A NEW STOCHASTIC ALGORITHM FOR THE CAPACITY & FLOW ASSIGNMENT PROBLEM AND AN ASSOCIATED RATE OF CONVERGENCE

Bao U. Nguyen

Defence Research and Development Canada – Centre for Operational Research and Analysis

101 Colonel By Drive, Ottawa, Canada K1A 0K2

E-mail: bao.nguyen@drdc-rddc.gc.ca

KEYWORDS

Agents, learning, intelligent systems, optimization, operations research

ABSTRACT

In this paper we examine the problem of optimizing the average time latency of a network using agents that are able to learn. The network design is constrained by a traffic matrix which dedicates specific flows between specific pairs of nodes. Although this is an analysis of an application, we only present two methodologies here, i.e. an algorithm for optimization and a corresponding conservative rate of convergence based on no learning. The application part will be presented in the near future once data is available. We expect the tools developed in this paper can be used to optimize a wide range of objective functions. They will not be limited to optimizing time latency.

BACKGROUND

As part of a Technology Investment Fund project on network-centric warfare (Ng et al. 2006) currently being conducted at the Defence Research and Development Canada Centre for Operational Research and Analysis, we aim to optimize the time latency of a defence network. An example of such a network is the one implemented at Defence Research and Development Canada – Atlantic known as the Networked Underwater Warfare Technology Demonstration Program (LeFrançois 2004). The time latency of a network can be determined in many ways. However, in this paper we describe a methodology designed to optimize time latency using agents that have the ability to learn. The main results of this paper are to provide an algorithm to do so and to derive the rate of convergence of the corresponding algorithm when no learning is in effect. Our methodology draws on (Oommen and Roberts 2000), which provides a heuristic algorithm to optimize a cost objective function.

LEARNING ALGORITHM

We examine a communication network that has a global maximum capacity C. The network flows must satisfy the traffic matrix (γ_{uv}). That is, there will be a dedicated flow from node u to node v that is greater than or equal to γ_{uv}. This ensures that node u can communicate with node v with the desired flow γ_{uv}. In addition, the time latency of the network depends on both the flow and the capacity of each link. We make use of the algorithm in (Oommen and Roberts 2000) which is described below.

The capacities of the links are represented by a vector $(c_1, c_2, ..., c_e)$, where e is the number of edges or links in the network, and c_i is chosen from a finite set of capacities such as $(0,1,2,...)$, where a capacity unit corresponds for example to 1200 *bps* (bits per second). For each link i and each possible capacity j, there is a triplet (I_{ij}, S_{ij}, D_{ij}), where I_{ij} is the probability that the current capacity j of the link i should be increased, S_{ij} is the probability that the current capacity j of the link i should remain unchanged, and D_{ij} is the probability that the current capacity j of the link i should be decreased. The capacity c_i of a link i is modelled as an agent whose

Proceedings 23rd European Conference on Modelling and Simulation ©ECMS Javier Otamendi, Andrzej Bargiela, José Luis Montes, Luis Miguel Doncel Pedrera (Editors)
ISBN: 978-0-9553018-8-9 / ISBN: 978-0-9553018-9-6 (CD)

learning process is encoded in the evolution of the triplet $\left(I_{ij}, S_{ij}, D_{ij}\right)$.

The final solution vector will consist of the capacities c_i such that S_{ij} probability values approach unity, e.g. 0.99. The closer this value is to unity, the more accurate is the solution. As is often the case with heuristic algorithms, (Oommen and Roberts 2000) did not provide the rate of convergence of their algorithm. Fortunately, it is possible to derive the rate of convergence for this algorithm at least in the case where agents do not learn. But before we present the derivation for the rate of convergence, we show below the pseudo code in (Oommen and Roberts 2000). The algorithm can be divided into three modules. Module 1 initializes the triplet $\left(I_{ij}, S_{ij}, D_{ij}\right)$, looks for a feasible solution and determines its value as dictated by the objective function. Module 2 searches the solution space. Module 3 updates the triplet $\left(I_{ij}, S_{ij}, D_{ij}\right)$.

The remainder of the section and the example below illustrate the pseudo code and are taken almost verbatim from (Oommen and Roberts 2000).

Example. Let us assume that the capacity j of a link i has been lowered, such that $j \leftarrow j-1$, resulting in a lower cost feasible solution. This means that we take the following steps.

1. Since the solution vector is feasible, we shall raise the decrease probability D_{ij}, as:

$$I_{ij}(t+1) = \lambda_{R1} \cdot I_{ij}(t)$$
$$S_{ij}(t+1) = \lambda_{R1} \cdot S_{ij}(t)$$
$$D_{ij}(t+1) = 1 - \left(I_{ij}(t) + S_{ij}(t)\right)$$

In general,

a. If the capacity was increased, raise D_{ij}.

b. If the capacity stayed the same, raise S_{ij}.

c. If the capacity was decreased, raise D_{ij}.

2. Now, since the solution vector results in a lower network cost, the stay probability S_{ij} should be raised. This is done as follows:

$$I_{ij}(t+1) = \lambda_{R2} \cdot I_{ij}(t)$$
$$D_{ij}(t+1) = \lambda_{R2} \cdot D_{ij}(t)$$
$$S_{ij}(t+1) = 1 - \left(I_{ij}(t) + D_{ij}(t)\right)$$

In general,

a. If the capacity was increased, raise D_{ij}.

b. If the capacity stayed the same, raise S_{ij}.

c. If the capacity was decreased, raise S_{ij}.

Similar steps are performed for each of the possible scenarios that are examined during the execution of the algorithm. This process continues until all the link capacities c_i have S_{ij} probability values that are close enough to unity.

Module 1. Initialize the triplet $\left(I_{ij}, S_{ij}, D_{ij}\right)$.

For ($i = 1$ to maxlinks(=e))

For ($j = 1$ to maxcaps(=C))

If ($j = 1$ (left-boundary-state))

$$I_{ij} = 1,\ S_{ij} = 0,\ D_{ij} = 0$$

End-If

If ($j =$ maxcaps (right-boundary-state))

$$I_{ij} = 0,\ S_{ij} = 0,\ D_{ij} = 1$$

End-If

If ($1 < j < C$ (internal-state))

$$I_{ij} = 1/3,\ S_{ij} = 1/3,\ D_{ij} = 1/3$$

End-If

End-For

End-For

Repeat

For (i =1 to maxlinks)

$$c_i = RAND(0, \text{maxcaps})$$

End-For

Until (network is feasible)

current-objective = calculate-objective()

For ($i = 1$ to maxlinks)

$$best - c_i = c_i$$

End-For

best-objective = current-objective()

Module 2. Search the solution space.

While (count<num-iterations) **and** (accuracy-level (all links) < required accuracy)

For ($i = 1$ to maxlinks)

$$Action_i = RAND\left(Increase_{ij}, Stay_{ij}, Decrease_{ij}\right)$$

If ($Action_i = Increase_{ij}$)

$$c_i = c_i + 1$$

End-If

If ($Action_i = Decrease_{ij}$)

$$c_i = c_i - 1$$

End-If

current-objective = calculate-objective()

End-For

For ($i = 1$ to maxlinks)

$$j = c_i$$

If (network is feasible)

If ($Action_i = Increase_{ij}$)

$$\textbf{\textit{Raise}}\left(D_{ij}, \lambda_{R1}\right)$$

End-If

If ($Action_i = Stay_{ij}$)

$$\textbf{\textit{Raise}}\left(S_{ij}, \lambda_{R1}\right)$$

End-If

If ($Action_i = Decrease_{ij}$)

$$\textbf{\textit{Raise}}\left(D_{ij}, \lambda_{R1}\right)$$

End-If

Else

Reset all links to best-objective capacities

End-If

If (network is feasible) **and** (current-objective < best-objective)

If ($Action_i = Increase_{ij}$)

$$\textbf{\textit{Raise}}\left(D_{ij}, \lambda_{R2}\right)$$

End-If

If ($Action_i = Stay_{ij}$)

$$\textbf{\textit{Raise}}\left(S_{ij}, \lambda_{R2}\right)$$

End-If

If ($Action_i = Decrease_{ij}$)

$$\textbf{\textit{Raise}}\left(S_{ij}, \lambda_{R2}\right)$$

End-If

For ($i = 1$ to maxlinks)

$$best - c_i = c_i$$

End-For

best-objective = current-objective()

End-If

End-For

End-While

Module 3. Procedure **Raise** – Updating the triplet $\left(I_{ij}, S_{ij}, D_{ij}\right)$. $\lambda_R = \lambda_{R1}$ is associated with a new feasible solution. $\lambda_R = \lambda_{R2}$ is associated with a new feasible solution that is also superior.

If (Action = Increase) $D_{ij} = \lambda_R \cdot D_{ij}$; $S_{ij} = \lambda_R \cdot S_{ij}$; $I_{ij} = 1 - \left(D_{ij} + S_{ij}\right)$

End-If

If (Action = Stay)

$I_{ij} = \lambda_R \cdot I_{ij}$; $D_{ij} = \lambda_R \cdot D_{ij}$; $S_{ij} = 1 - \left(I_{ij} + D_{ij}\right)$

End-If

If (Action = Decrease) $I_{ij} = \lambda_R \cdot I_{ij}$; $S_{ij} = \lambda_R \cdot S_{ij}$; $D_{ij} = 1 - \left(I_{ij} + S_{ij}\right)$

End-If

ALGORITHM EXTENSION

We model the capacity assignment in the same way as that of (Oommen and Roberts 2000) (see above). Hence, there is a triplet $\left(I_{ij}^{(c)}, S_{ij}^{(c)}, D_{ij}^{(c)}\right)$ associated with link i and capacity j, and the superscript c stands for capacity. In addition, we model in a similar way each path l as an agent that carries k units of flow and which connects node u to node v. Each path type agent is represented by a triplet $\left(I_{uvlk}^{(p)}, S_{uvlk}^{(p)}, D_{uvlk}^{(p)}\right)$ where the superscript p stands for path. These triplets are updated at each run depending on random numbers and whether the objective function is improved or not. For example, if the objective function is improved then both $S_{ij}^{(c)}$ and $S_{uvlk}^{(p)}$ are increased. This new algorithm is the first main result of this paper and can be used to optimize the average time latency of a network, as defined in (Mostafa and Eid 2000):

$$T = \frac{1}{\sum_{uv} \gamma_{uv}} \cdot \sum_{k} \frac{f_k}{c_k - f_k} \qquad (1)$$

where (u, v) represents node u and node v; each k represents a link while f_k and c_k are the flow and capacity of that link. We use enumeration to generate all possible paths that connect node u to node v. By modelling each path as an agent, we ensure flow conservation through each node. The flow through a link is then equal to the sum of the flows of all paths that traverse that link. For example, let $\{a, b, c, d, e\}$ be the set of nodes of a complete graph (all possible links) as shown on the LHS of Fig 1. Let's consider the link $(a-b)$; path1 be $(a-b-c)$ with 2 units of flow, path2 be $(a-b-d)$ with 1 unit of flow and path3 be $(c-a-b)$ with 3 units of flow. The flow through the link $(a-b)$ will be the sum of $2+1+3$ as each of the three paths traverse $(a-b)$. Note that the flow of a link ranges from zero to C (the maximum capacity of the network).

RATE OF CONVERGENCE ASSUMING NO LEARNING

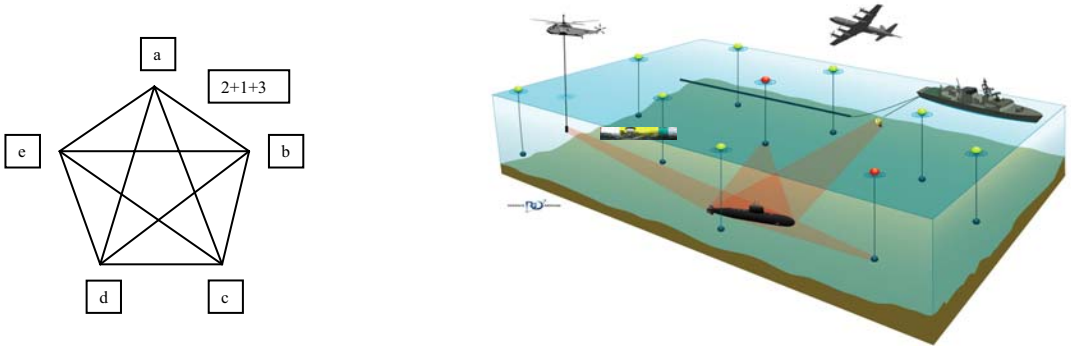

Fig 1 – An example of flows in a graph on the LHS and a defence network on the RHS.

We define the rate of convergence as the probability that a globally optimal state is found at least once as a function of number of iterations in a similar way to that in (Rajasekaran 1990). The rate of convergence that we derive below assumes no learning. Since the purpose of learning is to accelerate the convergence of the algorithm, we expect that this rate of convergence is a conservative estimate of the algorithm. We observe that for a link i, the flow through that link is regulated by a Markov chain as shown in Fig 2. Each flow state is labeled by a number. For example, the label 0 indicates that the flow is equal to zero (unit of flow), while the label 1 indicates that the flow is equal to one (unit of flow) etc. The connections between the labels show the transitions among the states.

The Markov chain shown in Fig 2 can be represented by a transition matrix P where P_{ij} is the probability that state i transitions into state j. For example, given $C = 4$, we get:

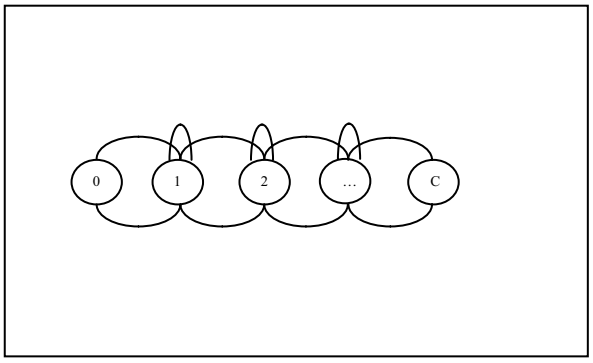

Fig 2 – A Markov chain.

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ C=4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ a & a & a & 0 & 0 \\ 0 & a & a & a & 0 \\ 0 & 0 & a & a & a \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \quad (2)$$

where $a = 1/3$. We model P that way so that the states with flow equal to zero or C are not considered feasible. If a flow of a link is equal to zero then there is no communication necessary. Hence, the probability of transition from state zero to state one is set to 100 %. If a flow of a link is equal to C then there is no capacity left for the remaining links, which can happen only when we have only one link in the network. However, the network that we consider consists of many links. Hence, the probability of transition from state C to state $C-1$ is also set to 100 %. Additionally, for $j = 1,...,C-1$, given link i, $P_{j,j-1}$ (the probability that state j transitions to state $j-1$) is equal to $D_{i,j} = a$; $P_{j,j}$ (the probability that state j stays the same) is equal to $S_{i,j} = a$; and $P_{j,j+1}$ (the probability that state j transitions to state $j+1$) is equal to $I_{i,j} = a$. For example, $P_{1,0} = P_{1,1} = P_{1,2} = a = D_{i,1} = S_{i,1} = I_{i,1}$ while all other transitions from state 1 are forbidden.

Lemma 1. The probability that the flow through a path is optimal is greater than or equal to:

$$p_f \geq 1 - \frac{C-2}{(C-1)^2} \cdot \left|Q_f^n\right| \quad (3)$$

where n is the number of iterations; P_f is the transition matrix associated with the flow of a path; Q_f is equal to P_f with the exceptions that the first row, first column, last row and last column elements are set to zero; and $\left|Q_f{}^n\right|$ is the sum of all elements of $Q_f{}^n$. That is,

$$\left|Q_f{}^n\right| = \sum_i \sum_j \left(Q_f{}^n\right)_{i,j}$$

For example, given $C = 4$, we get:

$$Q_f = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & a & a & 0 & 0 \\ 0 & a & a & a & 0 \\ 0 & 0 & a & a & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \left|Q_f\right| = 7 \cdot a \quad (4)$$

Note that, P_f and Q_f do not change with n. This is so as there is no learning. However, when learning is in effect, the elements of P_f and Q_f are updated through the triplets (I, S, D).

Proof of Lemma 1. To alleviate the notation, we suppress the subscript f associated with P and Q. Let the optimal flow be f^*. We wish to determine the probability that a random and non-optimal flow i transitions to another also random and non-optimal flow j. The probability that we pick a random flow such that $i \neq f^*$ is:

$$P\left(flow = i \neq f^*\right) = \left(C-2\right)/\left(C-1\right)^2$$

For example, given $C = 4$, there are five possible states belonging to $\{0,1,2,3,4\}$. A random flow can be equal to any of these five states. However, if we discard state zero and state four as non-feasible states, there remain only three states. Hence the probability that we pick a random and feasible flow i is $1/\left(C-1\right) = 1/3$. Further, if $f^* = 3$, then the probability that a random and feasible flow is not optimal $\left(C-2\right)/\left(C-1\right) = 2/3$. As a result, the probability that we pick a random flow i and that flow i is feasible and not optimal is $\left(C-2\right)/\left(C-1\right)^2 = \left(1/3\right)\cdot\left(2/3\right)$.

The probability that we start with a feasible flow $i \neq f^*$ and end up with another feasible flow $j \neq f^*$ is regulated by the transition matrix P. That is,

$$\begin{aligned} &P\left(i \neq f^* \to j \neq f^*\right) \\ &\leq P\left(i \neq f^*\right)\cdot P_{i,j} = \left(C-2\right)/\left(C-1\right)^2 \cdot P_{i,j} \end{aligned} \tag{5}$$

The probability that we start with *any* feasible flow $i \neq f^*$ and end up with *any* other feasible flow $j \neq f^*$ is the sum of the above expression over i and j such that $i, j \neq f^*$, i.e.

$$\left(C-2\right)/\left(C-1\right)^2 \cdot \sum_{i,j\neq f^*} P_{ij}$$

We further observe that

$$\sum_{i,j\neq f^*} P_{i,j} \leq \sum_{i,j} P_{i,j}$$

since all the elements of the transition matrix P are non-negative and the LHS sums over all elements i and j such that $i, j \neq f^*$, while the RHS sums over all elements i and j with no restrictions. That is, the sum on the RHS includes more elements of P than the sum on the LHS. Therefore,

$$\begin{aligned} &\left(C-2\right)/\left(C-1\right)^2 \cdot \sum_{i,j\neq f^*} P_{i,j} \\ &\leq \left(C-2\right)/\left(C-1\right)^2 \cdot \sum_{i,j} P_{i,j} \end{aligned}$$

Additionally, the sum on the RHS above runs over the states that are feasible and non-optimal, meaning that $i, j \neq 0, C$. This can be interpreted in a way such that the RHS above includes all transitions from a state i that is feasible to a state j that is also feasible. Therefore, all transitions from (to) 0 or C to (from) a feasible state are forbidden. This is equivalent to replacing P_{ij} with Q_{ij}. Repeating this argument n times, we get an upper bound for the probability q_f of not achieving the optimal state after n iterations:

$$\begin{aligned} q_f &\leq \left(C-2\right)/\left(C-1\right)^2 \cdot \sum_{i,j} {Q^n}_{i,j} \\ &= \left(C-2\right)/\left(C-1\right)^2 \cdot \left|Q^n\right| \end{aligned}$$

Therefore, the lower bound to the probability $p_f = 1 - q_f$ of achieving the optimal state satisfies:

$$p_f \geq 1 - \left(C-2\right)/\left(C-1\right)^2 \cdot \left|Q^n\right|$$

Lemma 2. The probabilistic bound in Eqn (3) of finding the optimal flow is an increasing function of n.

Proof of Lemma 2. This is true as

$$Q^{n+1} = Q^n \cdot Q = Q^n \cdot (P - \Delta)$$

where $\Delta = P - Q$. Simple algebra dictates that

$$\left|Q^n \cdot P\right| = \left|Q^n\right|$$

Hence,

$$\left|Q^{n+1}\right| = \left|Q^n \cdot P\right| - \left|Q^n \cdot \Delta\right| = \left|Q^n\right| - \left|Q^n \cdot \Delta\right|$$

Since $Q^n \cdot \Delta$ has only positive and zero elements, this means that $\left|Q^n \cdot \Delta\right| > 0$. Therefore,

$$\left|Q^{n+1}\right| < \left|Q^n\right|$$

As

$$p_f \geq 1 - (C-2)/(C-1)^2 \cdot \left|Q^n\right|$$

This means that p_f is an increasing function of n. For example, given $C = 4$,

$$\left|Q \cdot \Delta\right| = 4 \cdot a^2$$

Hence,

$$\left|Q^2\right| = \left|Q\right| - 4 \cdot a^2 < \left|Q\right|$$

Lemma 3. It turns out that the probability that the capacity of a link will be optimal is greater than or equal to

$$\begin{aligned} p_c &\geq \min_{c=1,\ldots C} \left\{1 - (c-2)/(c-1)^2 \cdot \left|Q_c^n\right|\right\} \\ &= 1 - (C-2)/(C-1)^2 \cdot \left|Q_C^n\right| \end{aligned} \tag{6}$$

Note that p_c is the probability of achieving the optimal capacity and $Q_C = Q_f$. But if we assume learning, then Q_f and Q_C will change as a function of n, in which case they will not necessarily evolve in the same way. Observe that the capacity of a link must be greater than the flow through that link; otherwise the average time latency shown in Eqn (1) is ill defined. If a flow is j, then the capacity ranges from $j+1$ to C. If we shift the capacity to the left by j, we get c ranging from 1 to $C - j$. Since $j \geq 0$, the largest value for c is C. Since the largest value of c is C and

$$(c-1)/c^2 \cdot \left|Q_{c+1}^n\right| \geq (c-2)/(c-1)^2 \cdot \left|Q_c^n\right|$$

for all c, as proved below, we can assert Lemma 3.

Proof of Lemma 3. We will proceed by induction on n. The first step is to prove that the lemma holds for $n = 1$. That is,

$$(c-1)/c^2 \cdot \left|Q_{c+1}\right| \geq (c-2)/(c-1)^2 \cdot \left|Q_c\right|$$

Since $(c-1)/(c) \geq (c-2)/(c-1)$, we only need to show that $1/c \cdot \left|Q_{c+1}\right| \geq 1/(c-1) \cdot \left|Q_c\right|$. This can be established by observing that $\left|Q_c\right| = (c - 3 + 4/3)$.

Now we assume that this is true for all $k = 1,\ldots,n$ and prove for $n+1$. We will show that this is true when c is odd. A similar proof can be shown when c is even.

Assume that c is odd. Let $Q_c^n = \left[\vec{d}_1^n ; \vec{d}_2^n ; ... ; \vec{d}_{c-1}^n \right]$ where $\vec{d}_i^n$ is the *ith* column of Q_c^n and $Q_{c+1}^n = \left[\vec{e}_1^n ; \vec{e}_2^n ; ... ; \vec{e}_c^n \right]$ where $\vec{e}_i^n$ is the *ith* column of Q_{c+1}^n. Note that we have removed the boundary rows and columns of Q_c^n and Q_{c+1}^n whose elements are 0. This will not affect the proof. Q_c^n obeys a recursion:

$$Q_c^n \cdot Q_c = a \cdot \begin{bmatrix} \vec{d}_1^n + \vec{d}_2^n ; \vec{d}_1^n + \vec{d}_2^n + \vec{d}_3^n ; \\ \vec{d}_2^n + \vec{d}_3^n + \vec{d}_4^n ; ... ; \vec{d}_{c-2}^n + \vec{d}_{c-1}^n \end{bmatrix} \quad (7)$$

Q_{c+1}^n obeys a recursion similar to the one above. Applying recursion and induction repeatedly we get p inequalities where $p = \left\lfloor \frac{c}{2} \right\rfloor$. Each time, the new inequality is obtained by removing the first term and the last term both on the LHS and on the RHS of the previous inequality.

$$\frac{1}{c} \cdot \left(\left| \vec{e}_2^{n-1} \right| + ... + \left| \vec{e}_{c-1}^{n-1} \right| \right) \geq \frac{1}{c-1} \cdot \left(\left| \vec{d}_2^{n-1} \right| + ... + \left| \vec{d}_{c-2}^{n-1} \right| \right)$$

$$\frac{1}{c} \cdot \left(\left| \vec{e}_3^{n-2} \right| + ... + \left| \vec{e}_{c-2}^{n-2} \right| \right) \geq \frac{1}{c-1} \cdot \left(\left| \vec{d}_3^{n-2} \right| + ... + \left| \vec{d}_{c-3}^{n-2} \right| \right)$$

...

$$\frac{1}{c} \cdot \left(\left| \vec{e}_p^{n-p} \right| + \left| \vec{e}_{p+1}^{n-p} \right| + \left| \vec{e}_{p+2}^{n-p} \right| \right) \geq \frac{1}{c-1} \cdot \left(\left| \vec{d}_p^{n-p} \right| + \left| \vec{d}_{p+1}^{n-p} \right| + \left| \vec{d}_{p+2}^{n-p} \right| \right) \quad (8)$$

$$\frac{1}{c} \cdot \left(\left| \vec{e}_p^{n-p-1} \right| + 2 \cdot \left| \vec{e}_{p+1}^{n-p-1} \right| + \left| \vec{e}_{p+2}^{n-p-1} \right| \right) \geq \frac{1}{c-1} \cdot \left(\left| \vec{d}_p^{n-p-1} \right| + \left| \vec{d}_{p+1}^{n-p-1} \right| \right) \quad (9)$$

where $\left| \vec{x} \right|$ is the sum of all elements of the vector $\vec{x}$. The lemma is true if the first inequality is true. But the first inequality is true if the second inequality is true. Repeating the argument tells us that the lemma is true if the last inequality is true. Again, by induction, Eqn (8) is true when we replace $n-p$ with $n-p-1$:

$$\frac{1}{c} \cdot \left(\left| \vec{e}_p^{n-p-1} \right| + \left| \vec{e}_{p+1}^{n-p-1} \right| + \left| \vec{e}_{p+2}^{n-p-1} \right| \right) \geq \frac{1}{c-1} \cdot \left(\left| \vec{d}_p^{n-p-1} \right| + \left| \vec{d}_{p+1}^{n-p-1} \right| + \left| \vec{d}_{p+2}^{n-p-1} \right| \right) \quad (10)$$

Combining Eqns (9) and (10), we get:

$$\frac{1}{c} \cdot \left(\left| \vec{e}_p^{n-p-1} \right| + 2 \cdot \left| \vec{e}_{p+1}^{n-p-1} \right| + \left| \vec{e}_{p+2}^{n-p-1} \right| \right) \geq \frac{1}{c} \cdot \left(\left| \vec{e}_p^{n-p-1} \right| + \left| \vec{e}_{p+1}^{n-p-1} \right| + \left| \vec{e}_{p+2}^{n-p-1} \right| \right) \geq \frac{1}{c-1} \cdot \left(\left| \vec{d}_p^{n-p-1} \right| + \left| \vec{d}_{p+1}^{n-p-1} \right| + \left| \vec{d}_{p+2}^{n-p-1} \right| \right)$$

Hence the proof is complete because we have shown that the inequality (9) is true.

Corollary. Combining the result of Lemma 1 to that of Lemma 3, we obtain the second main result of this paper, the lower bound to the probability of finding the optimal solution in a network which has e links:

$$p \geq \left(1 - \frac{C-2}{(C-1)^2} \cdot \left| Q_C^n \right| \right)^s \cdot \left(1 - \frac{C-2}{(C-1)^2} \cdot \left| Q_C^n \right| \right)^e = \left(1 - \frac{C-2}{(C-1)^2} \cdot \left| Q_C^n \right| \right)^{e+s} \quad (11)$$

Proof of Corollary. The probability of finding the optimal solution is the product of the probabilities that each path carries the optimal flow (provided by Eqn (3)) and the probabilities that each link has the optimal capacity (provided by Eqn (6)). That is, following the inequality sign in Eqn (11), the first factor is the lower probabilistic bound of finding the optimal flows where s is the total number of paths between all pairs of nodes, while the second factor is the lower probabilistic bound of finding the optimal capacities for e links.

Fig 3 below shows the probability of achieving the optimal solution as a function of the number of runs. The *C* value means the flow through each

link ranges from 0 to 20, 0 to 21 or 0 to 22. Fig 3 assumes the network shown on the LHS of Fig 1, i.e. there are ten links $(e=10)$ and ten pairs of nodes. Assume five paths per pair of nodes yields $s=50=10\cdot 5$. Based on Eqn (11), the lower bound for the probability of finding the optimal solution is:

$$p \geq \left(1-(C-2)/(C-1)^2 \cdot \left|Q_C^n\right|\right)^{60}$$

CONCLUSION

We have described a new agent-based algorithm that allows optimizing an objective function that depends both on the flow and the capacity of each link and which satisfies the traffic matrix constraint. In addition, we have derived a novel rate of convergence for this algorithm when assuming no reinforcement learning. We believe that we can further derive the rate of convergence when reinforcement learning is imposed. In a future paper, we will present this more complex derivation and analyze the time latency of an existing network known as the Networked Underwater Warfare Technology Demonstration Program, currently being conducted at Defence Research and Development Canada – Atlantic.

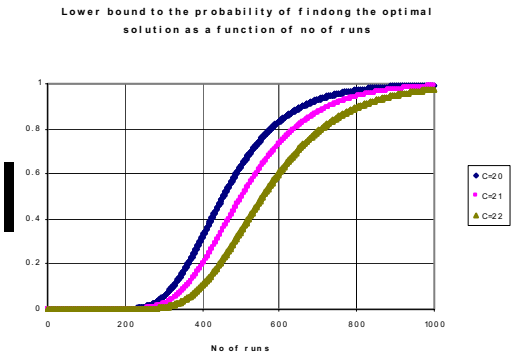

Fig 3 – Probability of achieving the optimal solution as a function of number of runs.

REFERENCES

LeFrançois, M. 2004. "Networked Underwater Warfare", Maritime Warfare Bulletin, Canadian Forces Maritime Warfare Centre, Issue 2004, pp 66-68.

Mostafa, M. E. and Saad M. A. Eid., 2000. "A genetic algorithm for joint optimization of capacity and flow assignment in packet switched networks", 17th national radio science conference, Minufiya University, Egypt, pp C5.1-C5.6.

Ng, K., R. Mitchell and B. U. Nguyen. 2006. Network-enabled operations and network-enabled capability performance metrics, Technology Investment Fund proposal, Defence Research and Development Canada – Centre for Operational Research and Analysis.

Oommen, B. J. and T. Dale Roberts. 2006. "Continuous Learning Automata Solutions to the Capacity Assignment Problem", IEEE Transactions on computers, Vol 49, No 6, pp 608-620.

Rajasekaran, Sanguthevar. 1990 "On the convergence time of simulated annealing", University of Pennsylvania, Department of Computer & Information Science (CIS) Technical Report.

AUTHOR BIOGRAPHY

BAO NGUYEN is a senior Defence Scientist at the Centre for Operational Research and Analysis at Defence Research and Development Canada. He holds a PhD in theoretical particle physics from McGill University, a BSc in physics, and a BSc in mathematics from the University of Ottawa/ California Institute of Technology. He has received many awards including a Gold Medal from the University of Ottawa, a Deputy Minister's Commendation for his work on NORAD long range beacons, and recently a National Practice Prize for the most outstanding application (on autonomous underwater vehicles) of operations research from the Canadian Operational Research Society. Dr. Nguyen has worked in several environments including NORAD & former US Space Command and US Northern Command, NATO Undersea Research Centre, and Defence Research and Development Canada – Atlantic.

ACKNOWLEDGEMENT

The author would like to thank Kevin Ng for bringing (Oommen and Roberts 2000) to his attention.