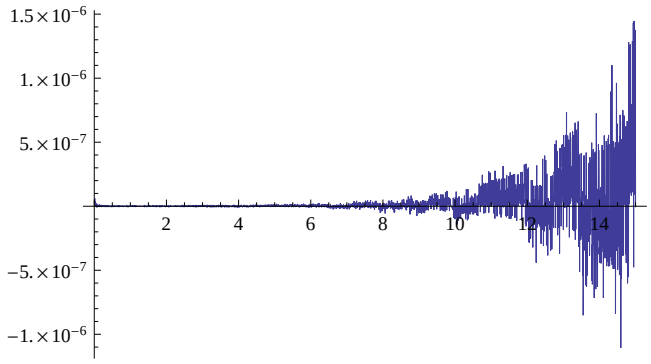
?Assuming

\!\(*RowBox[{"Assuming", "[", RowBox[{StyleBox["assum", "TI"], ",", StyleBox["expr", "TI"]}], "]"}]\) evaluates \!\(*StyleBox["expr", "TI"]\) with \!\(*StyleBox["assum", "TI"]\) appended to $Assumptions, so that \!\(*StyleBox["as "TI"]\) is included in the default assumptions used by functions such as Refine, Simplify and Integrate. ⟩⟩

***N*[kulonbseg[3, 15]]**

−8.690825836765725*^-11

Plot[*N*[kulonbseg[x, 15]], {x, 0, 15}, PlotRange → All]



kulonbseg[0.001, 19]

1.6689300537109375*^-6 − 4.2146848677759856*^-8i

kulonbseg[0, R]

Power::infy : Infinite expression \!\(*StyleBox[$\frac{1}{0}$, "MT"]\) encountered.⟩⟩

Power::infy : Infinite expression \!\(*StyleBox[$\frac{1}{0}$, "MT"]\) encountered.⟩⟩

∞::indet : Indeterminate expression \!\(*StyleBox[0Csch[R]ComplexInfinity, "MT"]\) encountered.⟩⟩

∞::indet : Indeterminate expression \!\(*StyleBox[0Csch[R]ComplexInfinity, "MT"]\) encountered.⟩⟩

∞::indet : Indeterminate expression \!\(*StyleBox[0Csch[R]ComplexInfinity, "MT"]\) encountered.⟩⟩

General::stop : Further output of \!\(*StyleBox[∞::indet, "MT"]\) will be suppressed during this calculation.⟩⟩

Indeterminate

?N

\!\(*RowBox[{"N", "[", StyleBox["expr", "TI"], "]"}]\) gives the numerical

value of \!\(*StyleBox["expr", "TI"]\). \n\!\(*RowBox[{"N", "[", RowBox[{StyleBox["expr", "TI"], ",",

StyleBox["n", "TI"]}], "]"}]\) attempts to give a result with \!\(*StyleBox["n", "TI"]\)-digit precision. ⟩⟩

(*===============*)

δ[5000., 15]

0.000486859

{#, "0.000486859"

NIntegrate [$e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]"0.000486859"}$ Sinh[r],

{r, 0, 15}, {ϕ, 0, 2π}]}&/@{1, 2, 4, 6, 8, 10, 12, 14, 14.5}

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

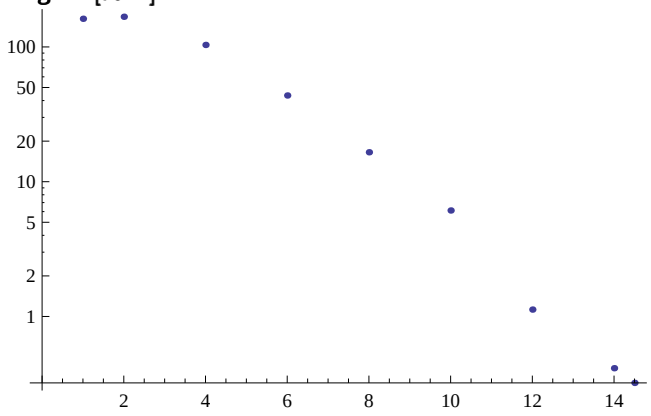
NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.⟩⟩

{{1, 161.779}, {2, 167.451}, {4, 103.471}, {6, 43.6322}, {8, 16.5357}, {10, 6.11631}, {12, 1.12525}, {14, 0.413167}, {14.5, 0.321

ListLogPlot[%50]



{#, "0.000486859"

NIntegrate $\left[e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]\text{"0.000486859"}}\text{Sinh}[r],\right.$

$\{r, 0, 15\}, \{\phi, 0, 2\pi\}]\}\&/@\text{Range}[6, 14.8, 0.5]$

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

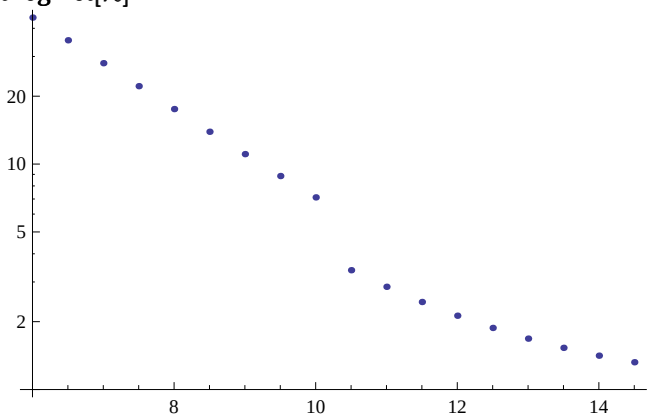
NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

$\{\{6., 43.6322\}, \{6.5, 34.4009\}, \{7., 27.014\}, \{7.5, 21.155\}, \{8., 16.5357\}, \{8.5, 12.9087\}, \{9., 10.0687\}, \{9.5, 7.84904\}, \{10., 6.1$

ListLogPlot[%]



lll = {{"6.", "43.6322"}, {"6.5", "34.4009"}, {"7.", "27.014"},

{"7.5", "21.155"}, {"8.", "16.5357"}, {"8.5", "12.9087"},

{"9.", "10.0687"}, {"9.5", "7.84904"}, {"10.", "6.11631"},

{"10.5", "2.38241"}, {"11.", "1.85561"}, {"11.5", "1.44509"},

{"12.", "1.12525"}, {"12.5", "0.876092"}, {"13.", "0.682024"},

{"13.5", "0.530877"}, {"14.", "0.413167"}, {"14.5", "0.321507"}}

$\{\{6., 43.6322\}, \{6.5, 34.4009\}, \{7., 27.014\}, \{7.5, 21.155\}, \{8., 16.5357\}, \{8.5, 12.9087\}, \{9., 10.0687\}, \{9.5, 7.84904\}, \{10., 6.1$

lll + ({0, 1}&/@Range[Length[lll]])

$\{\{6., 44.6322\}, \{6.5, 35.4009\}, \{7., 28.014\}, \{7.5, 22.155\}, \{8., 17.5357\}, \{8.5, 13.9087\}, \{9., 11.0687\}, \{9.5, 8.84904\}, \{10., 7.1$

***N*[*δ*[20000, 15]]**

0.00194744

ter[*x*, *d*, rr]

If[$d + x \leq$ rr, 2π(Cosh[d] − 1), -2π + 2(ArcCos[(Cosh[d]Cosh[x] − Cosh[rr])Csch[d]Csch[x]] + ArcCos[(−Cosh[x] + Cosh[d]Cosh[rr])Csch[d]Csch[rr]] + ArcCos[(−Cosh[d] + Cosh[x]Cosh[rr])Csch[x]Csch[rr]]) + 2ArcCos[(Cosh[d]Cosh[x] − Cosh[rr])Csch[d]Csch[x]](−1 + Cosh[d]) + 2ArcCos[(−Cosh[d] + Cosh[x]Cosh[rr])Csch[x]Csch[rr]](−1 + Cosh[rr])]

lllll =

{#, "0.00194744"

NIntegrate $\left[e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]\text{"0.00194744"}}\text{Sinh}[r]\right.$**,**

{*r*, 0, 14.8}, {*ϕ*, −*π*, *π*}]}&/@Range[0, 14.8, 1]

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

NIntegrate::eincr : The global error of the strategy GlobalAdaptive has increased more than \!\(*StyleBox[2000, "MT"]\) times.

7.13552876203328*^-11, "MT"]\) and \!\(*StyleBox[2.0843936436374634*^-11, "MT"]\) for the integral and error estimates

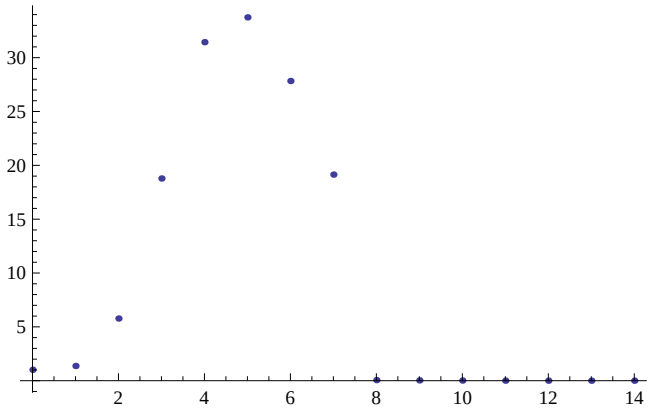
NIntegrate::eincr : The global error of the strategy GlobalAdaptive has increased more than \!\(*StyleBox[2000, "MT"]\) times.

$\{\{0, 1.01269\}, \{1, 1.36825\}, \{2, 5.77424\}, \{3, 18.7802\}, \{4, 31.4389\}, \{5, 33.7436\}, \{6, 27.833\}, \{7, 19.1408\}, \{8, 0.0544666\}, \{$

0.1^{5000}

1.0000000000004902749518849766 15.051499783199061*^-5000

ListPlot[lllll]



llllll =

{#, "0.00194744"

NIntegrate $\left[e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]\text{"0.00194744"}}\text{Sinh}[r],\right.$

$\{r, 0, 15\}, \{\phi, -\pi, \pi\}]\}\&/@\text{Range}[4, 5, 0.2]$

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

$\{\{4., 38.2182\}, \{4.2, 39.9059\}, \{4.4, 41.0104\}, \{4.6, 41.5458\}, \{4.8, 41.5498\}, \{5., 41.0766\}\}$

4.8π

15.0796

llll =

{#, "0.00194744"

NIntegrate $\left[e^{-\text{ter}[r,\text{ArcCosh}[-\text{Sinh}[r]\text{Sinh}[\#]\text{Cos}[\phi]+\text{Cosh}[r]\text{Cosh}[\#]],15]\text{"0.00194744"}}\text{Sinh}[r],\right.$

$\{r, 0, 15\}, \{\phi, 0, 2\pi\}]\}\&/@\text{Range}[6, 14.8, 0.5]$

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

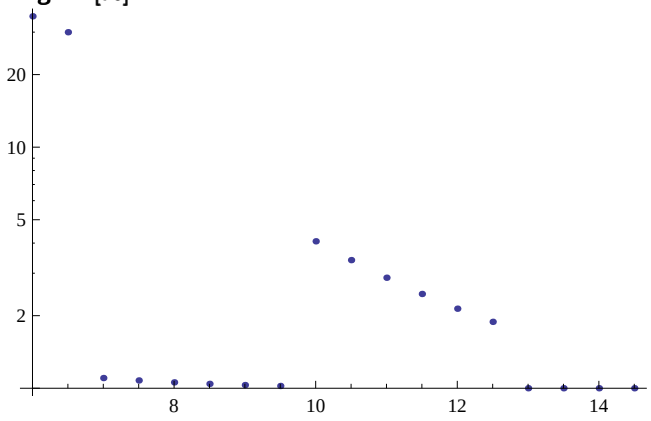
NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

General::stop : Further output of \!\(*StyleBox[NIntegrate::ncvb, "MT"]\) will be suppressed during this calculation.〉〉

$\{\{6., 33.9005\}, \{6.5, 28.9747\}, \{7., 0.102378\}, \{7.5, 0.0773255\}, \{8., 0.0578126\}, \{8.5, 0.042664\}, \{9., 0.0309513\}, \{9.5, 0.0219$

ListLogPlot[%]



llll

$\{\{6., 33.9005\}, \{6.5, 28.9747\}, \{7., 0.102378\}, \{7.5, 0.0773255\}, \{8., 0.0578126\}, \{8.5, 0.042664\}, \{9., 0.0309513\}, \{9.5, 0.0219$

llll + ({0, 1}&/@Range[Length[llll]])

$\{\{6., 34.9005\}, \{6.5, 29.9747\}, \{7., 1.10238\}, \{7.5, 1.07733\}, \{8., 1.05781\}, \{8.5, 1.04266\}, \{9., 1.03095\}, \{9.5, 1.02195\}, \{10., 4$

(*———————————— − *)

(*valamimiattnagyonlassankonvergalazintegral...*)

Exp[−ter[*r*, ArcCosh[−Sinh[*r*]Sinh[ru]Cos[*ϕ*] + Cosh[*r*]Cosh[ru]], 15]]

$e^{-\text{If}[r+\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}]-\text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]\leq 15, 2\pi(\text{Cosh}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}]-\text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]]-1), -2\pi+2(\text{ArcCos}[(\text{Cosh}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[$

ter[*r*, ArcCosh[−Sinh[*r*]Sinh[ru]Cos[*ϕ*] + Cosh[*r*]Cosh[ru]], rr]

$\text{If}[r + \text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]] \leq \text{rr}, 2\pi(\text{Cosh}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]] - 1), -2\pi + 2(\text{ArcCos}[(\text{Cosh}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]]\text{Cosh}[r] - \text{Cosh}[\text{rr}])\text{Csch}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]]\text{Csch}[r]] + \text{ArcCos}[(-\text{Cosh}[r] + \text{Cosh}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]]\text{Cosh}[\text{rr}])\text{Csch}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]]\text{Csch}[\text{rr}]] + \text{ArcCos}[(-\text{Cosh}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]] + \text{Cosh}[r]\text{Cosh}[\text{rr}])\text{Csch}[r]\text{Csch}[\text{rr}]]) + 2\text{ArcCos}[(\text{Cosh}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]]\text{Cosh}[r] - \text{Cosh}[\text{rr}])\text{Csch}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]]\text{Csch}[r]](-1 + \text{Cosh}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]]) + 2\text{ArcCos}[(-\text{Cosh}[\text{ArcCosh}[\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]]] + \text{Cosh}[r]\text{Cosh}[\text{rr}])\text{Csch}[r]\text{Csch}[\text{rr}]](-1 + \text{Cosh}[\text{rr}])]$

FullSimplify[%70]

$Aborted

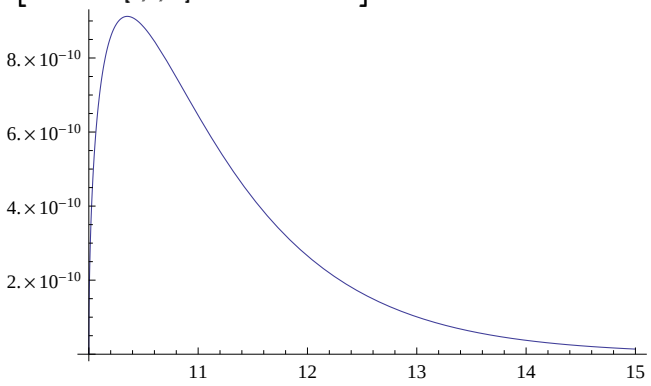
terapprox[r_, d_, rr_]:=

If[*r* + *d* ≤ rr, 2*π*(Cosh[*d*] − 1),

−2*π* + 2ArcCos[(Cosh[*d*]Cosh[*r*] − Cosh[rr])Csch[*d*]Csch[*r*]](Cosh[*d*])+

2ArcCos[(−Cosh[d] + Cosh[r]Cosh[rr])Csch[r]Csch[rr]](Cosh[rr]))]

Plot $\left[1 - \frac{\text{terapprox[5.,}d\text{,15]}}{\text{ter[5,}d\text{,15]}}, \{d, 10., 15\}\right]$



(* terapprox nagyon jo approximacio*)

FullSimplify[terapprox[r, ArcCosh[−Sinh[r]Sinh[ru]Cos[ϕ] + Cosh[r]Cosh[ru]], rr]]

terapprox[r, d, rr]

ReplaceAll[If[$d + r \leq$ rr, 2π(Cosh[d] − 1),

−2π + 2ArcCos[(Cosh[d]Cosh[r] − Cosh[rr])Csch[d]Csch[r]]Cosh[d]+

2ArcCos[(−Cosh[d] + Cosh[r]Cosh[rr])Csch[r]Csch[rr]]Cosh[rr]],

Cosh[d]-> − Sinh[r]Sinh[ru]Cos[ϕ] + Cosh[r]Cosh[ru]]

If[$d + r \leq$ rr, 2π((Cosh[r]Cosh[ru] − Cos[ϕ]Sinh[r]Sinh[ru]) − 1), −2π + 2ArcCos[((Cosh[r]Cosh[ru] − Cos[ϕ]Sinh[r]Sinh[ru])Cosh[r] − Cosh[rr])Csch[d]Csch[r]](Cosh[r]Cosh[ru] − Cos[ϕ]Sinh[r]Sinh[ru]) + 2ArcCos[(−(Cosh[r]Cosh[ru] − Cos[ϕ]Sinh[r]Sinh[ru]) + Cosh[r]Cosh[rr])Csch[r]Csch[rr]]Cosh[rr]]

Csch[ArcCosh[x]]

$\frac{1}{\sqrt{\frac{-1+x}{1+x}}(1+x)}$

FullSimplify[%81]

$\frac{1}{\sqrt{-1+x}\sqrt{1+x}}$

ReplaceAll[%82, x-> − Sinh[r]Sinh[ru]Cos[ϕ] + Cosh[r]Cosh[ru]]

$\frac{1}{\sqrt{-1+\text{Cosh}[r]\text{Cosh}[\text{ru}]-\text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]}\sqrt{1+\text{Cosh}[r]\text{Cosh}[\text{ru}]-\text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]}}$

ReplaceAll[%80, Csch[d] → %83]

If $\left[d + r \leq \text{rr}, 2\pi((\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]) - 1), -2\pi + 2\text{ArcCos}\left[(((\text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]\right.\right.$

FullSimplify[%84]

$Aborted

tuv[r_, ϕ_, ru_, rr_]:=If[−Sinh[r]Sinh[ru]Cos[ϕ] + Cosh[r]Cosh[ru] ≤ Cosh[rr − r],

2π((Cosh[r]Cosh[ru] − Cos[ϕ]Sinh[r]Sinh[ru]) − 1),

−2π+

2ArcCos[(((Cosh[r]Cosh[ru] − Cos[ϕ]Sinh[r]Sinh[ru])Cosh[r] − Cosh[rr])Csch[r])/
$\left(\sqrt{-1 + \text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]}\right.$
$\left.\sqrt{1 + \text{Cosh}[r]\text{Cosh}[\text{ru}] - \text{Cos}[\phi]\text{Sinh}[r]\text{Sinh}[\text{ru}]}\right)]$
(Cosh[r]Cosh[ru] − Cos[ϕ]Sinh[r]Sinh[ru])+
2ArcCos[(−(Cosh[r]Cosh[ru] − Cos[ϕ]Sinh[r]Sinh[ru]) + Cosh[r]Cosh[rr])Csch[r]Csch[rr]]
Cosh[rr]]
.001NIntegrate[Exp[−tuv[r, ϕ, 6., 15].001]Sinh[r], {r, 0, 15}, {ϕ, 0, 2π}]

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

40.6122

{#, "0.00194744"NIntegrate [$e^{-\text{tuv}[r,\phi,\#,15]"0.00194744"}$Sinh[$r$], {$r$, 0, 15},
{ϕ, 0, 2π}]}&/@Range[6, 14.8, 0.5]

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

NIntegrate::slwcon : Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration

General::stop : Further output of \!\(*StyleBox[NIntegrate::slwcon, "MT"]\) will be suppressed during this calculation.〉〉

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after \!\(*StyleBox[18, "MT"]\) recursive bisections in \

General::stop : Further output of \!\(*StyleBox[NIntegrate::ncvb, "MT"]\) will be suppressed during this calculation.〉〉

{{6., 33.9023}, {6.5, 28.9765}, {7., 0.102378}, {7.5, 0.0773255}, {8., 0.0578126}, {8.5, 0.042664}, {9., 0.0309513}, {9.5, 0.0219

%60

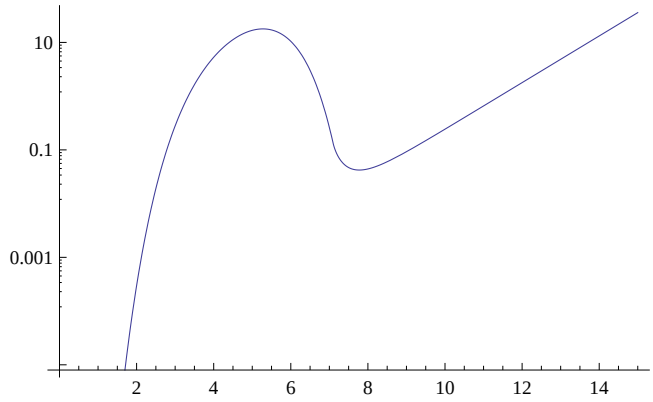
{{6., 33.9005}, {6.5, 28.9747}, {7., 0.102378}, {7.5, 0.0773255}, {8., 0.0578126}, {8.5, 0.042664}, {9., 0.0309513}, {9.5, 0.0219

(*konkluzio : $a$kozelitesnagyonjonaktunik, denemoldjafelazt$a$problemat, hogynemkonvergalazintegral...*)

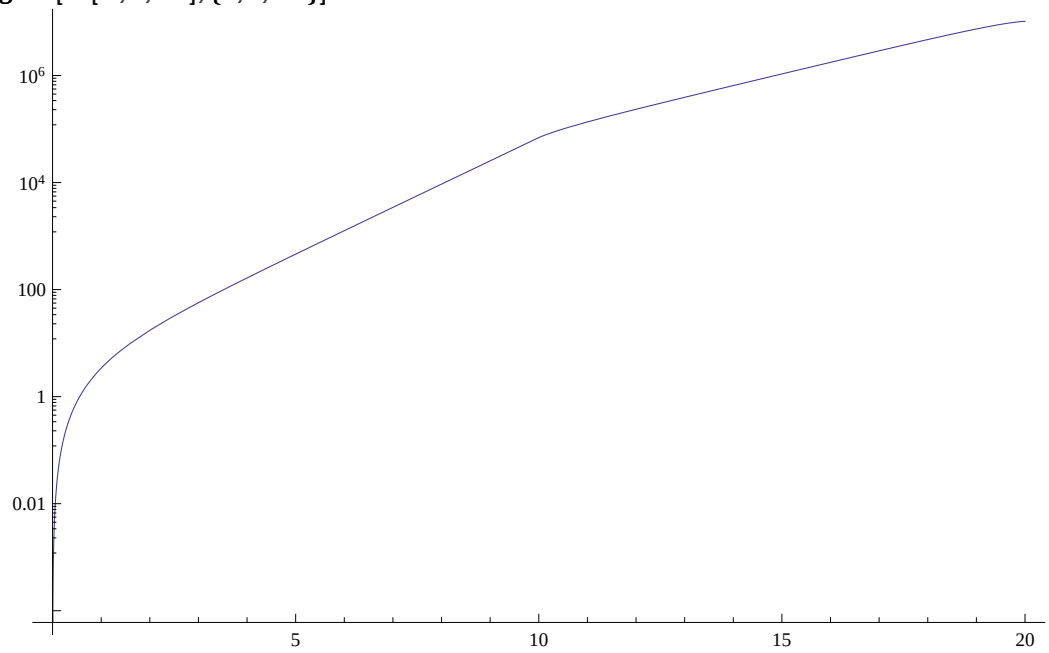
Integrate [$e^{-\text{tuv}[r,\phi,10,15].001}$Sinh[$r$], {$r$, 0, 15}]

$Aborted

LogPlot [$e^{-\text{tuv}[r,.02,10,15].001}$Sinh[$r$], {$r$, 0, 15}]




```
(*=============================================================*)
ter[r_, d_, rr_]:=If[r + d ≤ rr, 2π(Cosh[d] − 1),
−2π+
2(ArcCos[(Cosh[d]Cosh[r] − Cosh[rr])Csch[d]Csch[r]]+
ArcCos[(−Cosh[r] + Cosh[d]Cosh[rr])Csch[d]Csch[rr]]+
ArcCos[(−Cosh[d] + Cosh[r]Cosh[rr])Csch[r]Csch[rr]])+
2ArcCos[(Cosh[d]Cosh[r] − Cosh[rr])Csch[d]Csch[r]](−1 + Cosh[d])+
2ArcCos[(−Cosh[d] + Cosh[r]Cosh[rr])Csch[r]Csch[rr]](−1 + Cosh[rr])]
ter[5., d, , 15.]
LogPlot[ter[5., d, 15.], {d, 0, 20.}]
```




```
LogPlot[ter[r, 5, 15], {}]
FullSimplify[ter[r, d, r]]
```

$$\begin{cases} 2\pi(-1+\text{Cosh}[d]) & d \\ 2\left(-\pi+\text{ArcCos}\left[\text{Coth}[r]\text{Tanh}\left[\frac{d}{2}\right]\right]+\text{ArcCos}\left[\text{Coth}[r]\text{Tanh}\left[\frac{d}{2}\right]\right]\text{Cosh}[d]+\text{ArcCos}\left[\text{Coth}[r]^2-\text{Cosh}[d]\text{Csch}[r]^2\right]\text{Cosh}[r]\right) & \text{Tr} \end{cases}$$

ter[r, d, r]

$$\text{If}[d+r \le r, 2\pi(\text{Cosh}[d]-1), -2\pi+2(\text{ArcCos}[(\text{Cosh}[d]\text{Cosh}[r]-\text{Cosh}[r])\text{Csch}[d]\text{Csch}[r]]+\text{ArcCos}[(-\text{Cosh}[r]+\text{Cosh}[d]\text{Cosh}[r])\text{Csch}[d]\text{Csch}[r]]+\text{ArcCos}[(-\text{Cosh}[d]+\text{Cosh}[r]\text{Cosh}[r])\text{Csch}[r]\text{Csch}[r]])+2\text{ArcCos}[(\text{Cosh}[d]\text{Cosh}[r]-\text{Cosh}[r])\text{Csch}[d]\text{Csch}[r]](-1+\text{Cosh}[d])+2\text{ArcCos}[(-\text{Cosh}[d]+\text{Cosh}[r]\text{Cosh}[r])\text{Csch}[r]\text{Csch}[r]](-1+\text{Cosh}[r])]$$

FullSimplify[

$$-2\pi+$$

$$2(\text{ArcCos}[(\text{Cosh}[d]\text{Cosh}[r]-\text{Cosh}[r])\text{Csch}[d]\text{Csch}[r]]+$$

$$\text{ArcCos}[(-\text{Cosh}[r]+\text{Cosh}[d]\text{Cosh}[r])\text{Csch}[d]\text{Csch}[r]]+$$

$$\text{ArcCos}[(-\text{Cosh}[d]+\text{Cosh}[r]\text{Cosh}[r])\text{Csch}[r]\text{Csch}[r]])+$$

$$2\text{ArcCos}[(\text{Cosh}[d]\text{Cosh}[r]-\text{Cosh}[r])\text{Csch}[d]\text{Csch}[r]](-1+\text{Cosh}[d])+$$

$$2\text{ArcCos}[(-\text{Cosh}[d]+\text{Cosh}[r]\text{Cosh}[r])\text{Csch}[r]\text{Csch}[r]](-1+\text{Cosh}[r])]$$

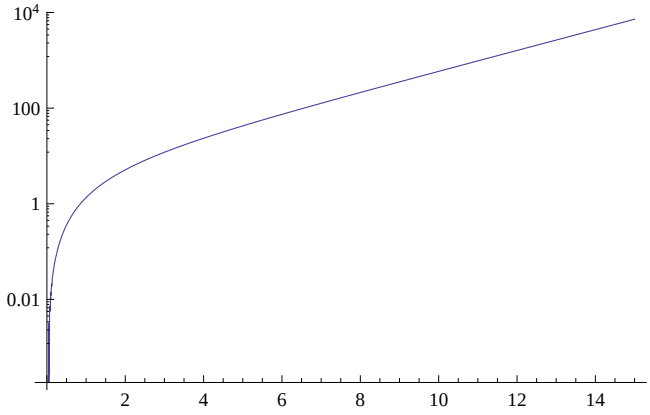
$$2\left(-\pi+\text{ArcCos}\left[\text{Coth}[r]\text{Tanh}\left[\frac{d}{2}\right]\right]+\text{ArcCos}\left[\text{Coth}[r]\text{Tanh}\left[\frac{d}{2}\right]\right]\text{Cosh}[d]+\text{ArcCos}\left[\text{Coth}[r]^2-\text{Cosh}[d]\text{Csch}[r]^2\right]\text{Cosh}[r]\right)$$

terrr[r_, d_]:=

$$2\left(-\pi+\text{ArcCos}\left[\text{Coth}[r]\text{Tanh}\left[\frac{d}{2}\right]\right]+\text{ArcCos}\left[\text{Coth}[r]\text{Tanh}\left[\frac{d}{2}\right]\right]\text{Cosh}[d]+\right.$$

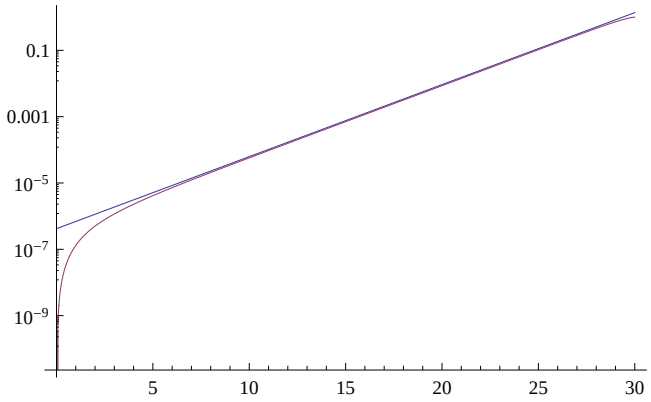
$$\left.\text{ArcCos}\left[\text{Coth}[r]^2-\text{Cosh}[d]\text{Csch}[r]^2\right]\text{Cosh}[r]\right)$$

LogPlot[terrr[15., d], {d, 0, 15}]

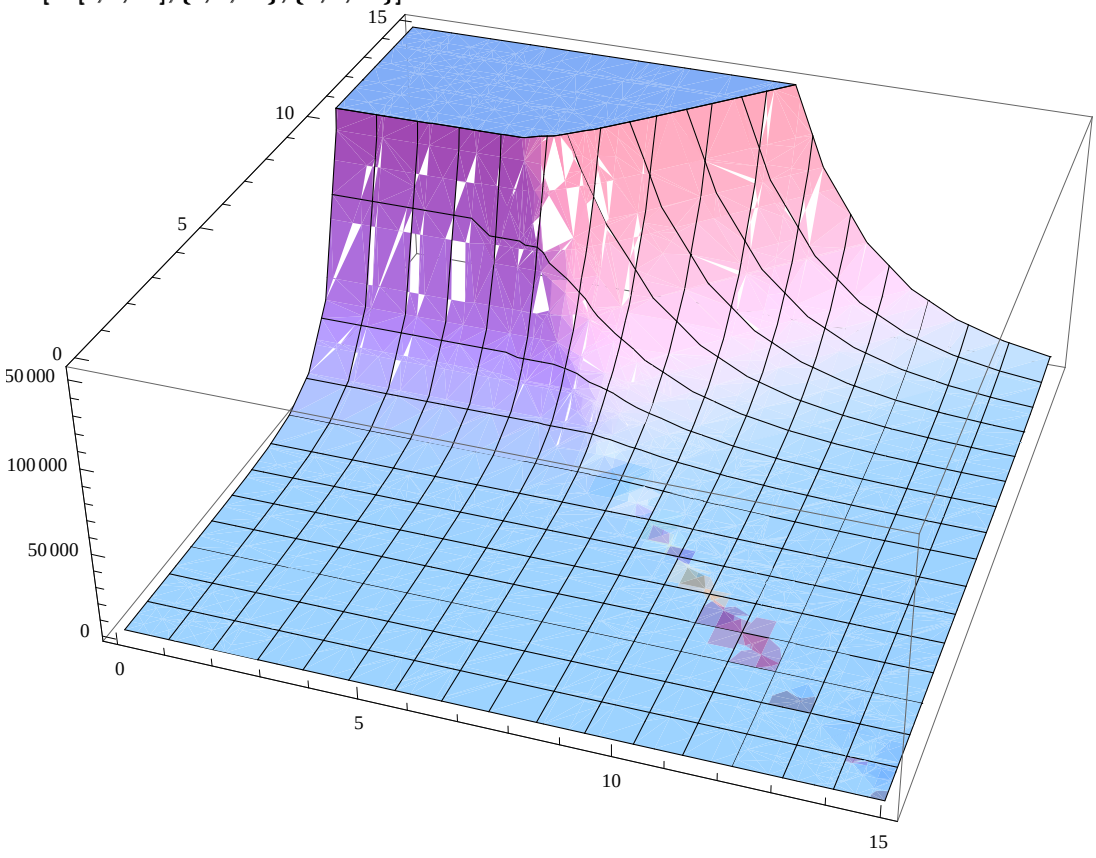


LogPlot[{4.3Exp[1/2d]/(2π(Cosh[15.] − 1)), terrr[15., d]/(2π(Cosh[15.] − 1))},

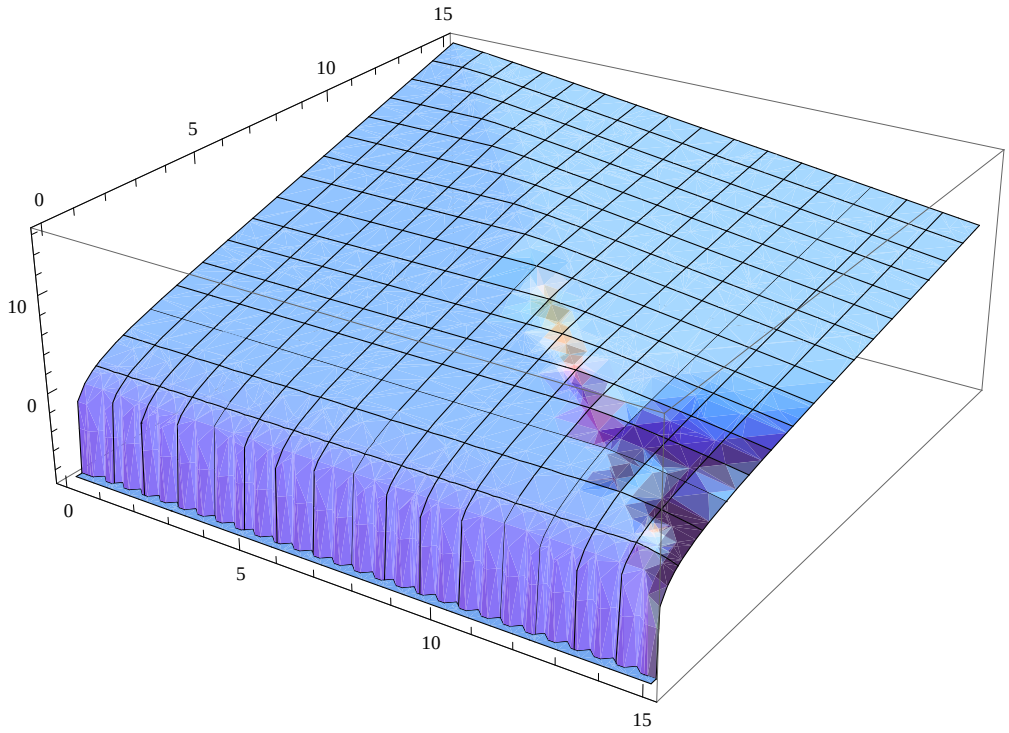
{d, 0, 30}]



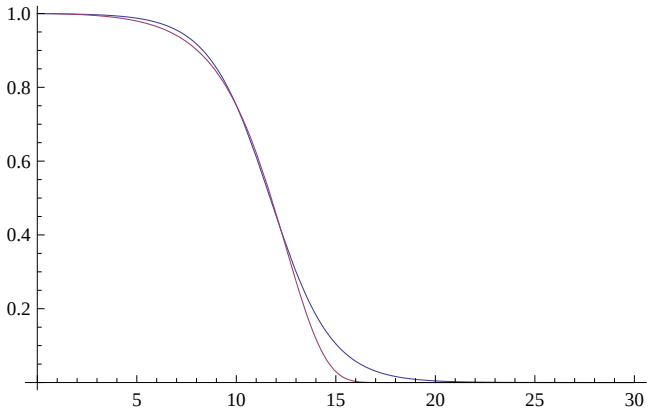
Plot3D[ter[r, d, 15], {r, 0, 15}, {d, 0, 15}]



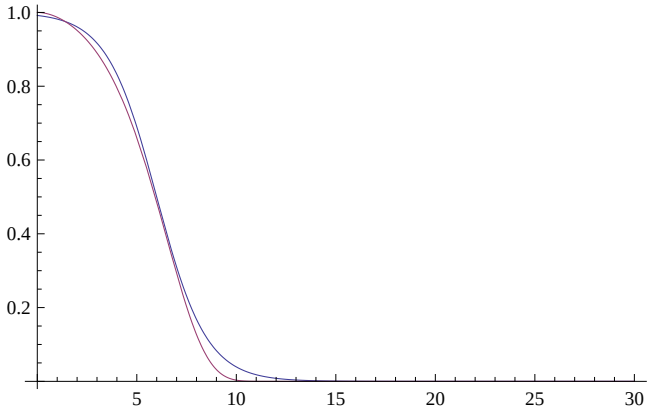
Plot3D[Log[ter[r, d, 15]], {r, 0, 15}, {d, 0, 15}]



$$\text{Plot}\left[\left\{\frac{1}{1+\text{Exp}[.65(d-11.7)]},(1-\text{terrr}[15.,d]/(2\pi(\text{Cosh}[15.]-1)))^{5000}\right\},\{d,0,30\}\right]$$



$$\text{Plot}\left[\left\{\frac{1}{1+\text{Exp}[.8(d-6)]},(1-\text{terrr}[15.,d]/(2\pi(\text{Cosh}[15.]-1)))^{100000}\right\},\{d,0,30\}\right]$$




```
{#, LogPlot [(1 - ter[#, d, 15]/(2π(Cosh[15.] - 1)))^5000, {d, 10, 15.}]} &/@
Range[0, 10, 2]
```

$$\{\{0,\},\{2,\},\{4,\},\{6,\},\{8,\},\{10,\}\}$$

```
{#, Plot [(1 - ter[#, d, 15]/(2π(Cosh[15.] - 1)))^5000, {d, 0, 15.}]} &/@{0, 0.1, 0.01}
```

$$\{\{0,\},\{0.1,\},\{0.01,\}\}$$

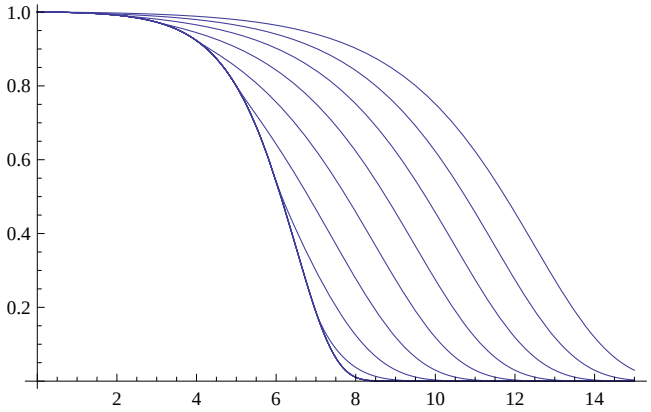
```
{#, Plot [(1 - ter[#, d, 15]/(2π(Cosh[15.] - 1)))^5000, {d, 0, 15.}]} &/@Range[1, 15]
```

$$\{\{1,\},\{2,\},\{3,\},\{4,\},\{5,\},\{6,\},\{7,\},\{8,\},\{9,\},\{10,\},\{11,\},\{12,\},\{13,\},\{14,\},\{15,\}\}$$

```
ReplaceAll[
"[%204[[1]][[2]],%204[[2]][[2]],%204[[3]][[2]],%204[[4]][[2]],%204[[5]][[2]],%204[[6]][[2]
],%204[[7]][[2]],%204[[8]][[2]],%204[[9]][[2]],%204[[10]][[2]],%204[[11]][[2]],%204[[12
]][[2]],%204[[13]][[2]],%204[[14]][[2]],%204[[15]][[2]]]", "204" → "204"]
```

$$[\%195[[1]][[2]],\%195[[2]][[2]],\%195[[3]][[2]],\%195[[4]][[2]],\%195[[5]][[2]],\%195[[6]][[2]],\%195[[7]][[2]],\%195[[8]][[2]],\%195$$

```
Show[%204[[1]][[2]], %204[[2]][[2]], %204[[3]][[2]], %204[[4]][[2]], %204[[5]][[2]],
%204[[6]][[2]], %204[[7]][[2]], %204[[8]][[2]], %204[[9]][[2]], %204[[10]][[2]],
%204[[11]][[2]], %204[[12]][[2]], %204[[13]][[2]], %204[[14]][[2]], %204[[15]][[2]]]
```



References

[1] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *ACM Conext*, 2009.

[2] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Comm. Review*, vol. 37, no. 4, pp. 181–192, 2007.

[3] D. Trossen, "Conceptual architecture: Principles, patterns and sub-component descriptions," in *Technical Report available at http://www.fp7-pursuit.eu/PursuitWeb*, D. Trossen, Ed., 2011.

[4] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: Line speed publish/subscribe inter-networking," *ACM SIGCOMM CompComm Review*, vol. 39, no. 4, pp. 195–206, 2009.

[5] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, "Revisiting IP multicast," *ACM SIGCOMM Computer Comm. Review*, vol. 36, no. 4, p. 26, 2006.

[6] M. Särelä, C. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott, "Forwarding anomalies in bloom filter-based multicast." in *IEEE INFOCOM*, Shanghai, China, 2011, pp. 2399–2407.

[7] C. Rothenberg, C. Macapuna, F. Verdi, M. Magalhães, and A. Zahemszky, "Data center networking with in-packet bloom filters," in *Brazilian Symposium on Computer Networks (SBRC)*, Gramado, Brazil, 2010.

[8] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended lans," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, no. 2, pp. 85–110, 1990.

[9] S. Deering, C. Partridge, and D. Waitzman, "Distance vector multicast routing protocol," *Internet Request For Comments RFC-1075*, 1988.

[10] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)," RFC 4601 (Proposed Standard), Internet Engineering Task Force, Aug. 2006, updated by RFCs 5059, 5796.

[11] M. Bag-Mohammadi and N. Yazdani, "A fast and efficient explicit multicast routing protocol," *IEICE Trans. on Communications*, vol. E-88B, no. 10, pp. 4000–4007, 2005.

[12] J. Bion, D. Farinacci, M. Shand, and A. Tweedly, "Explicit route multicast (ERM)," Internet Engineering Task Force, June 2000.

[13] M. Bag-Mohammadi, S. Samadian-Barzoki, and N. Yazdani, "Linkcast: Fast and scalable multicast routing protocol," in *IFIP Networking*, 2004.

[14] B. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[15] P. Elias, “Universal codeword sets and representations of the integers,” *IEEE Transactions on Inf. Theory*, vol. 21, no. 2, pp. 194–203, 1975.

[16] A. Kirsch and M. Mitzenmacher, “Less hashing, same performance: Building a better bloom filter,” in *Annual European Symposium on Algorithms (ESA)*, 2005.

[17] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, and Y. Tang, “On the false-positive rate of Bloom filters,” *Information Processing Letters*, vol. 108, no. 4, pp. 210–213, 2008.

[18] SNDlib, “Survivable fixed telecommunication network design library,” http://sndlib.zib.de.

[19] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The Internet Topology Zoo,” http://www.topology-zoo.org.

[20] J. M. Jaffe, “Bottleneck flow control,” *IEEE Transactions on Communications*, vol. 29, no. 7, pp. 954–962, Jul. 1981.

[21] E. L. Hahne, “Round-robin scheduling for max-min fairness in data networks,” *IEEE Journal on Selected Areas of Communication*, vol. 9, no. 7, pp. 1024–1039, Sep. 1991.

[22] A. F. T. Committee, “Traffic Management Specification - Version 4.0,” ATM Forum/95-0013R13, Feb 1996.

[23] D. P. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[24] J. Le Boudec, “Rate adaptation, congestion control and fairness: a tutorial,” available online: http://ica1www.epfl.ch/PS_files/LEB3132.pdf, Feb 2005.

[25] R. Denda, “The fairness challenge in computer networks,” Department for Mathematics and Computer Science, University of Mannheim, Tech. Rep. TR-00-006, 2000.

[26] J. Y. Le Boudec and B. Radunovic, “A unified framework for max-min and min-max fairness with applications,” in *Proceedings of 40th Annual Allerton Conference on Communication, Control, and Computing*, Oct 2002.

[27] Z. Cao and E. W. Zegura, “Utility max-min: an application-oriented bandwidth allocation scheme,” in *Proceedings of INFOCOM 1999*, vol. 2, March 1999, pp. 793–801.

[28] J. Ros and W. K. Tsai, “A theory of convergence order of maxmin rate allocation and an optimal protocol,” in *Proceedings of INFOCOM 2001*, vol. 2, Apr 2001, pp. 717–726.

[29] J. Kleinberg, Y. Rabani, and É. Tardos, “Fairness in routing and load balancing,” *Journal of Computer and System Sciences*, vol. 63, no. 1, pp. 2–20, 2001.

[30] D. Nace and L. Doan, “A polynomial approach to the fair multi-flow problem,” Tech. Rep., Heudiasyc, UTC, available online: http://www.hds.utc.fr/~dnace/recherche/Publication/TR-MMF.pdf, 2002.

[31] G. Ziegler, *Lectures on Polytopes*, ser. Graduate Texts in Mathematics. New York: Springer, 1998, vol. 152.

[32] B. Grünbaum, *Convex Polytopes*. Interscience Publishers John Wiley & Sons, Inc., New York, 1967.

[33] M. Iri, “On an extension of the maximum-flow minimum-cut theorem to multicommodity flows,” *Journal of the Operations Research Society of Japan*, vol. 13, no. 3, pp. 129–135, 1971.

[34] K. Onaga and O. Kakusho, “On feasibility conditions of multicommodity flows in networks,” *IEEE Transactions on Circuit Theory*, vol. 18, no. 4, pp. 425–429, 1971.

[35] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[36] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, 2nd ed. New York: John Wiley & Sons, 1990.

[37] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd ed. New York: John Wiley & Sons, 1993.

[38] K. Fukuda and A. Prodon, “Double description method revisited,” in *Combinatorics and Computer Science, 8th Franco-Japanese and 4th Franco-Chinese Conference, Brest, France, July 3-5, 1995, Selected Papers*, ser. Lecture Notes in Computer Science, M. Deza, R. Euler, and Y. Manoussakis, Eds., vol. 1120. Springer-Verlag, Berlin, 1996, pp. 91–111.

[39] M. L. Garcia-Osma, “TID scenarios for advanced resilience,” Tech. Rep., The NOBEL Project, Work Package 2, Activity A.2.1, Advanced Resilience Study Group, Sep 2005.

[40] B. Chinoy and H. W. Braun, “The national science foundation network,” Tech. Rep., CAIDA, available online: http://www.caida.org/outreach/papers/1992/nsfn/nsfnet-t1-technology.pdf, Sep 1992.