On the Number of MPLS LSPs using Multicast Tree Aggregation

Joanna Moulierac*, Alexandre Guitton† and Miklós Molnár*
* INRIA, University of Rennes I, France
† Birkbeck College, University of London, United Kingdom
Phone number: +33 2.99.84.71.94
Email: joanna.moulierac@irisa.fr, alexandre@dcs.bbk.ac.uk and miklos.molnar@irisa.fr

***Abstract*— Multicast tree aggregation is an efficient proposition that can solve the multicast forwarding state scalability problem. Existing works on tree aggregation have focused on developing and simulating protocols that build trees dynamically. However, the underlying problem of the impact of the tree construction algorithm on the performance of the protocols remains untouched. In this paper, we propose a study on the number of trees that need to be configured in a domain depending on the tree construction algorithm. We ran extensive simulations on several real domains and with different tree construction algorithms. Our results show that for a given set of multicast groups, even when this set includes all the possible groups, the number of trees that need to be configured is small. This allows a network administrator to configure off-line all these trees in order to maintain a stable set of trees and to have knowledge of the routes used by the multicast packets. Knowing the set of all the possible trees is also useful to determine the best subset to configure and to give an upper bound of the number of different trees.**

***Keywords*— Multicasting, tree aggregation, MPLS, control and management of network.**

I. Introduction

The research in multicast has highly interested the researchers during the last decade but unfortunately, multicast is not yet deployed over Internet. However, as the number of multimedia applications including video on demand, television on Internet and on line video games increases tremendously, multicast is becoming a necessity for network operators to spare the network resources.

The multicast tree aggregation is an efficient proposition that allows the deployment of group communications. The main goal of tree aggregation is to keep only a small number of multicast forwarding states in the routers in order to spare the memory of the routers, to speed up IP look-ups and to reduce the amount of control messages needed to maintain the forwarding states. Indeed, in multicast, aggregation of routing entries is more difficult than in unicast as the multicast IP addresses do not reflect the location of the members.

To reduce the number of entries, the multicast tree aggregation technique allows several multicast groups to share the same delivery tree. In other words, the packets for the groups can be multiplexed onto the same tree upon entering the domain. This can be achieved using IP encapsulation or MPLS labels. In this case, the MPLS labels can be distributed among the routers using LDP.

Because of these two different ways of deploying multicast tree aggregation, we will talk in this paper indifferently of trees or of MPLS LSPs.

Existing works on tree aggregation have focused on developing and simulating protocols that build trees dynamically. In this paper, our key concern is to study the impact of different tree construction algorithms on the performance of the protocol, which is still a research issue. The performance of a tree aggregation protocol is defined by the number of trees to be maintained and by the resource usage for each group. We first show that the way trees are built leads to major scalability issues. Then, we study several commonly used tree construction algorithms, along with the number of trees to be maintained in order to cover a set of multicast groups. We show with our simulations that it is possible to build a set of trees covering all the possible multicast groups in a given domain as this number is usually very small and few forwarding entries are needed in routers. This allows a network administrator to configure this stable set of trees in order (i) to plan the routing for the future multicast groups, (ii) to be aware of the routes used by the multicast packets and (iii) to maintain a stable set of forwarding entries. Moreover, minimizing the number of MPLS LSPs in a domain is of paramount importance for scalability issues.

The rest of the paper is organized as follows. Section II presents the main protocols achieving tree aggregation. In Section III, we present the main idea of our proposition and give an algorithm that computes the number of MPLS LSPs required, *i.e.*, the number of multicast trees. The results of the simulations are detailed in Section IV. Finally, the perspectives of our work are given in Section V.

II. Multicast tree aggregation

Tree aggregation reduces the number of multicast forwarding states and the tree maintenance overhead. To achieve this reduction, several multicast groups share the same delivery tree within a domain. Consequently, less trees are built and less forwarding entries are stored than with traditional multicast.

A. Aggregation protocols

The AM protocol (Aggregated Multicast) was proposed in [1], [2]. In AM, a centralized entity called the tree manager

is in charge of matching groups to labels and informing the border routers of this matching.

Several recent protocols have addressed the scalability problems of AM. For instance, STA [3] performs faster aggregations than AM by sorting the trees in the multicast tree set, addressing the time-scalability issue. In BEAM [4], the task of the tree manager is distributed among several routers, called core routers. The protocol AMBTS [5] splits the native tree in sub-trees before aggregating, while the protocol TALD [6] performs tree aggregation in large domains by splitting the network in several sub-domains and by managing the sub-domains independently. AMBTS and TALD can be applied in large domains, where other approaches fail.

In parallel, new protocols have been developed with additional constraints. AQoSM [7] and Q-STA [8] perform tree aggregation under bandwidth constraints. In these two protocols, the links have limited bandwidth capacities and groups have bandwidth requirements depending on the multimedia application. The protocol TOMA [9] deals with overlay multicast. Finally, the protocol AMFM [10] proposes a fault tolerant architecture.

In all these protocols, the tree construction algorithm is either unspecified or arbitrary. We believe that understanding how the tree construction algorithm impacts tree aggregation protocols can lead to further enhancement of the existing mechanisms or to protocols based on new concepts.

B. On the number of trees

Our work is based on the frameworks proposed in [11] and in [12]. Among the research papers on tree aggregation, the closest to our work is [13] which also considers the number of trees in a domain. However, our work tackles the problem from another angle. First, we consider that the branching routers of the trees can lead to multicast members. Indeed, in real networks the non-leaf routers used by a tree can be multicast capable and can be attached to members. Second, we propose an analysis of the upper-bound of the number of trees needed to span the set of all multicast groups considering different tree algorithm construction. Some scalability problems arise with this approach when the number of routers in the network is large, as the set of all the multicast groups cannot be computed. However these problems can be solved (although not optimally) by using a similar approach as TALD, *i.e.*, by splitting the domains into sub-domains and computing the set of trees in each sub-domain separately.

As we said earlier, the previous works do not focus on the impact of the tree construction algorithms on the number of MPLS LSPs or trees to be maintained. However, we believe that it is one of the most important parameter of tree aggregation protocols. That is why we are interested in answering the following questions:

- What is the impact of the tree construction algorithm on the number of different trees?
- How does the number of different trees evolves as a fraction of the bandwidth is allowed to be wasted?

III. Algorithm for configuring a set of trees

In this section, we present a naïve algorithm that computes a set of multicast trees for a given domain.

First of all, we will say in the following that a router is a member of a group if this router leads to group members. The real members of the group can be directly attached to the router or can be connected to it through several domains. We define a group by the set of the routers leading to members rather than by the set of the members themselves. With this definition, two groups can be identical even if they do not have the same multicast address.

Second, since the main goal of the tree aggregation is to reduce the number of trees, we will focus on shared tree. Such trees are usually built by tree aggregation protocols. However, our approach still holds when considering source-based trees. It can be noticed that for source-based trees, more trees are configured as all (source,group) pairs are considered whereas in the case of shared trees, only the groups are considered. Shared trees are built with CBT [14] or with PIM-SM [15]. The trees are rooted at a core router (or at a *rendez-vous* point) and cover the members of the groups. As the determination of the core router is still a research issue in the literature, we will consider in this paper three ways to choose it. This is described later in Section IV. Shared trees can also be built with the objective of minimizing the cost: in this case, approximate Steiner trees are used [16]. We will include such trees in our comparison. Even if they are not used in real networks, they provide us with insight of the lower bound of the bandwidth used for a set of groups.

A. On the number of trees compared to the number of groups

Previous works on tree aggregation were based on the following notice: two distinct multicast groups can be identical when projected on a domain. In other words, if we consider the routers that have members attached to them rather than the members themselves, it is easy to see that distinct members can be attached to the same edge routers. If n denotes the number of routers of the domain, the upper bound on the number of trees is only 2^n, instead of the whole space of multicast addresses.

However, we can go a step further. Let us consider the example shown on Fig. 1. A tree rooted at A and covering the routers $\{A, B, C, D, E, F\}$ is represented on the figure. This tree is the native tree for four different groups. Indeed, it can cover the groups $\{A, C, D, F\}$, $\{A, C, D, E, F\}$, $\{A, B, C, D, F\}$ and $\{A, B, C, D, E, F\}$. All these aggregations can be done without bandwidth wasted as the tree traverses the two branching routers B and E anyway in order to reach the leaves C and F.

Therefore, by noticing that packets for the groups traverse several routers that can be multicast capable before reaching the leaves, we see that the number of different trees is lower than the previous bound 2^n.

Note that in the previous description the trees aggregated together were identical, *i.e.*, no bandwidth is wasted. To further reduce the number of different trees in a domain, it is realistic to consider that the network administrators allow a given percentage of the multicast bandwidth to be wasted in order

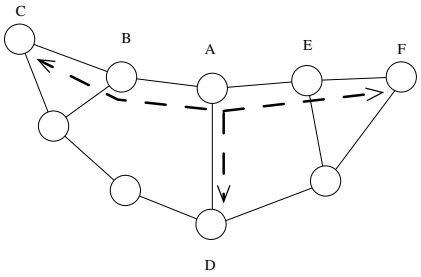

Fig. 1. The tree represented on the figure covers several multicast groups.

to ease the aggregation process: larger trees will be chosen to cover a group, but less such trees will be built. Let us consider once more the example shown on Fig. 1. A group $g = \{B, F, D\}$ can share the same tree as the one depicted, but some bandwidth will be wasted when packets for the group g reach router C unnecessarily. If the bandwidth used by the group g is small compared to the bandwidth available on the link (B, C), the advantage of having only one tree instead of two might prevail (remember that each tree requires control messages to be maintained).

B. Description of the Algorithm

Algorithm 1 computes the set $\mathcal{T}$ of different trees given the bandwidth bound bw. No more than bw percent of bandwidth is allowed to be wasted. Every possible group is considered in turn. For each group $g_i \in \mathcal{G}$, the algorithm computes a native tree t_i. If there is already a tree in $\mathcal{T}$ that covers the same edge routers as t_i and that does not waste too much bandwidth, then there is no need to add t_i in the set $\mathcal{T}$.

Algorithm 1 Computing the set of different trees.

Input: a domain with n routers, a tree construction algorithm $\mathcal{A}$, a bandwidth threshold bw
Output: a set of trees $\mathcal{T}$ covering all the 2^n groups

```
T ← ∅
for i = 1 to 2^n do
    g_i is the i-th group of the 2^n possible groups of G
    compute a native tree t_i covering g_i with algorithm A
    if there is a tree t ∈ T that covers at least the routers of
    t_i and if cost(t) ≤ (cost(t_i) + cost(t_i) * bw) then
        ignore t_i
    else
        add t_i to T
    end if
end for
return T
```

The algorithm complexity is $\mathcal{O}(2^{n+1})$, which scales badly with the number of edge routers of the domain. However, previous works [6] have shown that a large domain can be separated into several sub-domains having few edge routers, in such a way that the tree aggregation can be done separately in each sub-domain without suffering a significant drop of the performance. Therefore, if the domain is too large, it can be divided into several sub-domains, and then the algorithm presented above can be applied in each of the sub-domain in order to scale better.

Algorithm 1 computes the number of trees in a worst-case scenario where each group has an equal probability of being requested. Previous studies have shown that real groups show a high degree of correlation which can be approximated by affinity models. Our algorithm still applies for this scenario, resulting in a smaller number of trees to configure. Moreover, our algorithm can be applied for only a subset of multicast groups to be covered. This can be useful when a network administrator has traces of real multicast groups and wants to configure only a subset of trees covering these specific groups.

Algorithm 1 is a standard way to compute the total number of trees in a domain and has been described similarly in the past. However, the novelty of our paper is that the tree construction algorithm $\mathcal{A}$ used in Algorithm 1 is of major importance. In the following, we point out that $\mathcal{A}$ is a parameter that is even more important than the bandwidth threshold bw since it leads to a higher reduction of the total number of trees.

IV. Simulation results

We ran simulations of Algorithm 1 on different real graphs: the VSNL network [17] (which is given by an Internet mapping tool called Rocketfuel [18]), the Abilene network [19], the Nsfnet network [20] and the Geant network [21]. We ran the simulations for two bandwidth thresholds: when no bandwidth is wasted ($bw = 0\%$) and when up to 20% of bandwidth is allowed to be wasted. We compared the tree construction algorithms by varying the way the core (or the *rendez-vous* point) is chosen in CBT and PIM-SM and evaluating the number of trees generated. Recall that these protocols build shortest path trees rooted at a given core. We also consider minimum spanning trees. The four compared algorithms are:

- CBT-Fixed: a core router is initially chosen arbitrarily among the routers of the network and all the trees are rooted at this core.
- CBT-First: the member having the smallest lexicographical order is chosen as the core router.
- CBT-Rand: the trees are rooted at a core randomly chosen among the members.
- MST: a minimum spanning tree covering the routers that lead to members is built in the metrical closure graph. This tree is a 2-approximate Steiner tree.

A. Number of trees to be configured

Table I shows the total number of different groups and the number of trees needed to cover these groups. Note that for a domain with n multicast capable routers, there are $(2^n - n - 1)$ different groups. Indeed, we do not generated groups having one member or groups without members. The number of different trees is very low compared to the number of different groups. Therefore, it is possible to configure all the different trees in a domain.

When using CBT-Fixed, trees are very similar. Indeed, they are rooted at the same core and use frequently the same links

(since shortest paths trees are built). This means that CBT-Fixed builds few different trees. Oppositely, we can see that more trees are built with MST than with CBT-Fixed. Indeed, MST trees use less links than CBT-Fixed and the link usage is evenly distributed in the network. Thus, the different trees built by MST are completely different from each other and few aggregation can be found. However, with CBT-Fixed, the links around the core router experience an important stress, since they are shared by most of the trees.

With CBT-First, the trees frequently share the same core. Thus, there are few different cores and the link usage of trees with CBT-First is better distributed than with CBT-Fixed, but more trees are required. Finally, CBT-Rand chooses randomly the core for each group which implies that the trees are rooted at several different sources. The link usage is balanced but more trees are needed than with all others CBT algorithms. This shows that the choice of the core router has a great impact on the number of trees to be configured. With the MST algorithm, we can get the upper bound of the number of trees required when the metric considered is the network usage minimization.

On the impact of the topology: The topology of the network has also a great impact on the number of trees. Indeed, although the Abilene and the VSNL networks share the same number of nodes, the number of trees built in each of them is different. VSNL consists of fewer links and several routers have a degree of 1. Because of this small number of branching routers and depending on the tree construction algorithm, more trees are configured for VSNL than for Abilene. To explain this phenomenon, let us consider a network with n routers where the first $n-1$ routers are leaves and are all connected to the n-th router. This network is a star network. In this example, the number of trees is independent of the tree construction algorithm and is equal to $2^{n-1}-1$ (the number of permutations of the $n-1$ leaves minus the empty tree). In this example, the router linked to all the other routers has to keep $2^{n-1}-1$ forwarding entries. Another example to be considered is a path of n routers. In this network, there are 2 leaves and $n-2$ routers with a degree of 2: the number of different trees is $n(n-1)/2$, which is much smaller than on the previous example. These two examples show the impact of the network topology on the number of trees.

B. Number of forwarding entries per router

Figure 2 and Fig. 3 plot the number of forwarding entries per router for the Geant network and for Abilene. We compare the minimum, the maximum and the mean number of forwarding entries stored per router. The figures show that when bandwidth wastage is allowed ($bw = 20\%$), the number of forwarding entries is smaller than when $bw = 0\%$. The number of forwarding entries is strongly related to the number of trees. The impact of the tree construction algorithm for both metrics can be explained in the same way. What can be noticed is that the number of forwarding entries is small enough to be stored in routers.

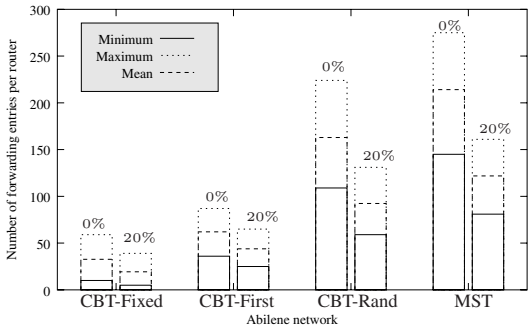

Fig. 2. Number of forwarding entries for Abilene network.

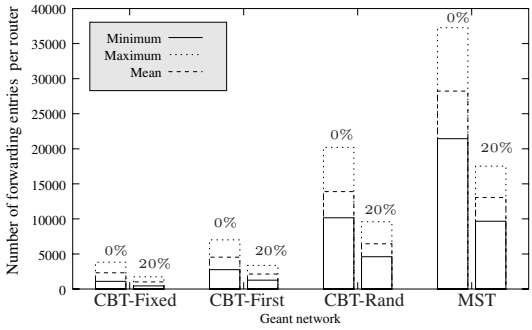

Fig. 3. Number of forwarding entries for Geant network.

C. Mean cost of trees per group

Figure 4 and Fig. 5 plot the mean cost of trees per group:

$$\frac{\sum_{g \in \mathcal{G}} \text{cost}(t(g))}{|\mathcal{G}|},$$

where $t(g)$ is the tree of minimum cost in $\mathcal{T}$ that covers g, $\text{cost}(t(g))$ is the cost of the tree $t(g)$ in terms of number of links and $|\mathcal{G}|$ is the number of different groups. This metric reflects the number of links loaded and gives an overview of the bandwidth used per group. These two figures show the compromise between the number of trees and the usage of the network resources. Indeed, the less trees to be maintained the higher the cost of trees per group resulting in loaded links. For the Geant network which has more than 260,000 different groups, a network administrator can configure less than 2,000 trees with CBT-Fixed and with $bw = 20\%$, but some bandwidth is wasted for each group (the trees use on average 12 links for each group). For a minimum bandwidth usage, MST with $bw = 0\%$ configures 75,000 trees but the trees only use an average of 9 links per group. It is interesting to notice that building MST trees with $bw = 20\%$ leads to approximately the same results as building CBT-Rand trees with $bw = 0\%$ for the Geant network.

V. Conclusion and Future Work

In this paper, we proposed a study of the impact of tree construction on the number of trees to be maintained in a domain. During the simulations, we generated all the possible groups for a given domain and we computed the set of trees needed to cover all these groups while varying the tree construction algorithms. Our study showed that very few trees

	bw	Number of $\neq$ groups $\lvert\mathcal{G}\rvert$	CBT-Fixed	CBT-First	CBT-Rand	MST
VSNL Network [17]	0%	2,036	173	239	279	293
(11 routers, 12 links)	20%	2,036	103	156	179	191
Abilene Network [19]	0%	2,036	59	131	292	370
(11 routers, 14 links)	20%	2,036	39	104	183	230
Nsfnet Network [20]	0%	16,369	395	958	3,299	4,785
(14 routers, 21 links)	20%	16,369	210	549	1,740	2,486
Geant Network [21]	0%	262,125	3,824	8,222	24,542	48,942
(18 routers, 30 links)	20%	262,125	1,748	4,126	11,924	23,449

TABLE I

TOTAL NUMBER OF GROUPS AND TOTAL NUMBER OF TREES

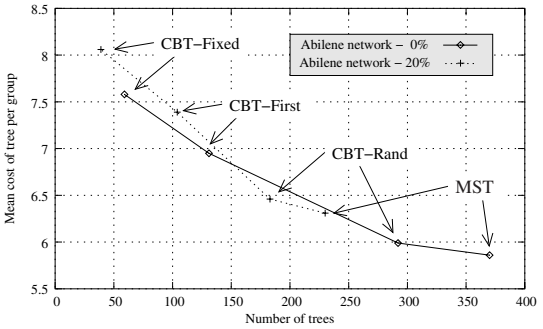

Fig. 4. Mean cost of trees per group for Abilene network [19].

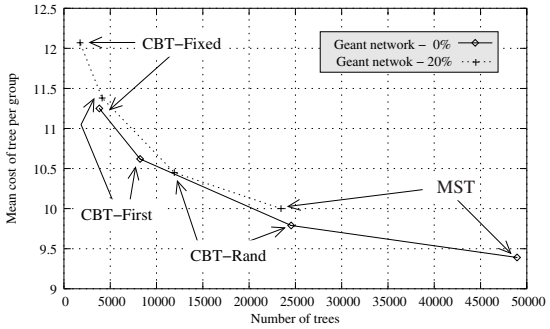

Fig. 5. Mean cost of trees per group for Geant network [21].

are needed to be configured in a domain and that it is possible for a network administrator to compute this stable set of trees off-line. This allows to manage more efficiently the domain and to be aware of the configured trees. We also studied the compromise between the number of trees configured and the mean cost of trees per group. When the network administrator has traces of real multicast groups, only a sub-set of all these trees can be computed. We proposed an algorithm that gives an upper bound on the number of trees needed depending on the tree construction algorithm. When these trees are configured, usual group-tree matching protocols can be used without modification. The number of trees is also dependent on the topology of the network.

Although most tree aggregation protocols focus on varying the bandwidth threshold, we have shown in this paper that they should rather focus on varying the tree construction algorithm, since it can lead to fewer trees. This is the first step towards hybrid protocols that deal simultaneously with the bandwidth and the tree construction algorithm.

REFERENCES

[1] M. Gerla, A. Fei, J.-H. Cui, and M. Faloutsos, "Aggregated Multicast for Scalable QoS Multicast Provisioning," in *Tyrrhenian International Workshop on Digital Communications*, September 2001.

[2] J.-H. Cui, J. Kim, D. Maggiorini, K. Boussetta, and M. Gerla, "Aggregated multicast — a comparative study," in *IFIP Networking*, ser. LNCS, no. 2345, May 2002, pp. 1032–1044.

[3] A. Guitton and J. Moulierac, "Scalable Tree Aggregation for Multicast," in *8th International Conference on Telecommunications (ConTEL)*, June 2005, best student paper award.

[4] J.-H. Cui, L. Lao, D. Maggiorini, and M. Gerla, "BEAM: A Distributed Aggregated Multicast Protocol Using Bi-directional Trees," in *IEEE International Conference on Communications (ICC)*, May 2003.

[5] Z.-F. Liu, W.-H. Dou, and Y.-J. Liu, "AMBTS: A Scheme of Aggregated Multicast Based on Tree Splitting," in *IFIP Networking*, ser. LNCS, no. 3042, May 2004, pp. 829–840.

[6] J. Moulierac, A. Guitton, and M. Molnár, "Multicast Tree Aggregation in Large Domains," in *IFIP Networking*, 2006.

[7] J.-H. Cui, J. Kim, A. Fei, M. Faloutsos, and M. Gerla, "Scalable QoS Multicast Provisioning in Diff-Serv-Supported MPLS Networks," in *IEEE Globecom*, November 2002.

[8] J. Moulierac and A. Guitton, "QoS Scalable Tree Aggregation," in *IFIP Networking*, ser. LNCS, no. 3462, May 2005, pp. 1405–1408.

[9] L. Lao, J.-H. Cui, and M. Gerla, "TOMA: A Viable Solution for Large-Scale Multicast Service Support," in *IFIP Networking*, ser. LNCS, no. 3462, May 2005.

[10] J.-H. Cui, M. Faloutsos, and M. Gerla, "An Architecture for Scalable, Efficient and Fast Fault-Tolerant Multicast Provisioning," *IEEE Network special issue on Protection, Restoration, and Disaster Recovery*, vol. 18, no. 2, pp. 26–34, March/April 2004.

[11] J. Moulierac, "On the number of multicast aggregated trees in a domain," in *2nd Student Workshop of IEEE Infocom*, April 2006.

[12] J. Moulierac and A. Guitton, "Distributed Multicast Tree Aggregation," INRIA, Research Report 5636, July 2005.

[13] L. Lao, J.-H. Cui, and M. Gerla, "Tackling Group-Tree Matching in Large Scale Group Communications," UCLA CSD, Technical Report 040022, June 2004.

[14] A. Ballardie, "Core Based Trees (CBT) Multicast Routing Architecture," IETF, RFC 2201, September 1997.

[15] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. E. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," IETF, RFC 2362, June 1998.

[16] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees in graphs," *Acta Informatica*, vol. 15, pp. 141–145, 1981.

[17] VSNL network, "http://www.cs.washington.edu/research/networking/-rocketfuel/interactive/."

[18] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *ACM SIGCOMM*, August 2002.

[19] Abilene network, "http://abilene.internet2.edu."

[20] Nsfnet network, "http://www.cybergeography.org/atlas/nsfnet_t1.gif."

[21] Geant network, "http://www.cybergeography.org/atlas/geant_large.gif."