Ressource Allocation in Communication Networks

Olivier Brun and Jean-Marie Garcia

LAAS / CNRS
7, Av. Du Colonel Roche, 31077 Toulouse, France
Tel : +33-561-336-478, E-mail : jmg@laas.fr

Abstract

A key problem in the design of communication networks is the planning of bandwidth allocation to communication demands. Shortest path routing is the traditional answer to this problem. However, it is well known that this can lead to poor network performances. We propose a new approach which proceeds in two steps. In the first step, the corresponding relaxated problem of load sharing routing is solved optimally by non-linear programming techniques. Then, a heuristic based on ACO techniques is used to derive a feasible solution to the original problem.

1 Introduction

Due to the increasing complexity of modern telecommunications networks, to the diversification of telecommunication services and to the concurrence between operators, the design of networks is becoming both increasingly strategic and difficult. There is thus a growing need for efficient network planning tools allowing to design networks at minimum cost. Such tools should also ensure QoS guarantees to communication flows and take into accounts network survivability and design evolution constraints.

Among the network planning problems faced by telecommunication operators is the network resource allocation problem. In [1], Christian Rolf Frei calls this problem the Resource Allocation In Networks (RAIN) problem and defines it as follows:

- **Given** a network of N nodes and M links represented by a graph $G=[X,U]$, the link bandwidth capacities C_u for all $u \in U$ and a set of K demands, where each demand $k=1, ..., K$ is defined by its source node $s(k)$, its destination node $t(k)$ and its bandwidth requirement d^k,
- **Find** one and only one route for each demand k so that the bandwidth requirements of the demands are simultaneously satisfied within the resource capacities of the links. In other words, if we let $p(k)$ be the number of paths in G having endpoints $s(k)$ and $t(k)$, and P_i^k be the i^{th} path between $s(k)$ and $t(k)$, for all $i = 1 \ldots p(k)$, then the problem amounts to finding a set of numbers $x_j^k \in \{0 , 1\}$ such that the following constraints are satisfied:

$$\sum_{k=1}^{K} \sum_{j=1}^{p(k)} P_j^k(u) x_j^k d^k \le C_u \qquad u = 1 \ldots M$$

$$\sum_{j=1}^{p(k)} x_j^k d^k = d^k \qquad k = 1 \ldots K$$

where we have used the same notation to denote the path P_j^k and the vector P_j^k defined by $P_j^k(u) = 1$ if $u \in P_j^k$ and 0 otherwise. The first set of constraints is related to resource capacity constraints while the second set of constraints is related to flow conservation constraints.

Direct applications of the RAIN problem include, for instance, the design of a virtual private network within a physical network, routing of virtual path connections (VPC) in an ATM network or routing the virtual channel connections (VCC) in the VPC network of an ATM network. Another important application is the placement of Label Switched Paths (LSP) in core MPLS networks.

The RAIN problem is intrinsically a combinatorial problem since flows routing is restricted to one and only one route. With this restriction, it was shown by Frei that the RAIN problem is NP-hard in the number of demands ([1]). In a previous work, Vedantham and Iyengar proved that, when the demands are subject to multiple additive or multiplicative QoS constraints, the allocation of every single demand is NP-hard by itself.

Several approaches were developed to solve the RAIN problem. They are based on linear programming techniques [2,4,5,6], greedy allocation heuristics [7,8], successive approximations techniques [3], meta-heuristics (Genetic algorithms [9], simulated annealing [10], etc.) or on constraint satisfaction programming techniques [1].

However, in our opinion, none of these approaches own *all* the desired features: (1) a feasible solution should be provided if at least one exists; (2) the method should compute the optimal solution according to some criterion, or at least provide some bounds on the distance between the computed solution and the optimal solution; (3) In case of allocation failure of one or more demands, explanations and correcting measures should be provided; (4) the solution should be obtained in a minimum computing time (although this is not the main issue in the network planning context).

In this paper, we propose a new approach to the RAIN problem. This approach is based on a combination of non-linear programming techniques and Ant Colony Optimization (ACO) as described in section 2. In section 3 extensive results and comparisons with other approaches are provided. Finally, in section 4, several conclusion are drawn.

2 A new approach to the RAIN problem

The main ideas of the algorithm we propose are the followings:

- To ensure that that the load induced by demands is well distributed, i.e. that no link is saturated while some other links are under-utilized, we use a non-linear penalty cost function. Beside a better utilization of network resources, it allows to provide a solution even when there are no feasible solution with respect to the capacity constraints (in which case the solution is analyzed to find correcting measures). Note also that this approach is also well suited to solve problems where a given blocking probability is allowed on network trunks.
- We observe that the optimal combinatorial solution is often quite ``close'' from the optimal load-sharing solution, i.e. the routes used by the former are generally a subset of the one used by the latter. Therefore the optimal load-sharing solution can be a good starting point when searching for an optimal solution to the RAIN problem. In this context, the variable x_j^k is considered as being continuous, and it represents the fraction (i.e. $0 \le x_j^k \le 1$) of flow k traveling along path P_j^k, for all $j = 1 \ldots p(k)$.

The main difficulty with this new approach concerns the method used to get a feasible solution to the RAIN problem from the optimal load-sharing solution. We propose to use an Ant Colony Optimization (ACO) technique.

2.1 Optimal load-sharing routing

Let $x^k = (x_j^k)_{j=1 \ldots p(k)}$ and $x = (x^k)_{k=1 \ldots K}$. Moreover, let us define $y_u^k = \sum_j P_j^k(u)\, x_j^k \; d^k$ and $y_u = \sum_{k=1}^K y_u^k$. Then y_u^k is the capacity used by flow k on link u, while y_u denotes the whole capacity used by all flows on link u. With these notations, the optimal load-sharing routing can be formulated as follows:

Minimize : $\Gamma(x) = \sum_{u=1}^{M} F(y_u, C_u)$

Subject to :

$$\sum_{j=1}^{p(k)} x_j^k = 1 \qquad k = 1 \ldots K$$

$$x_j^k \ge 0 \qquad j = 1 \ldots p(k) \quad k = 1 \ldots K$$

where the function F is given by (A_1 and A_2 are two adjustable parameters):

$$F(y,C) = \begin{cases} A_1 y^2 & \text{if} \quad y \le C \\ A_1 y^2 + A_2 (y - C)^2 & \text{if} \quad y > C \end{cases}$$

This optimization problem can be solved using the classical feasible directions methods, for instance the projected gradient algorithm. Let $x_j^k(q)$ be the fraction of flow k routed on route P_j^k at iteration q of the projected gradient algorithm and ν be the step done in the projected (minimization) direction. Then it is very easy to show that iteration q of the projected gradient algorithm proceeds as follows :

$$x_j^k(q+1) = x_j^k(q) - \nu \left\{ (p(k) - 1) \frac{\partial \Gamma}{\partial x_j^k} \; - \; \sum_{i \ne j} \frac{\partial \Gamma}{\partial x_i^k} \right\}$$

where,

$$\frac{\partial \Gamma}{\partial x_i^k} = d^k \sum_{u \in P_i^k} \frac{\partial F}{\partial y_u} \quad \text{and,}$$

$$\frac{\partial F}{\partial y_u} = \begin{cases} 2A_1 y_u & \text{if} \quad y_u \le C_u \\ 2A_1 y_u + 2A_2 (y_u - C_u) & \text{otherwise} \end{cases}$$

The efficiency of this algorithm will however be greatly reduced by the huge number of paths connecting the end-points of each demand in a real-sized network. Since each flow has finally to be routed on only one path, it appears natural to consider only a subset of these paths. The general idea of the path selection algorithm we have used is to progressively load the network with fractions of flows in R iterations (R being a small integer, say R=10). At the beginning of each iteration, all flows are untagged. During one iteration, the following steps are repeated K times:

1. Select an untagged flow k according to a uniform distribution,
2. Route demand d^k/R on its shortest path P_j^{*k} according to the metric defined by $F(y_u,C_u)$ (using Djikstra's algorithm). Tag flow k.
3. Update link loads y_u by propagating demand d^k/R along path P_j^{*k}, i.e. increment y_u by d^k/R for all link $u \in P_j^{*k}$.

This path selection algorithm allows to greatly reduce the number of paths since it yields at most R paths by flow. The projected gradient algorithm then allows to determine very efficiently the optimal load-sharing routing solution (but optimal only with respect to the subset of paths considered). However, this solution is not a valid solution of the RAIN problem since flows are routed on several routes. Nevertheless, this load-sharing

solution provides a good starting point to an ACO algorithm.

2.2 An ACO algorithm for the RAIN problem

The foraging behavior of real ants has been used by several authors to define a new population-based heuristic approach to combinatorial optimization problems, allowing positive feedback (equivalent to pheromone) to be used as the primary search mechanism [11,*12*]. Since its first development, Ant Colony Optimization (ACO) has been applied to a variety of problem areas, including for example the traveling salesman problem [13,14,15], static routing in circuit-switched telecommunications networks [16], and dynamic routing in packet-switched networks [17,18].

In our ACO algorithm, ants recursively build solutions to the RAIN by a random exploration of the solution space. Iterations are indexed by t, $1 \le t \le t_{max}$, where t_{max} is the user defined maximum number of iterations allowed. During iteration t, each ant l=1,...,m builds a solution in K steps by routing demands on a flow by flow basis, starting from a randomly chosen flow and moving from flow to flow according to a probabilistic transition rule. More precisely, during iteration t ant l iterates on the following steps until all flows are routed.

1. **Random selection of the flow k to be routed.** Ant l selects the flow k to be routed according to the following probabilistic rule:

$$P[\text{ant } l \text{ selects flow } k] = \begin{cases} \dfrac{1}{|\mathrm{N}_n^l(t)|} & \text{if} \quad k \in \mathrm{N}_n^l(t) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathrm{N}_n^l(t)$ is the set of all flows that have not yet been routed by ant l at step n of iteration t (initially, $\mathrm{N}_0^l(t)$ contains all flows). By exploiting $\mathrm{N}_n^l(t)$ ant l avoids routing a flow more than once. Once a flow k has been selected according to the previous probabilistic rule, the set of flows to be routed is updated according to $\mathrm{N}_{n+1}^l(t) = \mathrm{N}_n^l(t) - k$.

2. **Random selection of a path to route flow k**. The routing of flows k on path P_j^k is governed by a probabilistic rule given by:

$$P[\text{ant } l \text{ routes flow } k \text{ on path } P_j^k] = \frac{\left[\tau_j^k(t)\right]^\alpha \cdot \left[\eta_j^k\right]^\beta}{\sum_{i=1}^{p(k)} \left[\tau_i^k(t)\right]^\alpha \cdot \left[\eta_i^k\right]^\beta}$$

where,

- $\eta_j^k = 1/\sum_{u \in P_j^k} F(y_u+d^k, C_u)$ is a heuristic information, called visibility, which represents the heuristic desirability of routing flow k on path P_j^k ($1/\eta_j^k$ is the increase of cost due to the routing of flow k on path P_j^k).
- $\tau_j^k(t)$ is the amount of virtual pheromone trail associated to the routing of flow k on path P_j^k. Pheromone trail is updated at the end of each iteration t and is intended to represent the learned desirability of routing flow k on path P_j^k. It reflects the experience acquired by ants during problem solving. Initially, the pheromone trail $\tau_j^k(0)$ are set to $\tau_0 x_j^k$, where τ_0 is a small positive constant and x_j^k is the fraction of flow k routed on path P_j^k by the optimal load-sharing routing solution.
- α and β are two adjustable parameters that control the relative weights of trail intensity $\tau_j^k(t)$ and visibility η_j^k.

Let $P_j^{*k,l}(t)$ be the path randomly selected by ant l to route flow k at iteration t.

3. **Update of link loads.** Link loads are updated by propagating the demand d^k along path $P_j^{*k,l}(t)$.

Each ant l repeats K times the previous steps to build a solution. Once all ants have build a solution, the amounts of virtual pheromone trail associated to the routing decisions are updated in the following way:

$$\tau_j^k(t+1) = (1-\rho)\tau_j^k(t) \quad + \quad \sum_{l=1}^{M} \Delta\tau_j^{k,l}(t)$$

where ρ, $0 \le \rho \le 1$ is the coefficient of decay of the trail intensity, necessary to ensure an efficient solution space exploration, and $\Delta\tau_j^{k,l}(t)$ represents the quantity of pheromone associated by ant l to the routing of flow k on path P_j^k; the value $\Delta\tau_j^{k,l}(t)$ depends on how well the ant has performed:

$$\Delta\tau_j^{k,l}(t) = \begin{cases} Q \Big/ \sum_{u=1}^{M} F\Big(\sum_{q/u \in P_{j^*}^{q,l}(t)} d^q, C_u\Big) & \text{if} \quad \mathrm{P}_{j^*}^{k,l}(t) = P_j^k \\ 0 & \text{otherwise} \end{cases}$$

where the denominator is the cost of the solution build by ant l and Q is a parameter (set to the value of the cost of the optimal load-sharing routing solution).

3 Results

In this section, we present the results obtained on several testbed networks with the algorithm described in the

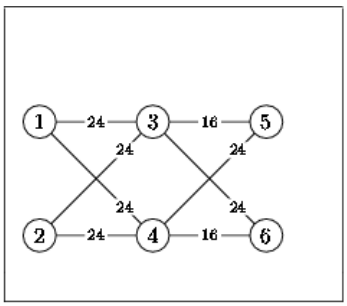

Figure 1. Testbed Network A

previous section. These results are compared with those obtained using the BIH algorithm [1]. Table 1 gives the values we have used for the different parameters in these experiments.

	α	β	ρ	Q	t_{max}	A_1	A_2
value	1.0	0.2	0.2	0.01	300	0.1	100.0

Table 1. Parameter values of the algorithm.

Consider first the testbed networks depicted on figures 1 and 2: 30 demands have to be routed in network A and 49 demands have to be routed in network B. Results obtained with these networks are given in table 2. For network A, both algorithms perform very well since the cost obtained is equal to the lower bound given by the projected gradient algorithm. However, the computing time of the BIH algorithm is far more important due to the huge number of backtracks it uses in such constrained cases (several links are saturated). For network B, the BIH algorithm is significantly much faster, but it provides a solution whose cost (7.8% from the lower bound) is greater than the cost of the solution obtained with our algorithm (2.2% from the lower bound).

	Network A		**Network B**	
	Cost	Computing time (s)	Cost	Computing time (s)
ACO algorithm	396.8	0.88	3069.8	5.85
BIH algorithm	396.8	32.18	3237.7	0.13

Table 2. Results obtained for networks A and B.

Consider now the network depicted on figure 3. This network is inspired from a real network interconnecting some major cities in Europe. For this network, we have randomly generated 50 instances of the RAIN problem (with numbers of demands ranging from 3 to 47). Among these 50 instances, 19 were infeasibles.

For 18 of the 19 infeasible instances, our algorithm has obtained the lower bound given by the projected gradient algorithm with computing times ranging from 0.6 second to 4.2 seconds (while the computing times of the BIH algorithm were sometimes up to approximately 2 hours). Moreover, the solutions obtained easily allow

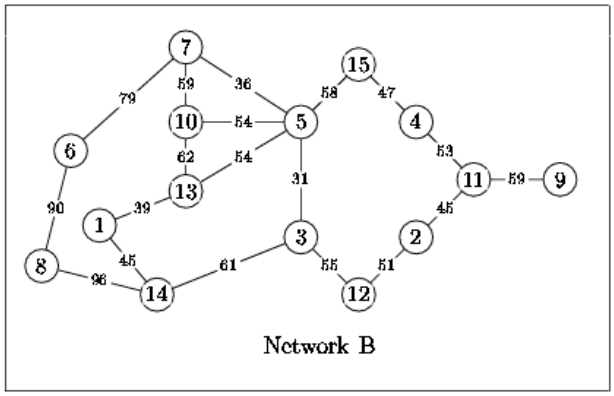

Figure 2. Testbed Network B

to modify the network (by adding capacity to some links) or the demands (by reducing the bandwidth requirements of some flows) so as to obtain a feasible solution whereas the BIH algorithm only conclude to the absence of a feasible solution. For the 31 feasible problems, the average computing time of our algorithm was 0.4 second and the average distance of the cost from the lower bound given by the projected gradient algorithm was 0.9%. The average computing time of the BIH algorithm was only 0.025 second, but the average distance of its cost from the lower bound was 8.42%.

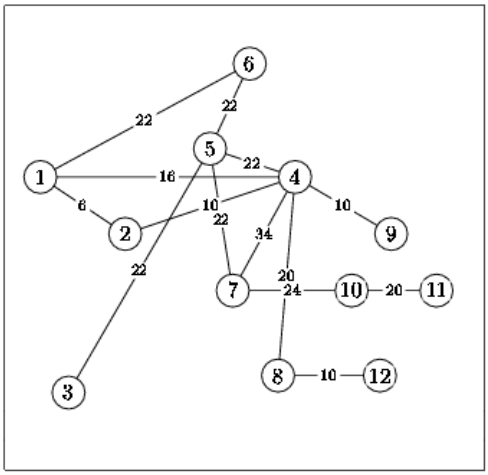

Figure 3. Testbed network inspired from a real European network.

4 Conclusion

We have presented a new approach to the RAIN problem and experimental results obtained on several networks. The method developed has several advantages: computing times are really reasonnables, solutions provided are in general near-optimal (as can be seen from experimental results) and the method provides guidelines to the network planner when the solution is not feasible. Although it can not be guaranteed that a feasible solution is found if at least one exists, it seems that the method performs very well even for very constrained problems. We are currently extending this

method to handle several priorities among demands.

References

[1] C. Frei and B. Faltings, "*Bandwidth allocation heuristics in communication networks*" 1ère Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL'99), 53-58, Roscoff, France, 1999.

[2] S. Cosares and I. Saniee, "*An optimization problem related to balancing loads on SONET rings*" Telecommunication Systems, 3:165-181, 1994.

[3] M. Minoux, "*Planification à court et à moyen terme d'un réseau de télécommunications*" Annales des Télécommunications, Vol. 29, No 11-12, 1974.

[4] M. Herzberg, D. J. Wells and A. Herschtal. "*Optimal Resource Allocation for Path Restoration in Mesh-Type Self-Healing Networks*" International Teletraffic Congress (ITC), 15:351-360, 1997.

[5] T. H. Wu "*Fiber Network Service Reliability*" Artech House, Boston-London, 1992.

[6] G. R. Ash "*Dynamic Routing in Telecommunications Networks*" McGraw Hill, 1998.

[7] P. Chanas, O. Goldschmidt and M. Burlet "*Routing Virtual Paths in ATM Networks*" Journal of Heuristics, 1999.

[8] P. Chanas "*Réseaux ATM: conception et optimisation*" PhD thesis, France Telecom CNET, Sophia-Antipolis, France, 1998.

[9] D. E. Goldberg "*Genetic Algorithm in Search, Optimization and Machine Learning*" Addison Wesley, 1989.

[10] S. Kirkpatrick, C.D. Gelatt and M. P. Vecchi "*Optimization by simulated annealing*" Science 220(4598):671-680, 1983.

[11] E. Bonabeau, M. Dorigo and G. Theraulaz "*Swarm Intelligence - From Natural to Artificial Systems*" Oxford University Press, 1999.

[12] D. Corne, M. Dorigo and F. Glover Eds. "*New Ideas in Optimization*" McGraw Hill, 1999.

[13] A. Colorni, M. Dorigo and V. Maniezzo "*Distributed Optimization by Ant Colonies*" Proc. First European Conf. on Artificial Life (ECAL'91), Paris, France, 1991.

[14] A. Colorni, M. Dorigo and V. Maniezzo "*An investigation of some properties of an ant algorithm*" Proc. Parallel Problem Solving from Nature Conference (PPSN'92), Brussels, Belgium, 1992.

[15] M. Dorigo, V. Maniezzo and A. Colorni "*The Ant System: optimization by a colony of cooperating agents*" IEEE Transactions on Systems, Man and Cybernetics, 26(1), 1993.

[16] R. Schoonderwoerd, O. E. Holland, J. L. Brutten and L. J. M. RothKrantz "*Ant based load balancing in telecommunication networks*" Adaptative Behaviour, 5(2), 1997.

[17] G. Di Caro and M. Dorigo "*Mobile Agents for Adaptative Routing*" Proc. 31st Hawaii Intl. Conf. Systems Sciences (HICSS-31), Kohala Coast, Hawaii, 1998.

[18] G. Di Caro and M. Dorigo "*AntNet: Distributed Stigmergetic Control for Communications Networks*" Jour. of Artificial Intelligence Research, 9:317-365, 1998.