Annales Univ. Sci. Budapest., Sect. Comp. 32 (2010) 231-245

CLUSTERING–LEARNING MODEL FOR REDUCING THE DELAY IN TRAFFIC GROOMING OPTIMIZATION

Vo Viet Minh Nhat and Le Manh Thanh
(Hue, Vietnam)

Abstract. Hopfield networks have been suggested as a tool for the optimization of traffic grooming. However, the optimization based on neural networks normally requires a considerable delay to find an optimal solution, which has limited practicability in optical data transport. This paper proposes a solution to reduce this delay. That is a clustering-learning model which (1) eliminates the arriving serving patterns that do not need optimizing its grooming, (2) removes the services which do not participate in the grooming optimization in an arriving service pattern, and (3) clusters arriving service patterns into the groups in which the service patterns belonging to each group have the same optimal grooming solution. For the last case, the learning function of this model checks if an arriving service pattern is similar to another optimized before: if one is found, the corresponding optimal grooming solution is returned immediately. If not, an optimization process is required to determine its optimal grooming solution. This solution is *"memorized"* and reused for next arriving service patterns.

1. Introduction

Traffic grooming has attracted the attention of researchers in the area of optical data transport with the aim of exploiting at maximum the potential bandwidth capacity of optical fibers. It can be considered under different approaches, such as the minimization of the number of used wavelengths, equipped adding/dropping multiplexers, or/and required wavelength converters [7, 6, 3, 5] which are formulated as integer linear functions and then minimized using heuristic algorithms. In [8], traffic grooming is an optimization of the

service-into-burst multiplexing and the service-between-burst switching, which is presented in the form of a Hopfield energy function and optimized by the principle *"the minimal energy, the optimal solution"* [2]. A limit of the optimization based on neural networks normally is to require a considerable delay to find an optimal solution, which is hard to be used in optical data transport. However, a characteristic of these methods is that the optimization process is done gradually basing on the principle *"the longer the optimization cycle is, the better the returned solution is"*. So, if the limited delay is short, a nearly-optimal solution can be returned for the current traffic grooming. As shown in Figure 1, the Hopfield energy function never increases with time. Assuming that the nearly-optimal solutions can be used from the time t_k, we can then stop the current optimization process at any time t_c ($t_c \geq t_k$) and the corresponding returned solution is nearly-optimal. Of course, if $t_c < t_k$, the optimization methods based on Hopfield networks cannot be used.

In [8], we have proposed solutions to reduce the delay of traffic grooming optimization based on Hopfield networks. That is a determination in advance mathematically of the connection weights and the activation thresholds of the Hopfield networks used for optimization.

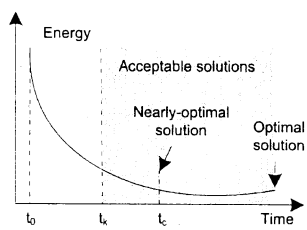

Figure 1. A nearly-optimal grooming solution can be returned if the permitted delay of traffic grooming is limited ($t_c \geq t_k$)

Moreover, the chosen constraint weights and the improvements in algorithms also help to minimize this delay. However, based on the simulation results in [8], the traffic grooming optimization based on Hopfield networks is only practical for the service patterns of small dimension (Figure 2). When this dimension increases, the delay becomes too long to be practicable. This paper proposes a more complete solution to reduce the delay of the traffic grooming optimization based on Hopfield networks. That is a clustering-learning model

which (1) eliminates the arriving service patterns that do not need to be optimized its grooming, (2) removes the services which do not participate in the grooming optimization in an arriving service pattern, and (3) clusters arriving service patterns into groups in which the service patterns belonging to each group have the same optimal grooming solution. For the last case, the learning function of this model checks if an arriving service pattern is *"similar"* to another optimized before: if one is found, the corresponding optimal grooming solution is returned immediately. If not, an optimization process is required to determine its optimal grooming solution. This solution is *"memorized"* and reused for next arriving service pattern.

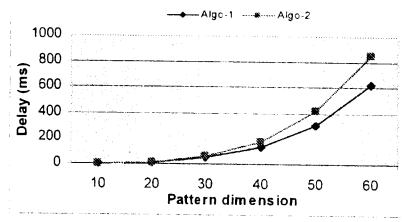

Figure 2. The optimization delay increases when increasing the dimension of service pattern

2. Overview of the traffic grooming optimization based on Hopfield networks

In [8], the traffic grooming is considered as multiplexing of arriving services into bursts at edge nodes and as switching services between bursts at core nodes. The bursts used to carry services are supposed to divide into timeslots. Arriving services (considered as a pattern of N services) are then represented by a two-dimension binary matrix (Figure 3a), where each line corresponds to a service and the number of columns is equal to the number of timeslots in each burst. On each line of the matrix a service is expressed by a chain of successive cells of value 1. The position of a service on a burst is important, e.g. for determining its destination. With this representation the optimization of multiplexing a service pattern into bursts is considered as an arrangement of these services into bursts so that the number of bursts used is minimal. Figure

3b illustrates an example of multiplexing a 4-services pattern into 2-bursts pattern.

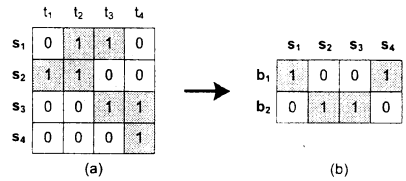

Figure 3. Example of multiplexing a service pattern into bursts

When the burst pattern arrives at a core node, the services carried on this burst pattern can be switched to another burst pattern which requires the exchanges of their bursts (i.e. wavelength conversion) or both the exchanges of their timeslots and bursts. Since the devices required for these switches are costly, the optimization of switching services between bursts is considered as a minimization of the switching cost. Figure 4 illustrates an example for all

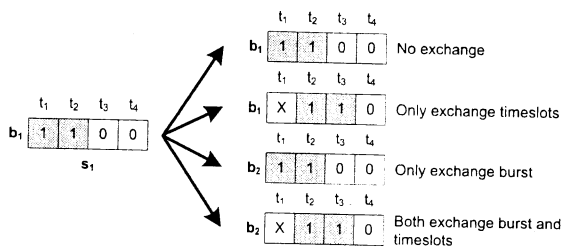

Figure 4. Example of switching a service between bursts

possible cases of switching services between bursts. By using the Hopfield networks as an optimization tool, these two problems of traffic grooming are formulated under objective functions and concerned constraints. The penalty function approach [4] is then used to transform them to a Hopfield energy function. From there we can determine the Hopfield networks with all connection weights and the activation threshold of each neuron.

2.1. Problematic

As presented in [9, 8], the delay problem of the optimization method based on Hopfield networks has been resolved. However, there exist in fact two other factors which influence the average delay of this optimization. That is the number of patterns which needs optimizing and the size of optimized service patterns. If we can reduce these two factors, more delay can be minimized.

Indeed, there are some arriving N-services patterns which always require N-bursts patterns for their multiplexing. That is the case in which each service in an arriving N-services pattern requires a complete burst for its multiplexing. So, the optimization for its multiplexing is unneeded.

In the case of an arriving N-services pattern multiplexed into a K-bursts pattern ($N > K$), an optimization for it is required because some services can be multiplexed into a same burst. However, there also the exist services which cannot be multiplexed with any other. There services are then removed to reduce the size of the service pattern optimized. It means that the delay of its optimization is also minimized.

In another case, there exists arriving different service patterns which are a same optimal grooming solution, called *"similar"* service patterns. These service patterns are then clustered into a group corresponding to the optimal grooming solution. When a new service pattern arrives, it is checked if it belongs to any existing optimized group: if one is found, the corresponding optimal grooming solution is returned immediately. If not, an optimization process is required to determine its optimal grooming solution. That is the principle of the learning function in our clustering-learning model. The following describes in detail the clustering-learning model.

3. The clustering-learning model for reducing the delay in traffic grooming optimization

3.1. Sorting the services in each service pattern

Let us consider arriving N-services patterns, each service of which is represented on T-timeslots. The number of possible representations of these N-services patterns is

$$n = (C_1^T)^N,$$

where C_1^T is the number of possible representations of a service on a burst. However, a N-services pattern has in fact $N!$ permutations of its representation. As shown in Figure 5, a 4-services pattern has $4! = 24$ permutations. If all these permutations are sorted in ascending or descending order, they have only one representation, as shown in Figure 5a and 5a'. In other words, the real number of possible representations of the N-services patterns is

$$n = \frac{(C_1^T)}{N!}.$$

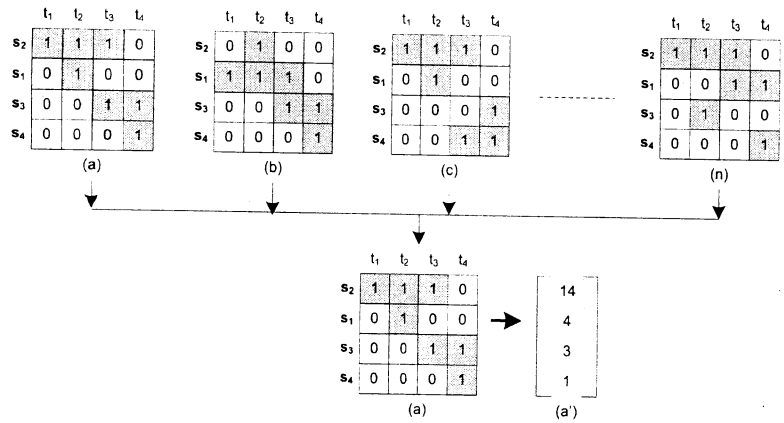

Figure 5. All possible permutations of a 4-services pattern

3.2. Eliminating the service patterns that do not need optimizing

We can recognize that there are some arriving N-services patterns which always require N-bursts patterns for their multiplexing. Those are the service patterns (e.g. Figure 6 with $N = 4$) for which any pair of services cannot be multiplexed on the same burst. In other words, there always exists at least a column having all values set to 1 on their representation matrix (e.g. column t_3 in Figure 6a and 6b). For these service patterns, the optimization of their multiplexing is unneeded. They are then eliminated from the optimization process.

3.3. Clustering based on the number of used bursts

Normally, with an arriving service pattern, we can determine the minimal number of bursts needed for its multiplexing. As shown in the examples of Figure 6, the minimal number of bursts needed is 3, because the maximal overlap on all columns of their representation matrix is 3 (in Figure 7a, service

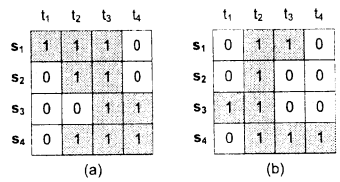

Figure 6. The services in these service patterns cannot be multiplexed

s_2 overlaps with s_3 and s_4 at timeslot t_3; and in Figure 7b, service s_2 overlaps with s_3 and s_4 at t_4). In other words, the minimal number of bursts needed is determined by the maximal number of overlapping cells (with value 1) on all columns. By this clustering way, the minimal number of bursts needed for an

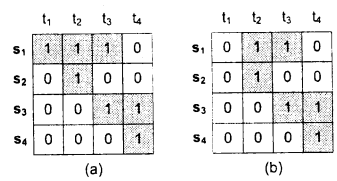

Figure 7. These service patterns require at least 3 bursts for their multiplexing

arriving service pattern cannot be lower than $M = \frac{N}{T}$, if N is multiple of T, or $M = \frac{N}{T} + 1$, otherwise. It means that we can cluster arriving N-services patterns into $(N - M)$ separated groups corresponding to the minimal number of bursts needed: the N-bursts group, the $(N - 1)$-bursts group, etc. and the

M-bursts group. For the N-bursts group, it corresponds to arriving service patterns which are unneeded to be optimized.

3.4. Removing the services which do not participate in the optimization process in a service pattern

For an N-services pattern clustered into the K-bursts group ($N > K$), some services in this pattern can be multiplexed into a same burst. But, there is the existence of the other services which cannot be multiplexed with any other. In other words, they are unneeded to participate in the multiplexing optimization and then they are removed to reduce the optimization delay. As shown in Figure 8a and 8b, the service s_1 in these two service patterns cannot be multiplexed with any other and then removed. A problem that appeared in this case is, with a given service pattern, how to search and remove the services which do not participate in its multiplexing optimization; and if this search-and-remove process takes a long delay. In fact, it is not necessary to have an exhaustive search-and-remove process. Based on the arriving services which are sorted, the following algorithm can return a good result, but it does not consume much time.

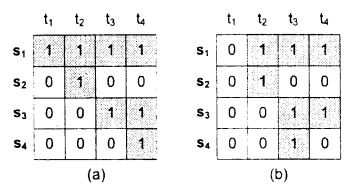

Figure 8. Service s_1 in these service patterns cannot be multiplexed with any other

Suppose that the services which are sorted in descending order in each service pattern.

1. Begin with the first service s_1 ($i = 1$);
2. Check if the service s_i can be multiplexed with the last service (s_N);
3. If yes, remove the service s_i and return the step 2 with service s_{i+1};
4. Terminate the algorithm. The rest services then really participate in the multiplexing optimization of the current service pattern.

By using this algorithm, a service pattern in the K-bursts group can reduce its size to an H-bursts group, where $H < K$. It means that the delay for its optimization process is then reduced considerably.

3.5. Clustering based on the same optimal grooming solution

The clustering based on the number of used bursts only clusters arriving services patterns into $N - M$ *"crude"* groups. In each of these groups, we can in fact cluster them into many other smaller groups which correspond to separate optimal grooming solutions. For example, two arriving service patterns in Figure 9 are all in the 2-bursts group, but corresponding to two

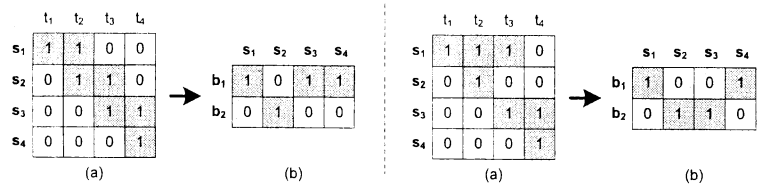

Figure 9. Two service patterns in the 2-bursts group have different optimal multiplexing solutions

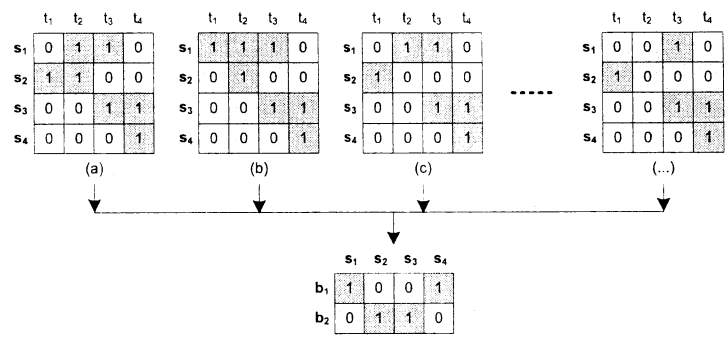

Figure 10. These service patterns in the 2-bursts group have the same optimal grooming solution

separate optimal multiplexing solutions. However, there also exist other service patterns in the 2-bursts group, which have the same optimal grooming solution, as shown in Figure 10. The simplest way to cluster the service patterns in a K-group ($N > K \geq M$) is based on the experiment: an arriving service pattern is first optimized to determine what its optimal grooming solution is. Basingon this result, this service pattern will be clustered into a right group. In conclusion, the clustering function in our clustering-learning model can be described hierarchically as shown in Figure 11. When a service pattern arrives, the services in this service pattern are sorted. The service pattern is next clustered by basing on the minimal number of used bursts. This phase also determines if this service pattern is unneeded to optimize its multiplexing (for the case in the N-bursts group). If it is the service pattern that needs optimizing, it is now checked if they have the services which do not participate in its multiplexing optimization. These services are then removed and the current service pattern is moved to a new group (with fewer bursts used). This

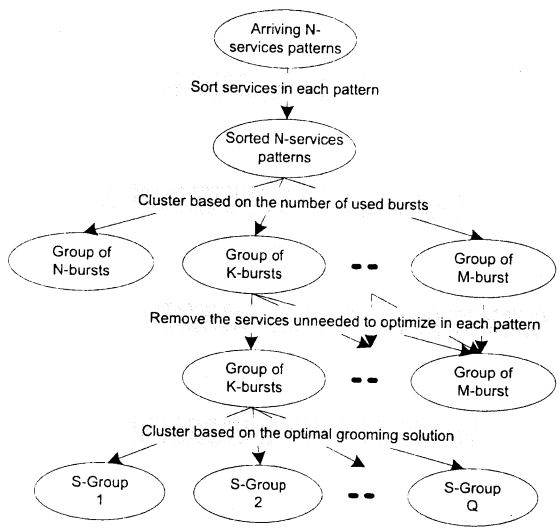

Figure 11. The hierarchic clustering of an arriving service pattern

service pattern is at last clustered one more time, but now based on its optimal grooming solution. This process clusters this service pattern into a smaller group in which all service patterns have the same optimal grooming solution.

3.6. Learning function

service pattern is at last clustered one more time, but now based on its optimal grooming solution. This process clusters this service pattern into a smaller group in which all service patterns have the same optimal grooming solution. Sometimes, an arriving service pattern is *"similar"* to another which has been optimized. The optimal grooming solution found and memorized can be reused for this service pattern. The learning function proposed in our clustering-learning model is based on this principle, as presented in Figure 12. Its objective is to eliminate the service pattern optimized and then reduce the average optimization delay. Initially, an arriving service pattern is input into the optimization unit (1) to determine its optimal grooming solution. Because of the time-limit for its grooming, a nearly optimal (instead of optimal) solution is returned (2). However, the optimization process is continued to determine an optimal solution. This result is then provided to the clustering unit (3).

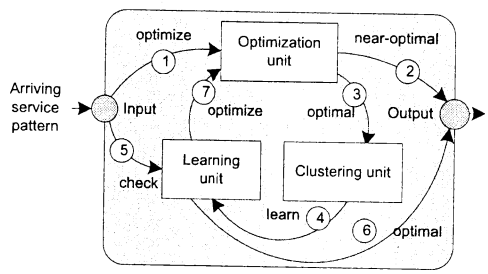

Figure 12. Learning model for reducing the delay

By the *"similarity"* of some arriving service patterns, the clustering unit clusters these service patterns into a same group, which corresponds to a found optimal grooming solution. The results of this clustering are then delivered to the learning unit to be stored (4).

When a new service pattern arrives: it is input to the learning unit to check if there exists a corresponding optimal solution (5). If one is found,

the corresponding optimal solution is returned immediately (6). If not, an optimization process is required to find its optimal solution (7).

However, the time required to check if a *"similarity"* service pattern exists will increase when the size of knowledge of optimized grooming solutions increases. This delay can exceed the time-limit for a grooming. The solution for this problem is that an arriving service pattern is input at the same time to the optimization unit (1) and the learning unit (5). The first result returned by one of two units is the optimal grooming solution for the current service pattern.

4. Simulation results and analysis

All simulations of our learning model were run on a PC Pentium 4, 1G RAM and 2.3 GHz. We tested the average delay for three cases: without clustering-learning model (as done also in [9]), with the clustering-learning model but without (CLModel-1) and with (CLModel-2) the operation of removing the services in a service pattern which does not participate in its optimization process. Essentially, the objective of CLModel-1 is to reduce the number of arriving service patterns that do not need optimizing, while CLModel-2 is for the purpose of minimizing the size of each arriving service pattern. Table 1 shows the result of our simulations over different size 5×8 (5 services of 8 timeslots), 10×8, 15×8, 20×8 and 30×8 of 32000 arriving service patterns.

Size	Without CLModel	With CLModel-1	With CLModel-2
5x8	0.101594	0.050313	0.022406
10x8	1.243156	1.133219	0.398906
15x8	5.878844	6.019969	2.277437
20x8	18.47891	18.45472	7.106563
25x8	42.76272	43.54294	17.67691
30x8	87.75941	87.70138	37.04206

Table 1. Average delay (ms) over the cases: without CLModel, with CLModel-1 and with CLModel-2

As shown in Table 1, the delay (in milliseconds) in the case with CLModel-1 is not better than the case without clustering-learning model. Instead (as shown in Figure 13), the probability of an arriving service pattern which does not need to be optimized (PU2) and is similar to another one optimized before (PoS) is too small to need optimizing (P2O), particularly for the service patterns with big size. So, no advantage is denoted between the case with CLModel-1 and the case without the learning model (see Figure 14).

However, in the case with the CLModel-2 model, the delay is considerably reduced in comparison with the two previous cases. Instead, as presented in [2], the delay of the optimization based on Hopfield networks is an exponential function of the problem size (the size of service pattern in this case). Therefore, if we can reduce the size of optimized service pattern, more delay can be minimized.

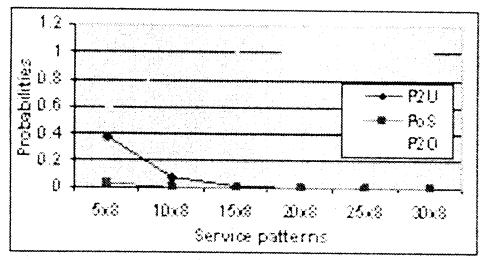

Figure 13. Probabilities over the cases: P2U, PoS and P2O

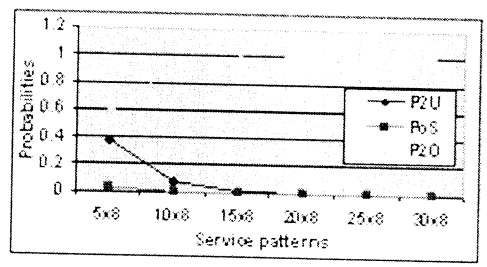

Figure 14. Comparison of the average delay of algorithms

5. Conclusion

The optimization based on Hopfield networks normally requires a considerable delay to find an optimal solution, but this has limited the applicability of this method in optical data transport, where the delay required for traffic grooming is short. So, reducing this delay is an important factor of our model of the traffic grooming optimization based on Hopfield networks. In [5, 6] we have proposed solutions to reduce this delay, but only for the method. There exist two other factors which influence the average optimization delay: the number of service patterns that need to be optimized and the size of optimized service patterns. This paper, therefore, has proposed a more complete solution to reduce these two factors. That is a clustering-learning model, which (1) eliminates arriving service patterns which do not need to be optimized its multiplexing, (2) removes the services which do not participate in the multiplexing optimization in an arriving service pattern, and (3) clusters arriving service patterns into the groups in which the service patterns in a group have the same optimal grooming solution. A learning function is also added to eliminate an arriving service pattern from its optimization process if it is *"similar"* to another optimized before. Basing on the simulation results, our clustering-learning model has proved its advantages. This improves the practicability of our model of traffic grooming optimization based on Hopfield networks.

References

[1] **Hu J.Q.,** Traffic grooming in WDM ring networks: A linear programming solution, *J. Opt. Networks,* **1** (11) (2002), 397-408.

[2] **Zhang X. and others,** An effective and comprehensive approach for traffic grooming and wavelength assignment in SONET/WDM rings, *SPIE Proc. Conf. All-Opt. Networking,* **8** (5) (2000), 608-617.

[3] **Wang J. and others,** Improved approaches for cost-effective traffic grooming in WDM ring networks: IPL formulations and single-hop and multihop connections, *J. Lightwave Technology,* **19** (11) (2001), 1645-1653.

[4] **Simmons J. and others,** Quantifying the benefit of wavelength add-drop in WDM rings with distance-independent and dependent traffic, *IEEE J. Lightwave Technology,* **17** (1) (1999), 48-57.

[5] **Vo V.M.N. and others,** Optimization of Services-into-Burst multiplexing based on Hopfield network, *IEEE Proc. of Systems Communications (ICHSN2005),* 2005, 235-240.

[6] **Vo V.M.N. and others,** Traffic switching optimization on optical routing by using Hopfield network, *Proc. of RIVF2004,* 2004, 125-130.

[7] **Lagoudakis M.G.,** *Neural networks and optimization problems - a case study: The minimum cost spare allocation problem,* technical report, University of Southwestern Louisiana, `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.49.4717`

[8] **Lillo W. and others,** On solving constrained optimization problems with neural networks: A penalty method approach, *IEEE Trans. on Neural Networks,* **4** (6) (1993), 968-972.

(Received February 15, 2009; revised September 12, 2010)

Vo Viet Minh Nhat and Le Manh Thanh
Hue University
Hue, Vietnam
`vvmnhat,lmthanh@hueuni.edu.vn`