9

Routing of Disjoint Backup Paths in ATM Networks

M. Pióro and M. Szcześniak
Institute of Telecommunications,
Warsaw University of Technology
Nowowiejska 15/19, 00-543 Warszawa, Poland
tel: +48 22 259820, e-mail: mpp@tele.pw.edu.pl

Abstract

The paper considers a virtual path restoration mechanism for ATM networks, based on preplanned disjoint backup paths. The purpose of the paper is two fold. First, the network model is introduced and two robust design tasks associated with routing of virtual paths protected with the mechanism are formulated. Then, it is discussed how the tasks can be solved with a stochastic discrete optimisation algorithm called Simulated Allocation. A network design example is reported, and the result of Simulated Allocation is compared with the results of two other methods.

Keywords

ATM networks, ATM cross connect, virtual paths, VP routing, robust design, network dimensioning, discrete optimisation

1 INTRODUCTION

The role of a virtual path (VP) (cf. Sato, Ohta and Tokizawa, 1990 and Fotadar et al., 1995) in carrying traffic in ATM networks can be compared to that of a circuit group in PSTN. A VP is composed of a sequence of VP links, interconnected at intermediate nodes, i.e. at ATM cross connects. While connecting two VP links, a node transfers cells from the incoming link to the outgoing one using the contents of the VP identifier of the incoming cells. A VP link between two nodes is created by assigning a part of capacity (cell rate) of the ATM link (called also "physical link") between the two nodes. Note that in general one ATM link supports many VP

links, and one VP link can belong to only one VP. From the traffic management point of view, an important advantage of the VP mechanism is that creating and cancelling of VPs is achieved by merely adjusting the cell routing tables residing in the memory of the ATM cross connects.

The use of virtual paths in ATM networks is, among other advantages, a potentially powerful means to achieve robustness against equipment failures and traffic shifts. The robustness can be achieved through adjusting the layout of VPs to fit the current network failure/traffic state (cf. Arvidsson, 1995). In the case of a failure of links and/or nodes, one of the possibilities is to restore entirely the broken nominal VPs. A simple restoration mechanism is the protection of nominal paths with single disjoint backup paths (SDBP): for each nominal VP a single link disjoint backup VP of the same capacity is preplanned and used to restore the nominal path in all failure situations in which the protected path is broken (Veitch, Smith and Hawker, 1995). To form the backup paths, spare capacity has to be added to the network links; note that the spare capacity is shared for the restoration purpose among all the failure situations. The use of preplanned single backup paths for the restoration of failed nominal VPs is a relatively simple, flexible, distributed, real time mechanism, potentially effective in coping with failure situations.

In order to use the SDBP mechanism, design algorithms are required for routing the nominal VPs, for routing their backup paths, and for dimensioning ATM links (assigning nominal and spare capacity). One of such tasks is encountered when a set of nominal virtual paths (VPs) of fixed capacity (cell rate) is already routed in the network of ATM links and when the existing nominal capacity (also expressed in cell rate) of the links has to be extended to carry the backup paths in failure situations.

In the paper we consider a general robust design problem (Task 1, Section 2.2) associated with the single backup path mechanism. One of the instances of the problem (Task 2, Section 2.2) is the above mentioned restricted task of finding a set of backup paths minimising the cost of additional protection capacity. After formulating discrete optimisation tasks, in Section 3, we demonstrate how to solve them by means of a stochastic optimisation algorithm of Simulated Allocation (SA) considered by Pióro and Gajowniczek (1995) and by Pióro and Szcześniak (1996). In Section 4 the effectiveness of SA is illustrated with an example of Veitch, Smith and Hawker (1995).

The considered optimisation tasks belong to the class of uncapacitated multicommodity integral flow problems (Assad, 1978) and as such are in general *NP*-complete (Garey and Johnson, 1979).

2 NETWORK MODEL AND PROBLEM FORMULATION

2.1 Network model

The flow networks considered in this paper are modelled by means of a multigraph $G=(V,E,\mathcal{P})$ composed of the set of nodes V, the set of links E, and the incidence re-

lation $\mathcal{P}$ defined on the product $V \times E \times V$ and determining how nodes are linked by links. As a multigraph, G allows for multiple links between the same pair of nodes.

The capacity of link $e \in E$ is denoted by $y(e)$ and expressed in **link capacity units** (LCU); one LCU is equal to M [cells/s]. In general M depends on the technical realisation of the ATM links. If ATM links are realised on SDH (SONET) paths then M is the maximum cell rate provided by one STM-1 synchronous path. The cost of one LCU on link e is denoted by $\xi(e)$, and the constant cost of providing ("opening") the link - by $\kappa(e)$. The total cost of network's links is expressed as:

$$C(y) = \sum_{e \in E} \xi(e) \cdot y(e) + \sum_{e \in E, y(e)>0} \kappa(e), \tag{1}$$

where $y: E \to \mathcal{N}$ is the function assigning capacity to links ($\mathcal{N} = \{ 0,1,... \}$ is the set of nonnegative integers); $\kappa, \xi: E \to \mathbb{R}^+$ are the cost functions ($\mathbb{R}^+$ is the set of nonnegative real numbers).

Only simple paths are considered in graph G (i.e. paths traversing each node at most once), and they are identified with the set of their links: $p = \{ e_1, e_2, ..., e_n \}$. The set of **demands** is denoted by D; each demand $d \in D$ is characterised by the following attributes:

- the pair of its end nodes $(s(d), t(d))$
- the demand volume $n(d)$, expressed in **demand capacity units** (DCU); one DCU is equal to m [cells/s] (in general DCU is much smaller than LCU)
- the set of admissible nominal paths $P(d)$; each path $p \in P(d)$ is a simple path between nodes $s(d)$ and $t(d)$, and can be used to allocate a nominal VP with capacity being a certain portion of the demanded $n(d)$ DCUs
- the set of admissible backup paths $B(d)$ ($B(d) \supseteq P(d)$); each path $p \in B(d)$ is a simple path between nodes $s(d)$ and $t(d)$, and can be used to allocate backup VPs, used to restore the capacity of a nominal VPs in the case of a failure.

Let S denote the considered (given) set of **failure situations** (states). In the paper we consider only link failures, since it is rather easy to generalise the approach to take into account also node failures. Each situation $s \in S$ is characterised by the binary coefficients $\alpha(s,e)$, indicating whether link e is available in situation s ($\alpha(s,e)=1$) or not ($\alpha(s,e)=0$). In the nominal situation $s_0 \in S$ all capacity is available and therefore all coefficients $\alpha(s_0,e)$ are equal to 1.

In order to allocate nominal VPs for demand $d \in D$, it is necessary to specify what portion (number of DCUs) $x_0(d,p)$ of the demand volume $n(d)$ is to be allocated to each admissible nominal VP $p \in P(d)$. The nonnegative integer valued function $x_0: Q \to \mathcal{N}$ (where $Q = \{(d,p): d \in D, p \in P(d)\}$) is called the nominal **allocation function** or the nominal **flow function**, and its values - the nominal **flows**. The allocation functions x_s for failure situations $s \in S \setminus \{s_0\}$ are defined analogously on the domain $Q' = \{(d,p): d \in D, p \in B(d)\}$. For a given flow function x, the quantity

$$r(d,x) = \sum_{p:\, p \in P(d)} x(d,p) \tag{2}$$

is called the **allocation degree** of demand $d \in D$ (expressed in DCUs), and

$$o(e,x) = \sum_{(d,p).\ d \in D, p \in P(d), e \in p} m \cdot x(d,p) \tag{3}$$

is called the **load** of link $e \in E$ (also expressed in cell rate) determined by the flow function x.

2.2 Single disjoint backup path design tasks

In the considered protection mechanism SDBP to each nominal flow $x_0(d,p)$ there is assigned a single protection path $p' \in B(d)$ used to restore the flow in all failure situations affecting (breaking) path p. Therefore all links of the protection path p' have to be available in any failure situation that affects any of the links of the basic path p (path p' may be called failure disjoint with p). Note that if the situations of set S cover failures of all links in the network, then the backup path must be disjoint with p. The protection path p' is used to restore the entire flow $x_0(d,p)$ in all situations which break path p. For pair $(d,p) \in Q$ the formal condition for path p' can be written as follows:

$$p' \in B(d)\ \&\ (\forall s \in S \setminus \{s_0\},\ (\exists e \in p,\ \alpha(s,e)=0) \rightarrow (\forall e' \in p',\ \alpha(s,e')=1)). \tag{4}$$

Task 1: Network design
For given
- demand set D
- failure situation set S and coefficients $\alpha(s,e)$

find
- link capacity function y
- allocation functions x_s for all situations $s \in S$ (including the nominal allocation function x_0 for the nominal state $s_0 \in S$)
- function $\varphi: Q \rightarrow \bigcup_{d \in D} B(d)$ assigning a protection path $p' = \varphi(d,p)$ to each pair $(d,p) \in Q$, satisfying condition (4)

minimising the network cost (1) subject to the constraints
- $\forall s \in S,\ (\forall e \in p,\ \alpha(s,e)=1) \rightarrow x_s(d,p)=x_0(d,p)$, for $d \in D,\ p \in P(d)$ (5)
- $\forall s \in S \setminus \{s_0\},\ (\exists e \in p,\ \alpha(s,e)=0) \rightarrow x_s(d,\varphi(d,p))=x_0(d,p)$, for $d \in D,\ p \in P(d)$ (6)
- $r(d,x_s) = n(d)$, for $d \in D,\ s \in S$ (7)
- $o(e,x_s)) \le \alpha(s,e) \cdot M \cdot y(e)$, for $e \in E,\ s \in S$. (8)

The second considered task consists in designing backup paths and spare link capacity in order to protect existing nominal VPs with the SDBP mechanism. It is a special, important case of Task 1 and more specifically it consists in
(i) finding protection backup paths and
(Ii) extending existing nominal link capacity with the spare capacity for the restoration purpose

for the nominal VPs allocated to existing ATM links, dimensioned for the nominal flow.

Task 2: Protection of nominal paths
The capacity of each link is composed of the existing (nominal) part $y'(e)$ and of the added (protection) part $y(e)$. The minimised function is the cost of the protection capacity (cf.(1)):

$$C(y) = \sum_{e\in E} \xi(e)\cdot y(e) + \sum_{e\in E, y'(e)=0, y(e)>0} \kappa(e), \tag{9}$$

and the modified constraint (8) assumes the form:

- $o(e,x_s) \le \alpha(s,e)\cdot M\cdot(y(e)+y'(e))$, $e\in E$, $s\in S$. (10)

Substituting in the formulation of Task 1 the cost function (9) for (1), and constraint (10) for (8), we arrive at Task 2. In the task, the nominal state s_0 is excluded from the set S. It is assumed that the nominal flows (VPs) are given and that they satisfy the nominal demand ($n(d)$, $d\in D$) only using the nominal capacity $y'(e)$, $e\in E$.

3 THE SIMULATED ALLOCATION METHOD

Simulated Allocation (SA) is a general stochastic combinatorial optimisation method with certain similarities to Simulated Annealing (Johnson et al., 1991). Still, as discussed in Pióro (1997) and in Section 3.4, the two approaches are quite different. Below we describe the idea of SA and its application for Task 1 (finding nominal and backup paths, together with dimensioning the network links). The extension of the SA application to Task 2 is straightforward.

3.1 Application of SA to Task 1

With the SA algorithm, a trajectory of a discrete time stochastic process (called the **allocation process**) is generated through consecutive allocations/disconnections of demand capacity units, one at a time. Let X be the state space of the process. Each state $x\in X$ determines uniquely the current allocation functions for all the situations $s\in S$ and is of the form:

$$x = (x_0(d,p), \varphi(d,p)),\ d\in D,\ p\in P(d), \varphi(d,p)\in B(d)), \tag{11}$$

where $\varphi(d,p)$ denotes the (single) backup path used to restore the nominal flow $x_0(d,p)$ of demand d on path p. Let $T(p)$ denote the subset of those failure situations which affect at least one link of path p. Thus in state x the flow $x_s(d,p)$ for $d\in D$, $p'\in B(d)$ and for failure situation $s\in S\setminus\{s_0\}$ is defined as follows:

$$x_s(d,p') = \sum_{p:\, p'=\varphi(d,p), p\in P(d), s\in T(p)} x_0(d,p) + \sum_{p:\, p'=p, p\in P(d), s\in S\setminus T(p)} x_0(d,p). \tag{12}$$

The symbol $|x|$ denotes the total number of demand capacity units allocated in state x, i.e. $|x|=\sum_{d\in D} r(d,x_0)$. State x is **maximal** if for each demand all of its DCUs are allocated, ie. if $r(d,x_0)=n(d)$ for all $d\in D$ and $|x|=L$, where L is the total number of demanded DCUs .

The time epochs (steps) i=0,1,2,... of the allocation process correspond to consecutive arrivals and disconnections of DCUs (from all demands). Let x^i be the process state in step i (all the components of the state will also be marked with superscript i, eg. the backup path for the nominal flow $x_0^i(d,p)$ will be denoted by $\varphi^i(d,p)$). The capacity of link $e\in E$ in state x^i is defined as the minimal number $y^i(e)$ satisfying the condition

- $o(e,x_s^i) \le \alpha(s,e)\cdot M\cdot y^i(e)$, for all $s\in S$. (13)

When advancing from step i to step i+1, it is determined whether the event in step i+1 will be an arrival of a DCU to be allocated in the network links, or a disconnection (deallocation) of a previously allocated DCU. The probability of the arrival event is in general a function $q(x)$ of the state. Clearly, for all maximal states x we put $q(x)$=0, and for the zero state $q(\mathbf{0})$=1. The probability of a disconnection is 1-$q(x)$. A typical (constant) allocation probability function $q(x)$ ($x\in X$) is defined by $q(x)$=1 for $|x|=\mathbf{0}$, $q(x)$=0 for $|x|=L$, and $q(x)=q_0$=const ($\frac{1}{2}<q_0<1$) for all other states. The state in step i+1, x^{i+1}, is constructed in the following way.

Allocation

Of all the DCUs not allocated in state x^i (there are in total $L-|x^i|$ units not allocated, and for demand d there are $n(d)-r(d,x_0^i)$ of such units) one is selected at random with uniform probability. Suppose a DCU of demand d has been chosen for allocation. Applying a certain **allocation rule** (cf. Section 3.2) one of the paths from the set $P(d)$ is selected (say path p) together with one protection path from the set $B(d)$ (say path $p'=\varphi^{i+1}(d,p)$ with property (4)). The considered DCU is allocated (added) to path p for the nominal situation and for all the failure situations not affecting path p, while path p' is now used to protect the whole nominal flow of path p (extended by 1 DCU with respect to the previous state). Note, that if the selected backup path p' is different from the hitherto assigned backup path $\varphi^i(d,p)$, all the backup flows for flow $x_0^i(d,p)$ defined in step i (i.e. all the flows $x_s^i(d,\varphi^i(d,p))$, $s\in T(p)$) are cancelled, and the whole nominal flow $x_0^{i+1}(d,p)$ is assigned to the new backup path p' in all the failure situations affecting path p. Hence, the new state is defined through:

(i) increasing by 1 the nominal flow $x_0^i(d,p)$
$x_0^{i+1}(d,p) := x_0^i(d,p)+1$

(ii) assigning the selected backup path p' to path p
$\varphi^{i+1}(d,p) := p'$

(iii) extending, if necessary, the capacity $y^{i+1}(e)$ of links on paths p and p' (i.e. for links $e \in p \cup p'$) and reducing, if possible, the capacity $y^{i+1}(e)$ of links on path $\varphi^i(d,p)$ (i.e. for $e \in \varphi^i(d,p)$), according to (13).

Disconnection

Of all DCUs already allocated in state x^i (there are $|x^i|$ such DCUs in total, and $r(d,x_0^i)$ of such units for demand d) one is selected for disconnection at random. Suppose we have chosen a DCU of demand d, allocated to path $p \in P(d)$. The state x^{i+1} is obtained through decreasing the nominal flow $x_0^i(d,p)$ by 1, and through applying the allocation rule to find a (new) protection path p' for p, from the set $B(d)$. Hence, the new state is defined through:

(i) decreasing by 1 the nominal flow $x_0^i(d,p)$
$x_0^{i+1}(d,p) := x_0^i(d,p)-1$

(ii) assigning the selected backup path p' to path p
$\varphi^{i+1}(d,p) := p'$

(iii) reducing, if possible, the capacity $y^{i+1}(e)$ of links on paths p and $\varphi^i(d,p)$, and extending, if necessary, the capacity $y^{i+1}(e)$ of links on path p', according to (13).

Clearly, in each step i, the network cost $C(y^i)$ is computed according to (1).

To start the generated trajectory an initial state x^0 has to be selected. A simple choice is the zero state ($x_s^0(d,p) \equiv 0$); the initial state can alternatively be found by some other design method, eg. by allocating all the nominal demand to the shortest paths, in the sense of the $\xi(e)$ metric. For a stopping criterion we can simple limit the number of steps of the generated trajectory, or assume some number of encountered maximal states. The final SA solution would then be the cheapest maximal state encountered during the simulation.

3.2 Allocation and disconnection rules

A typical (and natural) allocation rule is the selection of a pair (p,p') ($p' = \varphi^{i+1}(d,p)$) with property (4)) from the set $P(d) \times B(d)$, minimising the increase of the incremental cost $\Delta(p,p') = C(x^{i+1}) - C(x^i)$. In the case when the rule does not uniquely determine which path is to be selected, a second order rule can be applied, choosing for instance a nominal path which maximises the total free capacity of links (over all situations) in state x^{i+1} (after allocation of the capacity unit from demand d). Free capacity of a link is defined as the number of DCUs, with which we can additionally load the link without a need of increasing its current capacity.

The effectiveness of SA can be often improved through a modification of the above described allocation rule consisting in adding to the incremental cost $\Delta(p,p')$ the term $c(|x^i|) \cdot |p|$, proportional to the length of the nominal path measured as the number of its links (factor $c(|x^i|)$ is called the **fictitious link cost**). This favours selection of shorter paths. The fictitious cost $c(|x^i|)$ should be a nonincreasing func-

tion of the argument $|x^i|$, tending to 0 with $|x^i| \to L$, and assuming reasonable values with respect to the incremental cost $\Delta(p,p')$.

Another possibility is the use, in states x^i far from maximal, nonmodular real values for the capacity of links:

$$y^i(e) = max \{ o(e,x_s)/M: s \in S \}. \quad (14)$$

While allocating a DCU we can also select demand $d \in D$ in a way different from purely random, eg. assuming that allocation of certain demands has a priority. Similarly, while disconnecting a DCU the choice can be different from random. For instance, often it is advantageous to increase the probability of choosing a DCU allocated to a long path and thus consuming capacity m on several links.

Still another improvement can be achieved by disconnecting in certain states more than one demand capacity unit. This can be done eg. in maximal states, but not necessarily every time a maximal state is reached. Such an action is called an application of a reflecting barrier and it typically consists in disconnecting all nominal flows allocated to links from a randomly selected set of links.

Also, learning allocation/deallocation rules can be considered with decisions depending on the so far generated trajectory. This gives a possibility of detecting, and consequently avoiding erroneous local allocation/deallocation decisions, not advantageous from the global point of view. Here methods developed for Tabu Search could be applied (Glover and Laguna, 1993).

3.3 Incremental link cost

When selecting a nominal allocation path for demand d, it is reasonable to use a predefined set $P(d)$ (static selection) of several short (in terms of the number of links) paths. The backup paths, however, should be the shortest paths rather with respect to the incremental link costs, a notion discussed below. It follows that the backup paths must be found dynamically, since such a metric is state dependent.

When considering a nominal path p from the predefined set $P(d)$ for allocating a DCU, the protecting backup path $p' = \varphi^{i+1}(d,p)$ is found with the Dijkstra shortest (cheapest) path algorithm (cf. Chen, 1990). The cost of allocating one additional DCU on a backup path is the sum of the incremental costs of all links forming the path. The notion of the **incremental link cost**, used as the metric for the shortest path algorithm is described below.

Suppose the generated allocation trajectory is in state x^i and we wish to define the incremental link costs corresponding to a fixed demand $p \in D$ and its nominal path $p \in P(d)$. To do this we hypothetically modify the state x^i to obtain state x' by decreasing the backup flows $x_s^i(d,\varphi^j(d,p))$ for $s \in T(p)$ by $x_0^i(d,p)$, increasing the flow $x_0^i(d,p)$ by 1, and not changing the rest of the flows. This flow modification will in general affect the link capacities and network cost. Denote the modified link capacities by $y'(e)$ ($y^i(e)$ will denote the original capacities, before the state modification).

At this stage the desired incremental link costs are computed and then the shortest backup path for p is found.

Consider link e in the modified state. If we use the link to carry the backup flow $z=x_0^i(d,p)+1=x_0'(d,p)$ in situation $s \in T(p)$, then its capacity $y'(e)$ (computed according to (14)) will have to be increased by

$$\Delta y(e,s,\boldsymbol{x}^i) = max\ \{\ (o(e,x_s')+m\cdot z)/\alpha(s,e) - y'(e)\cdot M),\ 0\} \tag{15}$$

(for $\alpha(s,e)=0$ the increase is infinite). It follows that the incremental cost of link e in allocation state $\boldsymbol{x}^i$ should be defined as

$$\Delta(e,\boldsymbol{x}^i) = max\ \{\ \xi(e)\cdot\Delta y(e,s,\boldsymbol{x}^i)/M:\ s \in T(p)\ \}. \tag{16}$$

For the modular link capacity given by (13), the formula (15) has to be modified:

$$\begin{aligned}
&\Delta y(e,s,\boldsymbol{x}^i) = \lceil\lceil(o(e,x_s')+m\cdot z)/M\rceil - y'(e)\rceil\cdot M, && \text{if } o(e,x_s')+mz > My'(e) \text{ and } \alpha(s,e)=1\\
&\Delta y(e,s,\boldsymbol{x}^i) = 0, && \text{if } o(e,x_s')+mz \le My'(e) \text{ and } \alpha(s,e)=1\\
&\Delta y(e,s,\boldsymbol{x}^i) = \infty, && \text{if } \alpha(s,e)=0,
\end{aligned} \tag{17}$$

and the incremental link cost computed according to (16). Above we have assumed that the constant cost $\kappa(e)$ is 0 for all links. The formulae (15)-(17) can be easily adjusted to the non-zero constant costs case.

It is rather straightforward that if the incremental link cost is taken for the link metric, then the shortest path algorithm will find a backup path which minimises, for given nominal path p, the increase in the total network cost. In Section 4 the static selection of the nominal path and dynamic selection for the backup path will be referred to as Rule 1. A simplification is Rule 2, which selects the backup path also statically, but from the set $B(d)=P(d)$.

3.4 The SA Algorithm

The pseudocode for the SA algorithm is as follows:

```
begin
  step:= 0;
  min_cost:= ∞;
  x:= 0;
  repeat
    step:= step+1;
    if random<q(x) then allocate(x) else disconnect(x);
    if |x|=L and current_cost<min_cost then
      begin min_cost:= current_cost; x_opt:= x end
  until step=step_limit or min_cost=cost_lower_bound
end
```

Identifiers "allocate" and "disconnect" identify the allocation and disconnection procedures, respectively. Both procedures change the current state identified by "x" (initial state is the one with all flows equal to 0). Identifier "step_limit" is the upper bound for the length of the generated trajectory of the allocation process, "cost_lower_bound" is the lower bound for the cost function (1). After termination, the best solution found by the algorithm is stored in variable "x_opt".

In order to estimate the number of steps required to reach a maximal allocation state assume the constant allocation probability function ($q(\boldsymbol{x})=q_0$=const for $0<|\boldsymbol{x}|<L$). It can be shown (cf. Feller, 1961) that the upper bound for the expected number of steps (epochs) of the generated trajectory of the allocation process required to reach the first maximal state from a state with exactly l allocated units is equal to $(L-l)/(2q_0-1)$ (provided $q_0>½$). For instance, starting from the zero state and assuming $q_0=⅔$, the first maximal state will be reached in only $3L$ steps on the average.

A certain similarity of SA to Simulated Annealing (Johnson,1991) maybe noticed, still there are two basic features distinguishing the two approaches:

(i) SA wanders mostly through infeasible points of the optimisation space $\boldsymbol{X}$ (through non-maximal states) and there is an intrinsic mechanism (allocation probability $q(\boldsymbol{x})>½$) forcing the Markov chain to visit feasible solutions (maximal states) sufficiently often. In Simulated Annealing some form a penalty function would have to be introduced in order to force the chain visiting feasible solutions.

(ii) Referring to Simulated Annealing, in SA there is no temperature, piecewise constant in time, decreasing with the evolution of the algorithm. In these terms, in SA such a temperature is state dependent.

A more detailed discussion on relation of SA to Simulated Annealing is presented in Pióro (1997).

3.5 Effective implementation of SA

An effective procedure for generating the allocation trajectory is obtained with the use of the **allocation vector** $A[j]$ (j=1,2,...,L). For the state $\boldsymbol{x}^i$ in time epoch i, the constant length of A is $L=N(i)+M(i)$, where $N(i)$ is the current number of DCUs waiting for allocation, and $M(i)$ is the total number of DCUs already allocated in $\boldsymbol{x}^i$ ($M(i)=\sum_{d\in D} r(d,x_0^i)$). In each state $\boldsymbol{x}^i$ the first $N(i)$ entries of vector A describe the DCUs to be allocated (by specifying for each unit the demand d to which it belongs), and the last $M(i)$ entries describe currently allocated units (by specifying for each unit its demand d and the nominal path $p\in P(d)$).

Suppose the trajectory is in epoch i. If the next event is an arrival of DCU (this happens with probability $q(\boldsymbol{x}^i)$) one of the entries $A[1],A[2],...,A[N(i)]$ is chosen at random, say entry j corresponding to a unit of demand d. Then the allocation rule is applied to choose the nominal path $p\in P(d)$ and its backup path $\varphi^{i+1}(d,p)\in B(d)$, and

the capacity unit is allocated by appropriate adjusting the current flow function and by exchanging the entries *A[j]* and *A[N(i)]* in the allocation vector, writing additionally to *A[N(i)]* the identification of the chosen path *p*. If the next event is a disconnection one entry from the sequence *A[N(i)+1], A[N(i)+2],...,A[L]* is selected at random, a DCU corresponding to it is disconnected, a new backup path for *p* is selected, and, finally, the chosen entry is exchanged with *A[N(i)+1]*.

The above described procedure of the trajectory generation does not require any comparison operations and is therefore very fast. However, it assumes the purely random choice of allocated/deallocated capacity units. If we wish to use a more complicated drawing probability distribution, the procedure will have to be extended (using binary search, event list or a heap for generating events) leading to longer computation times.

4 NUMERICAL EXAMPLE

In this section we present some results obtained with SA for Tasks 1 and 2 for the network example considered by Veitch, Smith and Hawker (1995). The SA results were obtained on a PC/Pentium 60 MHz class computer with a C++ program. In the considered example the failure situations correspond to the entire failures of single links (one situation ≡ one entire link failure). The network consists of 20 nodes, 31 link and 190 demands. There is a single demand capacity unit between each pair of nodes ($n(d) \equiv 1$) and the demand capacity unit m is equal to 1. The modularity of the link capacity unit M is also equal to 1. For each link the marginal cost $\xi(e)$ is equal to 1, and the constant cost $\kappa(e)$ to 0. The detailed description of the example is available from the authors.

The results for Task 2 consisting in routing of protection paths and finding spare link capacity are given in Table 1. The results of SA (first row) are compared with the results of two methods (row two and three) given in Veitch, Smith and Hawker (1995).

Table 1. Results for Task 2

algorithm	nominal cost	protection cost	computation time (*sec*)
SA	570	174	50
heuristic algorithm	570	240	-
Simulated Annealing	570	228	-

Table 2 presents the results for the full design Task 1, illustrating the influence of three variants of the path selection rule on the cost of the nominal and protection capacity, and on the number of VP identifiers (VPI) required in the network for the resulting VP layout. Recall that the fictitious link cost ($c(|x|)$, cf. Section 3.2) penalizes the use of too long VPs. It is independent of the actual capacity taken up by

a VP and is proportional to the number of ATM links traversed by the path. By changing this parameter with respect to the link capacity cost it is possible to influence the lengths of nominal and backup VPs. In the considered network, for the fictitious link cost equal to 1 the nominal flows are realised on the shortest paths. This is reflected in Table 2 by the smallest nominal cost. However, spare capacity cost is higher in this case but, on the other hand, the total number of VP identifiers used is decreased.

Table 2. Influence of different allocation rules in Task 1

	fictitious cost	nominal cost	protection cost	number of VPI
Rule 1	0	650	172	1656
Rule 2	0	670	207	1650
Rule 2	1	570	245	1422

If the fictitious link cost is a decreasing function of the number of allocated DCUs it serves as a guidance for the algorithm to use shorter paths rather than to exploit idle capacity on the longer paths in low allocation states. The unused capacity may then be used by other demands in high allocation states.

Results for Tasks 1 and 2 solved for other (SDH) networks can be found in Pióro and Szcześniak (1996).

5 FINAL REMARKS

In the paper we have discussed the problem of robust VP network design. The application of Simulated Allocation to this problem seems to be quite effective. Of course, SA, as a stochastic approach to combinatorial optimisation, does not guarantee optimal solutions in reasonable time. Nevertheless, the obtained results and their comparison with the results of other methods confirm capability of SA to find, by using simple code, good suboptimal solutions in short computation time, and these are exactly the desired features of a practical engineering method.

Besides, SA allows to take easily into account such additional requirements as:

- aggregation of demand capacity units into jointly allocated modules
- nonlinear cost functions (stepwise, with constant opening cost, etc.)
- nonlinear constraints
- specific limitations on the flow rearrangement (as in Task 1)
- availability of all admissible paths in the search for a solution (dynamic path feature).

The above requirements cannot be directly considered by Linear Programming, even by Mixed Integer Linear Programming. It is important to note that Task 2,

even with relaxed constraints on modularity of VPs and links capacity, is not tractable by Linear Programming (it is tractable with Mixed Integer Linear Programming) and Task 1 is not tractable even with Mixed Integer Linear Programming.

It is also important that due to similarity to the common simulation approach used in performance analysis of telecommunication systems, the SA algorithm is rather easy to understand, control and implement.

Acknowledgements: This paper was prepared under the Polish Research Grant KBN 8T11D00808. Its final version was written during the first author's stay with the Department of Communication Systems, Lund Institute of Technology, Sweden.

6 REFERENCES

Arvidsson, Å. (1994) Management of Reconfigurable Virtual Path Networks, Proceedings of the 14th International Teletraffic Congress, *Teletraffic Science and Engineering*, Vol.1a, Elsevier.

Assad, A.A. (1978) Multicommodity Network Flows - a Survey. *Networks*, Vol.8.

Chen, W-K. (1990) *Theory of Nets: Flows in Networks*. John Wiley & Sons.

Feller, W. (1961) *Introduction to Probability Theory and Its Applications*. Vol.I, John Wiley and Sons.

Fotedar, S., Gerla, M., Crocetti, P. and Fratta, L. (1995) ATM Virtual Private Networks. *Communications of the ACM*, Vol.38, No.2

Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability - A Guide to the Theory of NP-Completness*. W.H.Freeman and Company.

Grover, F. and Laguna, M. (1993) Tabu Search, in *Modern Heuristic Techniques for Combinatorial Problems*, (ed. C.R.Reeves), Blackwell Scientific Publications.

Johnson, D.S., Aragon, C.R., McGeoch, L.A. and Schevon, C. (1991) Optimization by Simulated Annealing: an Experimental Evaluation. *Operations Research*, Vol.39, No.1.

Pióro, M. and Gajowniczek, P. (1995) Stochastic allocation of virtual paths to ATM links, in *Performance Modelling and Evaluation of ATM Networks* (ed. D.D.Kouvatsos), Chapman & Hall.

Pióro, M. and Szcześniak, M. (1996) Robust Design Problems in Telecommunication Networks. *Proceedings of the 10th ITC Specialist Seminar "Control in Communications"*, Lund.

Pióro, M. (1997) Simulation Approach to the Optimisation of Integral Flow Networks. Paper submitted to *International Conference on Optimization and Simulation*, Singapore.

Sato, K-I., Ohta, S. and Tokizawa, I. (1990) Broad-Band ATM Network Architecture Based on Virtual Paths. *IEEE Trans. on Comm.*, vol.38, No.8.

Veitch, P.A., Smith, D.G. and Hawker, I. (1995) A Comparison of Pre-Planned Techniques for Virtual Path Restoration. *Proceedings of the 3rd IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, Ilkley.

7 BIOGRAPHY

Michał Pióro received M.Sc. (1973) in Applied Mathematics, and Ph.D. (1979) and D.Sc. (1990) in Telecommunications, all from Warsaw University of Technology (Poland) where he holds a professor position at the Institute of Telecommunications. His research interests include teletraffic and optimisation theories applied to network modelling, performance analysis and design. He is an author of several tens of technical papers presented in international journals and conference proceedings. He wrote two books: *Design of Non-Hierarchical Circuit Switched Networks with Advanced Routing* (1989, in English) and *Foundations of the Digital Telecommunication Networks Design* (1995, in Polish). For three years since 1984 he had been with the Department of Communication Systems, Lund Institute of Technology (Sweden). For one year since 1990 he had worked as a consultant in traffic routing for Alcatel SESA in Madrid (Spain).

Michał Szcześniak received M.Sc. in Telecommunications from the Warsaw University of Technology (Poland) in 1996. His research interests include modern optimisation algorithms, robust network design and mobile network modelling. He is currently a Ph.D. student at the Institute of Telecommunications, Warsaw University of Technology.