

A Genetic-Based Fault-Tolerant Routing Strategy for Multiprocessor Networks

Peter K. K. Loh and Venson Shaw

Nanyang Technological University
School of Applied Science
Nanyang Avenue, Singapore 639798
Tel: 65-7991242, Fax: 65-7926559
E-mail: askkloh@ntu.edu.sg

Abstract. We have investigated the adaptation of AI-based search techniques as topology-independent fault-tolerant routing strategies on multiprocessor networks [9]. The results showed that these search techniques are suitable for adaptation, as fault-tolerant routing strategies with the exception that the routes obtained were non-minimal. In this research, we investigate the adaptation of a genetic-heuristic algorithm combination as a fault-tolerant routing strategy. Our results show that such a hybrid strategy results in a viable fault-tolerant routing strategy, which produces minimal or near-minimal routes with a corresponding significant reduction in the number of redundant node traversals. Under certain fault conditions, this new hybrid routing strategy outperforms the purely heuristic ones.

1. Introduction

Much research has been centred on the use of genetic algorithms [2,4,15,16]. Since a genetic algorithm is essentially an optimization technique, many have applied it to applications requiring improvements in their end-results e.g. producing efficient VLSI layouts, enhancing factory automation, network design and reducing transportation problems. Few have applied genetic algorithms to communications to manage routing [11-13]. In [13], a Markov model for an unreliable link is used as part of three heuristic routing table optimisation algorithms. Routing tables are obtained such that the overall network grade-of-service (NGOS) is minimised under different call control rules. These algorithms are then used to demonstrate the improvements in trunk network fault-tolerance (assessed in terms of NGOS) that can be achieved through routing table updating in the event of link (trunk group) failure. In [12], the three algorithms are then used to demonstrate the improvements in trunk network fault-tolerance that can be achieved through network augmentation i.e. the addition of extra links to a network, either in parallel with existing links, or between nodes that previously had no direct connection. In [11], an alternative approach to routing table optimisation is taken, which makes use of a genetic algorithm. The genetic algorithm is applied indirectly as an optimization search technique to achieve dynamic routing control rather than adapted for use directly as a routing strategy. Here, the network

model is assumed to be constant with no mechanism incorporated to tolerate faulty network components. Instead, based on traffic conditions at a given time, the routing control is invoked if the maximum *call loss-rate* (fraction of traffic that a link is unable to carry) exceeds a specified threshold. In this paper, we describe a new hybrid fault-tolerant routing strategy formed by the combination of genetic and heuristic algorithms. Our results show that our new routing strategy produces minimal or near-minimal routes, which reduce significantly the redundant traversals of nodes [3]. In addition, under certain fault patterns, the hybrid strategy outperforms the purely heuristic ones.

2. The Network Model

We can represent, without loss of generality, a multiprocessor network of arbitrary topology as an undirected graph G defined as $G = (N, L)$ where N is the set of nodes and L the set of bidirectional links. If n_i and l_j denote a specific node and a specific link respectively, we have $n_i \in N (i = 0, 1, \dots, |N| - 1)$ and $l_j \in L (i = 0, 1, \dots, |L| - 1)$ where $L = \{ (n_i \rightarrow n_j), (n_j \rightarrow n_i) \mid \forall (n_i, n_j) \in N \}$. Some assumptions made with our network model are network nodes are homogeneous, communication links have identical communication cost, each message fits into a node buffer and each message is transmitted as a complete unit. We adopt the *fail-stop* mode where a network component is either operational or not (faulty). A faulty node is deemed equivalent to having all its links faulty. Routing information in message header is shown in Figure 1.

Control Flag	Destination Node	Pending Nodes	Traversed Nodes
1 bit	6 bits	20 bytes	20 bytes

Figure 1: Format of Message Header

Control Flag- indicates whether through routing or backtracking should be performed.

Destination Node - This field indicates the identity of the destination node.

Pending Nodes - Stack containing nodes that form the route in visitation order.

Traversed Nodes - This is a stack containing visited nodes in the route.

For the software implementation of the current prototype, the length of each field is indicated in Figure 1. However, it is apparent that with this implementation, the header size does not scale well with increase in network size. In an actual implementation, for reduced inter-processor latency, the header can be encoded-decoded and/or compressed-decompressed with additional routing hardware [10].

Modelling Network Routing

A search Θ performed on the problem space S , comprising a set of objectives O and a set of search paths P , may be defined from a given starting objective o_s with a specified target objective o_t as $\Theta(o_s, o_t) : S \times S \rightarrow S^*$, where $S = (O, P)$ and $S^* = (O^*, P^*)$. Since a search path may be traversed both ways and a physical network link supports bi-directional communications, our problem space is a logical representation of a multiprocessor network. An objective in our problem space corresponds to a node while a specific search path corresponds to a physical route in the multiprocessor network. We further define $(o_s, o_t) \in O$ and $(o_s, o_t) \in O^* \Rightarrow$ successful search and $(o_s, o_t) \in O$ and $(o_s \in O^*, o_t \notin O^*) \Rightarrow$ unsuccessful search. In a multiprocessor network, a route \mathfrak{R} from a source node n_s to a destination node n_d may be defined as $\mathfrak{R}(n_s, n_d) : G \times G \rightarrow G^*$, where $G = (N, L)$ and $G^* = (N^*, L^*)$. We can also define: $(n_s, n_d) \in N$ and $(n_s, n_d) \in N^* \Rightarrow$ complete route and $(n_s, n_d) \in N$ and $(n_s \in N^*, n_d \notin N^*) \Rightarrow$ incomplete route. Conceptually, a search is therefore logically equivalent to a routing attempt. A successful search corresponds to a complete route, which includes the source and destination nodes. On the other hand, an unsuccessful search corresponds to an incomplete route, which does not reach the destination node. The definition, however, does not specify the search technique and hence the routing strategy. The definition also does not constrain $\mathfrak{R}(n_s, n_d)$ to be minimal or non-minimal. Finally, fault tolerance is not explicitly specified.

3. Design of Hybrid Routing Strategy

Since a genetic-based strategy is used, we consider first the generation of the initial population of routes. The initial population should contain some *valid* routes (which lead to the destination). However, in a large-scale network, the large number of possible routes between each node pair may exceed memory constraints specified by routing support. To keep the route tables small, the number of initial candidate routes are limited. This limit is defined as the *population size*. The initial population is generated randomly for a fault-free network of N nodes using the following algorithm:

```

for source node 1 to  $N$  //  $N$  is the network size //
  for destination node 1 to  $N$ 
    if current node = destination node, quit
    else
      for route 1 to  $R$  // population of routes //
        current node = start node
        save source node to route table
        counter = 0 // node can't be found //
        do
          select new node randomly
          check whether new node is in route table
          if new node connected, save in route table
          if new node = destination node, quit
          while (no new node found and counter <>  $MAX$ )
        endfor
      endfor
    endfor
  endfor

```

3.1 The Chromosome

The chromosome in the routing strategy represents the route and each gene represents a node in the route. Formally, for any route $\mathfrak{R}(n_i, n_j)$, we have an associated chromosome $\zeta(F)$, where F is the set of genes. Let such a $\mathfrak{R}(n_i, n_j)$ be defined on $G = (N, L)$. Then, for any $n_k \in \mathfrak{R}(n_i, n_j)$, we have an associated gene $g_k \in F$. The length of the chromosome therefore plays an important role in the routing strategy as it affects the communications performance directly. Since the number of nodes to reach the destination varies on different routes, a specific length limit cannot be defined. To maintain unique identities for each gene (node), we can set $\|F\| \leq N$, the network size. If the route involves any backtracking, the path will be longer and the same node(s) will be traversed. Despite backtracking, the distance it travels to the same node(s) will be calculated. The total distance traversed is thus used to compute the fitness value as a selection criterion. The fitness function is discussed in the next sub-section.

3.2 Fitness function

The GA uses the fitness function to perform route selection. In this application, the fitness value is determined by computing the cost of distance travelled to reach the destination. Fitness value for individual route is determined by the following equation:

$$F = \frac{(D_{\max} - D_{\text{route}})}{D_{\max}} + \Delta,$$

where F is the fitness value

D_{\max} is the maximum distance between source and destination nodes in route,

D_{route} is the distance travelled by the specified route, and

Δ is a predefined small positive value to ensure non-zero fitness scores

Thus, the higher the fitness value, the more optimum the route.

3.3 Selection Scheme

We investigate three different selection schemes [17] used to selected the routes:

- *Roulette Wheel Selection*
- *Tournament Selection*
- *Stochastic Remainder*

Each selection scheme employs a different approach. Roulette Wheel Selection is based on the random selection of generating a possible value between 0 and 1, multiplied by the sum of fitness values. The result is called the *roulette value*. The route is then determined by adding the fitness value of each individual route until the sum is equal to or greater than the roulette value. In Tournament Selection, fitness values are compared and the highest selected. Finally, Stochastic Remainder uses probabilities to select the individual route. One of the objectives in our performance analysis is to identify which selection schemes exhibit the best performance in determining optimum route.

3.4 Crossover Operation

Since chromosomes represent solutions to routes, a valid gene sequence of the chromosome (valid route) cannot contain two similar nodes. Different crossover types for permutation, order-based problems are covered in [4,5,17]. Crossover techniques like PMX and Order Crossover will not only destroy the route but also result in invalid source and destination nodes. A secondary constraint is selected routes used to perform crossover operation might contain different route lengths (in terms of number of nodes required to reach the destination). To solve these problems, we use a *one-point random crossover* in our implementation. Figure 2 illustrates this. With this technique, only selected nodes from the crossover point are switched. This means that source and destination nodes can remain intact. If selected route contains only two nodes (source and destination are immediate neighbours), no crossover is performed and specified route is included in next generation if it passes the reroute routine.

					Crossover Point		
					↓		
Route m:	1	2	3	4	5		
Route n: 1	2	7	8	9	10	5	
Route m':	1	2	3	4	9	10	5
Route n':	1	2	7	8	5		

Figure 2: One-Point Random Crossover

3.5 Mutation Operation

Population at this stage may still contain invalid routes. This can be solved by using mutation and reroute operations. Mutation helps maintain diversity of population pool. E.g., in route *m'* of Figure 3, swapping the 5th and 7th nodes will result in a shorter route to the destination. At this stage, although the complete route still needs to be verified for correctness, the source and destination nodes are intact. In the implementation stage, two types of mutation are employed: *swap mutation* and *shift mutation* [1].

3.6 Reroute Operation

REROUTE

```

select 2 routes from old population
perform crossover and mutation
start the repair function
check validity of generated offspring (new route) for:
    • node duplication
    • route validity
    • destination validity
if valid new route, save in new population
else perform backtracking or discard route

```

The hybrid routing strategy is as follows:

```
initialise population
select two parents
perform one-point crossover and swap- or shift-mutation
REROUTE // verify correctness of route //
BACKTRACKING
remove invalid nodes
start from valid node
repeat
    select randomly new node connected to valid node
    if new node selected = destination node
        save route to new population
        quit
until (new node = destination node) or counter = MAX
```

If the counter reaches MAX, the destination is not found. In this case, the new population will contain partial and possibly full routes. The above step is repeated for a number of generations until new population contains optimal solution. In this case, the routes are valid and reach the destination. The backtracking operation enables the message packet to “retract” when it encounters a *deadend* (a node that is almost completely disconnected save for a single link).

4. Performance Analysis

With population generation and fitness evaluation defined and specified, we next identify the combination of selection and exploration techniques exhibit the best performance. Roulette Wheel (RW), Tournament Selection (TS), and Stochastic Remainder (SR) are first analysed, in combination with both *random* reroute (RR) and *hill climbing* reroute (HCR) strategies [6]. A 25-node mesh network is modelled with varying fault percentage. Ten tests were conducted, each over 20 generations with variations in crossover and mutation probabilities. Generation number is determined from observation that fitness values for all selection schemes reach optimal at around 10 generations (Figure 4). More comprehensive testing with different network models and fault percentages [14] showed the same trend. An unsuccessful reroute means that the destination cannot be reached despite existence of valid path. Different crossover and mutation rates influence the number of successful routes. Inherent randomness in RR reduces its suitability for rerouting [6,9] and it has been shown that number of unsuccessful routes increase significantly as generation number increases [14]. With only shift mutation (tests 3, 5 and 9), number of unsuccessful routes may be reduced. Further analysis shows when swap mutation is used with probability rates above 0.5 and 0.2, number of unsuccessful routes increases and decreases respectively [14]. Using crossover rate of 0.2 and mutation rate of 0.1 shows an increase in unsuccessful routes. This indicates that swap mutation must be used with low probability and a lower bound on crossover and mutation probabilities exists. Omitting crossover necessitates raising mutation rates especially for SR as evinced by the routing times.

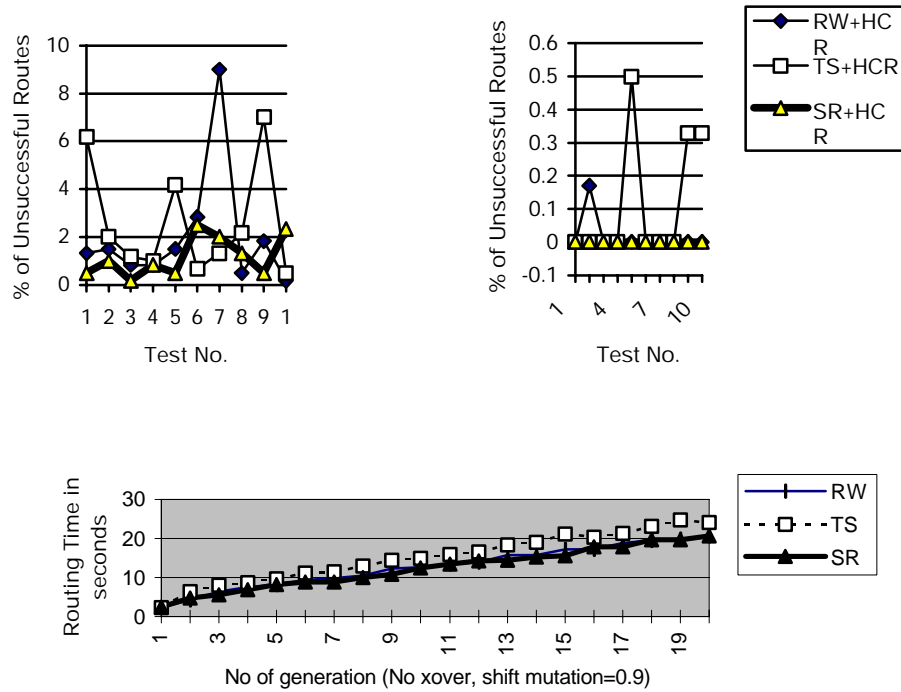


Figure 3 : Performance Comparison of Genetic Selection and Exploration Combinations

SR with HCR clearly outperforms other combinations in having the lowest average percentage of unsuccessful reroutes as well as routing time. With a minimal percentage of unsuccessful routes, SR contributes significantly to the reduction of premature convergence as well as maintaining diversity in the route population. With random one-point crossover, the result does not perform to expectation as it increases the number of unsuccessful routes appreciably (see tests 6 to 8 and 10 for 20 generations). Of 10 tests performed, test 3 provides the best solution using only swap mutation of 0.1. Omitting crossover and having a high shift mutation rate also works favourably as mentioned previously. Mutations rate for both swap and shift mutations should be low (below 0.5) for better results. Higher mutation rates result in more node traversals. However, applying shift mutation with zero crossovers also work favourably. Notably, tests for roulette wheel selection show that applying shift mutation with no crossover, result obtained compared favourably. Tests for TS used tournament size range 2 to 7. Analysis shows that lower rates of mutation perform much better (with exception of omitting crossover which necessitates raising mutation rates). In RW, no comparison is done between between individual routes. It is expected that the individual route selected might not be optimum. TS should provide a better performance as it does comparisons before selection. Overall results incline favourably towards SR selection, with least time required when using shift mutation

alone. In the following tests, we compare performance of the GA-based hybrid strategy with Depth-First Search (DFS), Hill-Climbing (HILL) and Best-First Search (BEST) [8]. A larger mesh model of 49 nodes is used. For 40% fault links, two different tests are run, each with a different fault pattern. A *uniform traffic* model is utilised to filter off variations in message traffic over the network. Under the uniform traffic model, a node sends messages to every other node in the mesh with equal probability. Results in Figure 5 shows that, in several cases, the GA-based hybrid strategy outperforms the heuristic strategies in terms of reduced node traversals and routing times. Penalty is the need for progressive optimisation of route population over 20 generations. Significant performance difference in test case with 40% faults shows that fault patterns has appreciable influence on the routing strategy. Success in determining the routes, especially when deadends are encountered, depend on the reroute routine to accomplish fault-tolerance. Trends indicate that GA-based strategy is reasonably scalable with increasing faults.

5. Conclusion

This research has shown that it is possible to adapt and apply a GA-based search technique as a fault-tolerant routing strategy in a multiprocessor network. To reduce overall communications latency, however, dedicated hardware support is required.

References

1. Buckles,B.P., Petry,F.E., Kuester,R.L., "Schema survival rates and heuristic search in GAs", *2nd Intl Conf Tools Art. Intelligence*, '90, pp322-327.
2. Chen,M. and Zalzala, A.M.S., "Safety considerations in the optimization of paths for mobile robots using genetic algorithms", *Proc. 1st Intl Conf. on Genetic Algorithms in Engring Systems: Innovations and Applications*, 1995, pp 299-306.
3. Dally, J. and Seitz,C.L., Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, *IEEE Tran. Comput.*, VC36, N5, May 87, pp 547-553.
4. Goldberg, D.E., "Genetic and evolutionary algorithms come of age", *Communications of the ACM*, Vol. 37, No. 3, March '94, pp 113-119.
5. Jog, P., Suh, J. Y., and Van Gucht, D., "The Effects of Population Size, Heuristic Crossover and Local Improvement on a GA for Travelling Salesman Problem", *Proc. 1st Intl Conf. Genetic Algorithms and Applications*, July85, pp 110-115.
6. Korf, R.E., *Encyclopedia of Artificial Intelligence*, John Wiley, V2, 1987.
7. Lin,S., Punch, W.F.,Goodman, E.D.,"Coarse-grain parallel genetic algorithms: categorization and new approach", *Proc. Symp Par and Dist Proc*, '94, pp 28-37.
8. Loh, P.K.K., "Heuristic fault-tolerant routing strategies for a multiprocessor network, *Microprocessors and Microsystems*, V19, N10, Dec. 1995, pp 591-597.
9. Loh, P.K.K., "Artificial intelligence search techniques as fault-tolerant routing strategies", *Parallel Computing*, Vol 22, No, 8, October 1996, pp 1127-1147.

10. Loh, P.K.K. and Wahab, A., "A Fault-Tolerant Communications Switch Prototype", *Microelectronics and Reliability*, V37, N8, July 1997, pp 1193-1196.
11. Sinclair, M.C., "Application of a Genetic Algorithm to Trunk Network Routing Table Optimisation", *IEE Proc. 10th UK Teletraffic Symp*, April 93, pp 2/1-2/6.
12. Sinclair, M.C., "Trunk Network Fault-Tolerance through Network Augmentation", *Proc. 4th Bangor Symp on Comms*, Bangor, May92, pp 252-256.
13. Sinclair, M.C., "Trunk Network Fault-Tolerance through Routing Table Updating in the Event of Trunk Failure", *Proc. 9th UK Teletraffic Symp*, Apr92, pp 1-9.
14. Tay, K.P., "An Indestructible Worm", P114-97, NTech. University, 1997.
15. De Jong, K.A. Spears, W.M., and Gordon, D.F., "Using genetic algorithms for concept learning", *Machine Learning*, V13, No. 2-3, Nov-Dec 1993, pp 161-188.
16. Srinivas, M. & Patnaik, L.M., "GAs: a survey", *Computer*, V27N6, Jun'94, pp17-26.
17. Srinivas, M. & Patnaik, L.M., "Adaptive probabilities of crossover and mutation in genetic algorithms", *Tran Sys, Man Cybernetics*, V24 N4, Apr94, pp 656-667.

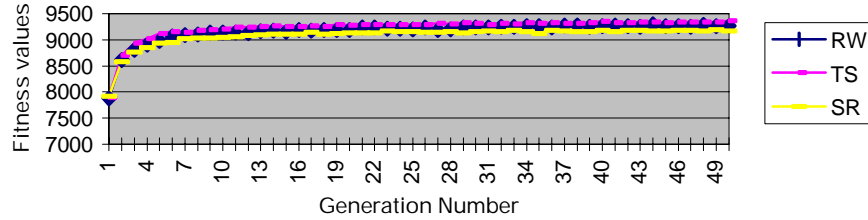


Figure 4: Variation of Fitness Values with Genetic Selection Scheme

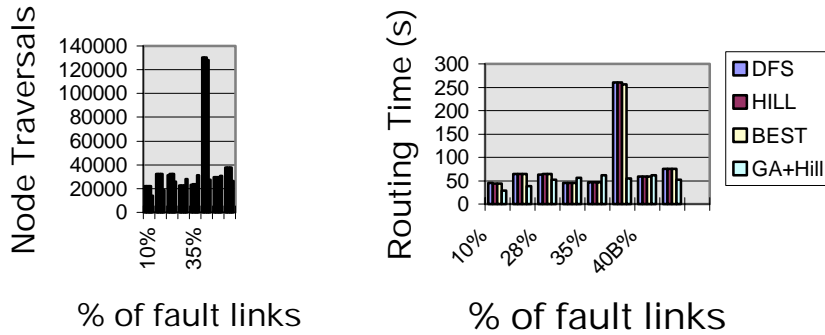


Figure 5: Fault-Tolerant Performance Comparison of Heuristic vs GA-based strategies