Applied Mathematics and Computation 217 (2011) 5338–5346

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Discrete Capacity Assignment in IP networks using Particle Swarm Optimization

Emilio Carlos Gomes Wille *, Eduardo Yabcznski, Heitor Silvério Lopes

Federal Technology University of Paraná - UTFPR. 80230-901, Av. Sete de Setembro, 3165, Curitiba-PR, Brazil

ARTICLE INFO

Keywords:
IP networks
Quality-of-Service
Discrete Capacity Assignment
Particle Swarm Optimization

ABSTRACT

This paper presents a design methodology for IP networks under end-to-end Quality-of-Service (QoS) constraints. Particularly, we consider a more realistic problem formulation in which the link capacities of a general-topology packet network are discrete variables. This Discrete Capacity Assignment (DCA) problem can be classified as a constrained combinatorial optimization problem. A refined TCP/IP traffic modeling technique is also considered in order to estimate performance metrics for networks loaded by realistic traffic patterns. We propose a discrete variable Particle Swarm Optimization (PSO) procedure to find solutions for the problem. A simple approach called Bottleneck Link Heuristic (BLH) is also proposed to obtain admissible solutions in a fast way. The PSO performance, compared to that one of an exhaustive search (ES) procedure, suggests that the PSO algorithm provides a quite efficient approach to obtain (near) optimal solutions with small computational effort.

1. Introduction

The classical approaches to the optimal design and planning of packet networks, extensively investigated in their early days [1,2], focused on the network-layer infrastructure. Such approach neglected end-to-end (e2e) Quality-of-Service (QoS) issues, and Service Level Agreement (SLA) guarantees. Today, TCP/IP (Transmission Control Protocol/Internet Protocol) is the most widely used set of protocols of the Internet, being TCP answerable for a great amount of the total traffic volume [3]. Recently, investigations considering Internet traffic have shown that IP packets do not arrive at router buffers following a Poisson process [4], but there is a correlation degree, which can be partly due to the TCP control mechanisms. This traffic profile may produce transient network congestion, which can cause performance problems to the network system.

To avoid these problems we need a correct modeling. In [5], the authors proposed a network design and planning approach that considers the dynamics of packet networks, as well as the effect of protocols at the different layers of the Internet architecture on the e2e QoS experienced by end-users. The proposed approach firstly maps the end-user performance constraints into transport-layer performance constraints, and then into network-layer performance constraints. This mapping process is then considered together with a refined TCP/IP traffic modeling technique that is both simple and capable of producing accurate performance estimates for general-topology packet networks loaded by realistic traffic patterns. However, the optimization procedure employed was based on continuous link capacities. Generally, this is not the case in real-world systems.

In this paper, we present and solve the Discrete Capacity Assignment (DCA) problem subject to e2e QoS constraints (specifically we consider the average packet delay), where link capacities are picked from a set of discrete values. This leads to a

* Corresponding author.
E-mail address: ewille@utfpr.edu.br (E.C.G. Wille).

0096-3003/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2010.12.003

constrained combinatorial optimization problem, classified as NP-complete [6]. Therefore, we use the Particle Swarm Optimization (PSO) [7] heuristic to find solutions for the problem. Originally, PSO was created to work with continuous variables. In this work we also extend PSO in order to deal with discrete variables. To evaluate the performance of our proposed approach, PSO results (and its effectiveness) will be compared with the optimal solutions found by an exhaustive search (ES) procedure. Finally, a simple heuristics called Bottleneck Link Heuristic (BLH) is also proposed to obtain, in a fast way, admissible solutions to the DCA problem.

The rest of the paper is organized as follows. Section 1.1 briefly mentions some previous work in the field of packet network design. Section 2 describes basic assumptions, and presents the formulation of the optimization problem. Section 3 illustrates the heuristic solution procedures. Numerical results are discussed in Section 4. Conclusions are given in Section 5.

1.1. Related work

In this section we briefly summarize previous work on discrete capacity allocation. The objective is to establish a parallel between this work and related publications which focus on similar problems and use metrics related to the transmission delay.

Maruyama and Tang [8] propose a heuristic solution for the discrete allocation problem with different classes of IP packets. The authors aimed at minimizing costs and satisfy average transmission delay constraints, for different classes of IP packets. The problem was modeled in a way similar to ours, but packets were classified according their priority level (priority assignment). This classification was done according to the length of packets, the transmission delay constraints, the average length of the path (end-to-end) and the transmission rate. The algorithm used to solve the problem limits the number of priority levels according to the number of classes. This is done for limiting the processing time. Several experiments were done, showing a significant decrement in cost when different priority levels were considered for the different classes of packets.

Ersoy et al. [9] proposed two artificial intelligence-based metaheuristics to solve the problem previously cited in [8]. They used simulated annealing and genetic algorithms, besides a deterministic method, devised by the authors, named MT-CA. Simulated annealing is based on an analogy with thermodynamics, when observing the behavior of atoms of a given material during controlled cooling, after being heated. This method performed particularly well when the number of packets and priority levels were high. On the other hand, genetic algorithms were shown to be slower than simulated annealing, but still faster than MT-CA. Also, as the size of the network grows, the processing time required by genetic algorithms increases less than that required by MT-CA. This suggests the applicability of genetic algorithms to problems with large-scale networks.

In Liu et al. [10], a different approach was proposed. Given a set of discrete capacities and respective costs, for each link both the speed (type) and number of channels (capacity) are selected, constrained by the transmission end-to-end average delay for each source/destination pair. The method is divided in three steps. First, the capacity of the link is chosen. Next, the number of channels for the link is computed. Then, a function for computing the cost of the network is evaluated. The algorithm prevents unnecessary increment of link capacity by considering two functions. The first one computes the significance of the link, given by the number of routes through it. The second function evaluates the economy of the link, considering the variation of the average delay for the link as the cost of the chosen channels change.

Ferreira and Luna [11] confront an extension of the capacity assignment problem. They consider the problem of capacity and flow assignment where the routing assignments and capacities are considered as decision variables, also known as capacity and flow assignment problem (CFA problem). The interested reader may refer to the references therein for a deeper discussion on this subject.

It is worth to point that, in the above cited works, the queue model used was the classic $M/M/1$ queue [12], while in our approach we preferred a queue with *batch arrivals* in order to better model the system. The batch arrival model, like $M_{[X]}/M/1$ queue, is motivated physically from TCP's behavior [13].

2. Problem statement

2.1. Network and queueing models

The network infrastructure is represented by a graph $G = (V, E)$ in which V is a set of nodes (with cardinality N) and E is a set of edges (with cardinality L). Each node represents a network router and the edges represent physical links connecting a router to another. Each output interface of each router is modeled by a queue with finite buffer. For a given link (i,j), the flow f_{ij} is defined as the average quantity of information transported by this link, while its capacity C_{ij} is a measure of the maximal quantity of information that can be transmitted (both are given in bits per second-bps).

According to [4], IP packets do not arrive at router buffers following a Poisson process. However, there is a correlation degree, which can be partly due to the TCP control mechanisms. In [14], the authors for the first time abandon the Markovian assumption in favor of a long range dependence (LRD) traffic model. However, the relation among traffic, capacity and queueing delay is not expressed in closed-form. As a consequence, the mathematical formulation of the problem may be unnecessarily complicated, without any benefits.

In order to consider the traffic burstiness induced by TCP, we choose a specific kind of queue to model each router inside the network topology [5]. Thus, we choose the $M_{[X]}/M/1$ queue, i.e. a Markovian queue with *batch arrivals* [12]. The batch size varies between 1 and W with distribution $[X]$, where W is the maximum TCP window size expressed in segments. Indeed, a wide range of experiments performed in [13] certify the accurate estimates of network layer metrics by considering $M_{[X]}/M/1$ models.

Lets consider that packet lengths are exponentially distributed with mean $1/\mu$ (bits/packet) [1,2]. Additionally, we define the *arrival rate* $\lambda = \mu \cdot f$ (packets/s), and the *link utilization factor* $\rho = f/C$. Hence, packets experience delays caused by routers which *average value* ($\overline{T}$) given by the following expression (notice that the subscript (ij) was dropped for the sake of clarity):

$$\overline{T} = \frac{K}{\mu}\frac{1}{C-f} \quad \text{with} \quad K = \frac{m'_{[X]} + m''_{[X]}}{2m'_{[X]}} \tag{1}$$

where $m'_{[X]}$ and $m''_{[X]}$ are the first and second moments of the batch size distribution [12].

The traffic requirements and the traffic routing uniquely determine the flow of each link. The average traffic requirements between nodes are represented by a traffic matrix $\Gamma = \{\gamma_{sd}\}$, where the traffic γ_{sd} between a node pair (s,d) represents the average number of bps sent from source s to destination d.

The traffic demand between a source/destination pair (s,d) is routed along the minimum cost path. If multiple paths exist with the same minimum cost, the traffic demand is evenly divided among all these paths. These paths are assumed fixed and known in advance.

Let $\Delta = \left\{\delta_{ij}^{sd}\right\}$ be a matrix whose elements are equal to one if link (i,j) is in the path (s,d) or zero, otherwise. Thus, a flow on link (i,j) results from the sum of the traffics that are routed on this link, i.e. $f_{ij} = \sum_{s,d} \delta_{ij}^{sd} \gamma_{sd}$.

2.2. The Discrete Capacity Assignment problem

In this subsection we focus on the Discrete Capacity Assignment problem, i.e. the selection of the link capacities from a discrete set of values. We solve the DCA problem considering the e2e delay constraints. Given the network topology, the traffic requirements, and the routing, the DCA problem is formulated as the following optimization problem:

$$Z_{DCA} = min \sum_{i,j} \upsilon_{ij} \cdot d_{ij} \cdot C_{ij} \tag{2}$$

subject to:

$$\overline{T}_{sd} = \frac{K}{\mu} \sum_{i,j} \frac{\delta_{ij}^{sd}}{C_{ij} - f_{ij}} \leqslant Delay_{sd}, \quad \forall (s,d) \tag{3}$$

$$C_{ij} > f_{ij} > 0, \quad \forall (i,j) \tag{4}$$

$$C_{ij} \in S, \quad \forall (i,j) \tag{5}$$

The objective function (2) represents the total link cost, which is the sum of the cost functions of all links (i,j). The cost function is a linear function, where d_{ij} is the physical length of the link, and υ_{ij} is the monetary cost in \$/Mbps/km/year. Eq. (3) is the e2e packet delay constraint for each source/destination pair. It says that the total amount of delay $\overline{T}_{sd}$ experienced by all the flows routed on path (s,d) should not exceed $Delay_{sd}$. If multiple paths exist for an specific source/destination pair these constraints are split accordingly. Constraints (4) are non-negativity constraints. S corresponds to a set, with cardinality N_s, of discrete capacities (constraints (5)). $Delay_{sd}$ is obtained following the procedure presented in [5].

We notice that the above stated DCA problem is a constrained combinatorial optimization problem, and its global optimum can be found using an exhaustive search method. However, this algorithm is very time-consuming. A suboptimal, but quite good, solution to this problem can be found using the proposed PSO heuristic.

3. Solution methods

In this section we present the solution methods used to solve the DCA problem.

3.1. Particle Swarm Optimization

The Particle Swarm Optimization is a population-based metaheuristic introduced around a decade ago by Kennedy and Eberhart [7]. PSO belongs to a class of methods known as evolutionary computation and it is inspired in the emerging properties of collective behavior of some animals, such as bird flocking, bee swarming, and fish schooling. For instance, in birds flocking, the velocity of each element is dynamically adjusted according to the velocity of the other surrounding elements. The influence of a bird on another is sensed by keeping an average distance between them.

At each instant of time (t), the position and velocity of particles are adjusted according to the position (X) and velocity (V) in the previous time ($t-1$), as follows:

$$V_i(t) = V_i(t-1) + \varphi_1 \cdot r_1 \cdot [X_{blp} - X_i(t-1)] + \varphi_2 \cdot r_2 \cdot [X_{bgp} - X_i(t-1)] \quad (6)$$

$$X_i(t) = X_i(t-1) + V_i(t) \quad (7)$$

where φ_1 and φ_2 are positive constants; r_1 and r_2 are normally distributed random values in the range [0,1]; $X_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{id}(t))$ represent the current location of the i-th particle; d is the dimension of the vector represented by a particle; X_{blp} represent the previous best local position of the i-th particle; X_{bgp} represent the best global position found up to the moment by the whole swarm; $V_i(t) = (v_{i1}(t), v_{i2}(t), \ldots, v_{id}(t))$ is the velocity of the i-th particle.

Eq. (7) defines how the position of particles is dynamically adjusted, taking into account the movement in the d-dimensional space given by Eq. (6). Notice that this Equation consists of three terms. The first term is the *momentum* part, meaning that the velocity cannot be changed abruptly. The second term is the *cognitive* part, that allows the particle to learn from its own flying experience, keeping track of the best position it has encountered up to the moment. The third term is the *social* part, and represents the collaboration among particles, allowing a particle to learn from the swarm's experience. The balance among these three parts determines the global or local convergence of the swarm.

When the sum of the three terms of Eq. (6) exceeds a user-defined value $\pm V_{max}$, the velocity is clamped to this value. High values of V_{max} can make the particles potentially fly towards good solutions, but without reaching them. On the other hand, low values of V_{max} can lead particles to converge to a local solution.

The original procedure for implementing PSO is as follows [15]:

(1) Initialize a population of particles with random positions and velocities on the d-dimensional search space of the problem.
(2) For each particle, evaluate its vector as a possible solution for the problem, by using a fitness function.
(3) Compare each particle's fitness with its X_{blp}. If the current value is better than X_{blp}, then set X_{blp} equal to the current value of $X_i(t)$.
(4) Among all particles, identify the one with the best fitness, and assign to X_{bgp} the current position of such particle. In some applications, only the surrounding neighbors can be considered instead of the whole swarm for updating velocity.
(5) Update the velocity and position of all particles according to Eqs. (6) and (7) and, if necessary, restrict velocity to $\pm V_{max}$.
(6) Loop to step (2) until a stop criterion is met, usually a sufficiently good solution or a maximum number of iterations.

There are many variations of the algorithm above described. In general, it is useful to have some strategy to preserve diversity in the swarm, so as to avoid fast convergence to local optima. Periodic explosions of the swarm can do this job by observing the agglutination of particles around a local optimum and resetting them to random positions, but keeping X_{bgp} unchanged [16].

Originally, PSO was created to deal with problems with continuous variables. An important extension of the algorithm was the introduction of binary rather than continuous variables, thus allowing PSO to deal with combinatorial problems [17,18]. In this implementation, each element of vector $X_i(t)$ is a bit, and the velocity is used as a probability to determine whether each bit will be in 0 or 1. After the evaluation of the n-th element of the velocity vector of the i-th particle (Eq. (6), now, with binary values), the following sigmoid function is used to decide which value will have the particle:

$$s(v_{in}) = \frac{1}{1 + \exp(-v_{in})} \quad (8)$$

if $r_i < s(v_{in})$ then $x_{in} = 1$; otherwise, $x_{in} = 0$. r_i is a random number, which comes from an uniform distribution (in the range [0,1]).

Another useful improvement in the basic PSO is the use of an *inertia weight* w, that multiplies the velocity term in Eq. (6). Large values of w facilitate a global search, and small values favor local search. In general, w starts large and is decremented during the evolution of the algorithm. Usual values for the parameters are: $\varphi_1 = \varphi_2 = 2$ and w decreases from 0.9 to 0.4 [15].

3.2. Applying PSO to the network design

As explained before, the problem solution corresponds to the selection of capacities for each link (among the set of the possible ones), always respecting the delay constraints.

The key point in a constrained optimization process is to deal with the constraints. Many methods were proposed for handling constraints. It is possible to group them into four categories: methods based on preserving feasibility of solutions; methods based on penalty functions; methods that make a clear distinction between feasible and infeasible solutions; and other hybrid methods. To cope with the constraints, we proposed two modifications regarding the original PSO.

(1) During initialization, all the particles are repeatedly initialized until they satisfy all the constraints.
(2) During the evolving process of PSO, our algorithm is based on a penalty function (if constraints are not satisfied) using the following fitness function, where η is the penalty value:

$$F = \begin{cases} \sum_{(i,j)} \upsilon_{ij} \cdot d_{ij} \cdot C_{ij}; & \overline{T}_{sd} \leqslant Delay_{sd}, \quad \forall (s,d) \\ \eta \cdot \sum_{(i,j)} \upsilon_{ij} \cdot d_{ij} \cdot C_{ij}; & \text{otherwise} \end{cases} \tag{9}$$

Thus, the following procedure is done to apply the PSO algorithm to the DCA problem: after the evaluation of Eqs. (6) and (7), each variable $x_{in}(t)$ is rounded up to the nearest integer. Then, this value is mapped to one element of the capacities set S. The proposed algorithm is explained below with the aid of an example. For instance, having three values to be selected from S for the capacities of each link, we set $x_{min} = 0$ and $x_{max} = 3$. Let $X_i = (0.45, 2.67, 2.21)$, after rounding up we obtain $X_i = (1,3,3)$. If $S = \{15,20,50\}$, then the selected capacities are: $C = (15,50,50)$. If a capacity C_{ij} is smaller than the respective flow f_{ij}, it is important to select the next value for C (from S), respecting constraints (4). After the evaluation of the fitness, X_{blp} and X_{bgp}, are updated, the stop criterion is verified and, if it was not reached, the cycle is repeated.

3.3. The Bottleneck Link Heuristic

In this section we propose a simple heuristic to obtain feasible solutions (not optimal ones) to the DCA problem. This heuristic relies on the concept of *bottleneck link*. In our context, the bottleneck link for a given data path is the link that produces the largest delay. The heuristic corresponds to identify and eliminate the bottleneck link, keeping solutions admissibility, in an iterative fashion [19]. The proposed Bottleneck Link Heuristic (BLH) is as follows:

(1) Initialize each link with a capacity value C greater than its respective link flow. It is important to select the smaller value for C (from S), respecting constraints (Eq. (4)).
(2) Evaluate the delays constraints (Eq. (3)):
  (a) If all constraints are satisfied the algorithm ends.
  (b) Otherwise continue.
(3) Make a list of link delays sorted in decrescent order.
(4) Find the bottleneck link:
  (a) If all capacity values (from S) were already tested for this link, eliminate the link at the top of the list of link delays and loop to step (4).
  (b) Otherwise select the next value of capacity for this link (from S).
(5) Loop to step (2).

4. Computational experiments and results

In this section, we present results obtained considering three randomly generated topologies. We emphasize that our approach does not rely on the properties of any specific network topology and, therefore, it can be applicable to any topology or graph. The traffic matrices were generated by picking the traffic intensity of each source/destination pair from an uniform distribution. For each topology, we solved the DCA problem using both the PSO and the ES approaches. The algorithms (PSO and ES) were implemented using C programming language, and the experiments were done in a desktop computer with 1 GHz processor, and 512 kBytes of RAM.

The set of experiments aimed at investigating the impact of the network dimension on the network cost, PSO performance and processing time. For the PSO we set: $w = 0.5$, $\varphi_1 = \varphi_2 = 2.0$, $\eta = 4.0$, and $\upsilon_{ij} = 1.0$. In the ES approach the number of iterations grows very quickly as the number of network links increases. In this case, it is necessary to search in a solution space of about $(N_s)^L$ candidate solutions, where N_s is the cardinality of set S and L the number of links.

4.1. PSO and ES results

Topology 1: First we considered a simple network, with 12 links and 11 delays constraints, where $S = \{4,6,10,20,50\}$ Mbps, $d_{ij} = 150$ km, $\forall (i,j)$, and $Delay_{sd}/K_1 = 0.72$, $\forall (s,d)$. The link flows (traffic 1.a), in Mbps, evaluated based on the traffic matrix, are $f = (16,11,21,21,2,14,24,19,9,3,8,21)$, as shown in Fig. 1. The optimal solution corresponds to the following list of capacities $C = (20,20,50,50,4,20,50,50,20,6,10,50)$, and the best (minimum) cost value is 52500/year (both found by the ES approach). It is important to note that the solutions space for this problem has about $(N_s)^L = 2.4 \times 10^8$ candidate solutions.

The PSO performance has been evaluated, and the results are presented below. Six different combinations of the number of particles (N_p) and the number of iterations (N_{itr}) were considered (each combination is called a test). Since the PSO is an stochastic algorithm, each test was repeated 500 times with different random seeds. The success rate, best and average objective function values were recorded. The success rate corresponds to the ratio of runs for which the PSO found the global minimum rather than being trapped into a local minimum. A second set of experiments was done using another traffic profile (traffic 1.b), which is twice the traffic 1.a.

Tables 1 and 2 summarize the performance of the PSO approach for the above scenario. These tables clearly show the impact of the PSO parameters on the algorithm performance; as expected, the cost standard deviations (*St.Dev.*) reduce

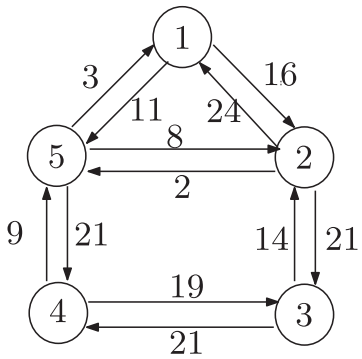

Fig. 1. Topology 1.

by increasing the N_p and N_{itr} parameters. A small standard deviation indicates that a increased number of good solutions are close to the optimal one.

Topology 2: The second set of tests aimed at investigating the design of a network with 24 links and 16 delays constraints, where $S = \{6, 8, 34, 51, 155\}$ Mbps, and $Delay_{sd}/K_1 = 0.72$, $\forall(s,d)$. The range of link flows were between 5 Mbps and 54 Mbps, with average 21.4 Mbps (traffic 2.a). The link distances were uniformly distributed in the interval 5–50 km. The solutions space for this case has about $(N_s)^L = 5.9 \times 10^{16}$ candidate solutions. The best cost value found by the ES approach was 17938.4/year.

For each test, 500 runs were performed. Tables 3 and 4 summarize the performance of the PSO approach for the above problem.

We observed that PSO successfully found the optimum for all cases within few iterations and with a high success rate, even in the case of small number of particles.

Table 1
Tests for topology 1 and traffic 1.a.

	N_p	N_{itr}	Succ. rate	Best	Average	St. Dev.
1	5	50	387/500	52500	53548.5	2215.2
2	10	50	474/500	52500	52737.0	997.4
3	30	50	500/500	52500	52500.0	0.0
4	5	100	391/500	52500	53515.5	2180.1
5	10	100	460/500	52500	52863.0	910.9
6	30	100	500/500	52500	52500.0	0.0

Table 2
Tests for topology 1 and traffic 1.b.

	N_p	N_{itr}	Succ. rate	Best	Average	St. Dev.
1	5	50	492/500	73500	73593	599.7
2	10	50	500/500	73500	73500	0.0
3	30	50	500/500	73500	73500	0.0
4	5	100	486/500	73500	73659	330.6
5	10	100	500/500	73500	73500	0.0
6	30	100	500/500	73500	73500	0.0

Table 3
Tests for topology 2 and traffic 2.a.

	N_p	N_{itr}	Succ. rate	Best	Average	St. Dev.
1	10	100	276/500	17938.4	18864.1	430.0
2	30	100	475/500	17938.4	18021.1	83.8
3	50	100	500/500	17938.4	17938.4	0.0
4	10	300	265/500	17938.4	18988.7	233.4
5	30	300	483/500	17938.4	17997.2	37.9
6	50	300	500/500	17938.4	17938.4	0.0

Table 4
Tests for topology 2 and traffic 2.b.

	N_p	N_{itr}	Succ. rate	Best	Average	St. Dev.
1	10	100	409/500	32750.4	33136.6	160.4
2	30	100	500/500	32750.4	32750.4	0.0
3	50	100	500/500	32750.4	32750.4	0.0
4	10	300	430/500	32750.4	33040.0	76.0
5	30	300	500/500	32750.4	32750.4	0.0
6	50	300	500/500	32750.4	32750.4	0.0

Topology 3: Finally, we analyzed a more challenging problem. The design of a network with 48 links and 25 delays constraints and $Delay_{sd}/K_1 = 0.72$, $\forall(s,d)$. The range of link flows were between 5 Mbps and 90 Mbps, with average 33.3 Mbps (traffic 3.a). The distances were uniformly distributed in the interval 3–25 km. In this case, we used two sets of capacities: $S = \{8,34,44,51,155\}$ Mbps, with $(N_s)^L = 3.5 \times 10^{33}$ candidate solutions; and $S = \{10,25,34,44,51,70,90,120,155,190\}$ Mbps, with a solution space of about $(N_s)^L = 10 \times 10^{48}$. The corresponding best (minimum) cost values are 18908.0/year and 12605.8/year, respectively.

For each test, 500 runs were performed. Tables 5 and 6 summarize the performance of the PSO approach.

It is observed that the results obtained for the problems with $N_s = 5$ are more unstable (Table 5). There was not much improvement with large N_{itr} and N_p and, indeed, a worsening of the standard deviations. These results can be explained if we consider the huge solutions space of the associated problems. In Table 6 we need to actually select large parameter values in order to improve the success rates, and to reduce standard deviations.

4.2. BLH Results

The main purpose of the BLH approach is to obtain admissible solutions in a fast way. The BLH was successful in finding the optimal values for the first and second problems. On the other hand, for the third problem, with the first set of capacities it found a cost value of about 19289.9/year, and 12887.4/year with the second set of capacities case.

Obviously, the BLH approach, being a heuristic, does not always find optimal solutions. If we perform row permutations in routing matrix Δ (responsible for delays evaluation), the BLH will produce different solutions. This cannot be considered a misfunctioning of the heuristic, indeed the fundamental BLH purpose is to satisfy delay constraints and do not minimize costs. Table 7 summarizes the results.

Table 5
Tests for topology 3, traffic 3.a, and $N_s = 5$.

	N_p	N_{itr}	Succ. rate	Best	Average	St. Dev.
1	200	500	125/500	18908.0	19185.2	162.5
2	300	500	132/500	18908.0	19179.8	154.2
3	200	800	131/500	18908.0	19180.0	110.7
4	300	800	148/500	18908.0	19168.9	99.2
5	800	1000	234/500	18908.0	19089.0	84.9

Table 6
Tests for topology 3, traffic 3.a, and $N_s = 10$.

	N_p	N_{itr}	Succ. rate	Best	Average	St. Dev.
1	800	1000	53/500	12605.8	12702.1	149.0
2	1000	2000	75/500	12605.8	12673.1	123.3
3	3000	4000	247/500	12605.8	12612.6	13.9

Table 7
BLH and ES results.

Topol./traff.	BLH	ES
1/1.a	52500	52500
1/1.b	73500	73500
2/2.a	17938.4	17938.4
3/3.a ($N_s = 5$)	19289.9	18908
3/3.a ($N_s = 10$)	12887.4	12605.8

Table 8
Processing times.

	12 Links	24 Links	48 Links
ES	Few seconds	5 h	6 days
PSO	Tenths of seconds	Tenths of seconds	20 s[a], 8 min[b]
BLH	Few seconds	Few seconds	Few seconds

[a] Problem 5 of Table 5.
[b] Problem 3 of Table 6.

4.3. Computational effort

Finally, Table 8 shows the processing times required to optimally solve the presented set of problems. It can be observed that the ES method becomes promptly unfeasible as the size of the problem grows. The BLH approach is faster, but it is expected that, as long as the problem size grows, the gap between BLH and PSO results becomes more prominent. The PSO presents a good tradeoff between processing times and results.

5. Conclusion and future work

We have considered the QoS design of packet networks and, in particular, the Discrete Capacity Assignment problem, where capacity assignments are considered the decision variables. In order to better model the network system, following recent research, we preferred to use a queue with *batch arrivals* instead of the classic $M/M/1$ queue. We have formulated the problem as a combinatorial optimization problem with constraints. A discrete PSO approach was used to obtain feasible solutions, and its performance was compared with an exhaustive search method. Examples of application of the proposed design methodology to different network configurations have been discussed.

The success rate of PSO is more sensitive to the number of particles than the number of iterations. Although the ideal number of particles for the first two problems were between 30 and 50, very good results were obtained even for a quite small number of particles (5–10). Considering a more challenging problem (the third one), larger parameter values needed to be selected in order to improve the success rate. It is important to note that the number of fitness evaluations required by PSO in the worst case is still an extremely small portion of the search space. This fact, by itself, stresses the efficiency of PSO as a heuristic search method. Also, the PSO presents a good scalability, taking into account the three instances of different sizes tested. Due to the NP-completeness of the problem, the exhaustive search method becomes promptly unfeasible as the size of the problem grows.

Overall, the computational results suggest that the PSO algorithm provides a quite efficient approach to obtain (near) optimal solutions with small computational effort. On the other hand, the BLH approach, although faster, will lose its performance as the problem size grows.

Future work will propose an enhancement of the BLH approach by considering multiple random restarts. The idea can be stated as follows: starting with an initial routing matrix Δ, the BLH computes a solution. Then a random row permutation is performed on matrix Δ and the BLH process is restarted. This process continues until no solution improvement can be obtained. The best solution over all iterations is the final result.

Acknowledgments

The authors are grateful to the referees for their valuable comments and suggestions on earlier version of this paper.

References

[1] L. Kleinrock, Queueing Systems, Computer Applications, Vol. II, Wiley Interscience, New York, 1976.
[2] M. Gerla, L. Kleinrock, On the topological design of distributed computer networks, IEEE Trans. Commun. 25 (1977) 48–60.
[3] M. Mellia, A. Carpani, R.L. Cigno, Measuring IP and TCP behavior on edge nodes, IEEE Globecom 2002 (2002).
[4] V. Paxson, S. Floyd, Wide-area traffic: the failure of Poisson modeling, IEEE/ACM Trans. Network. 3 (1995) 226–244.
[5] E.C.G. Wille, M. Mellia, E. Leonardi, M. Ajmone-Marsan, Algorithms for IP networks design with end-to-end QoS constraints, Comput. Networks 50 (2006) 1086–1103.
[6] J.F. Hayes, Modeling and Analysis of Computer Communications Networks, Plenum Press, New York, 1984.
[7] J. Kennedy, R.C. Eberhart, A new optimizer using particle swarm theory, in: Proceedings of the 6th International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[8] K. Maruyama, D.T. Tang, Discrete link capacity and priority assignments in communication networks, IBM J. Res. Develop (1977) 254–263.
[9] C. Ersoy, A. Levi, O. Gumrah, Artificial intelligence search techniques for discrete link capacity assignment in prioritized multiservice networks, Int. J. Comput. Syst. Sci. Eng. (2000) 191–197.
[10] X. Liu, J. Igarashi, H. Miyamoto, A capacity assignment method with different types of channel capacities for packet switched networks, in: 2nd Switching Network Systems Division, NEC Corp., 1990, pp. 194–199.
[11] R. Ferreira, H. Luna, Discrete capacity and flow assignment algorithms with performance guarantee, Comput. Commun. 26 (10) (2003) 1056–1069.
[12] X. Chao, M. Miyazawa, M. Pinedo, Queueing Networks – Customers, Signals and Product Form Solutions, John Wiley, New York, 1999.
[13] M. Garetto, D. Towsley, An efficient technique to analyze the impact of bursty tcp traffic in wide-area networks, Perform. Eval. 65 (2) (2008) 181–202.

[14] C. Fraleigh, F. Tobagi, C. Diot, Provisioning IP backbone networks to support latency sensitive traffic, IEEE Infocom 03 (2003).
[15] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kauffmann, San Francisco, 2001.
[16] H.S. Lopes, L.S. Coelho, Particle swarm optimization with fast local search for the blind traveling salesman problem, in: Proceedings of the 5th International Conference on Hybrid Intelligent Systems, 2005, pp. 245–250.
[17] H.S. Lopes, F. Hembecker, W. Godoy Jr., Particle swarm optimization for the multidimensional knapsack problem, in: Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms, vol. LNCS 4431, 2007, pp. 358–365.
[18] R.C. Eberhart, X. Hu, Y. Shi, Swarm intelligence for permutation optimization: a case study of n-queens problem, in: IEEE Swarm Intelligence Symposium, 2003, pp. 243–246.
[19] E. Yabcznski, Algorithms for solving the discrete capacity assignment problem in TCP/IP networks, MSc. dissertation, Graduate Program on Electrical Engineering and Industrial Informatics (CPGEI), Federal University of Technology – Paraná (UTFPR) - Brazil (in Portuguese), 2007.