Research on network load balancing method based on simulated annealing algorithm and genetic algorithm

Xu Yingzhao, GengQingYang
Institute of Computer, Xi’an Shiyou University, Xi’an, Shaanxi, 710065, China
867309248@qq.com

Abstract. Due to the high burstiness and high real-time performance of modern network load, it is difficult to meet the requirements of purely simulated annealing algorithm or genetic algorithm for network resource utilization and flow control problems, resulting in low network resource utilization. Congestion is serious. In order to make network load balancing more reasonable, a method of combining simulated annealing algorithm with genetic algorithm is proposed. Firstly, the genetic algorithm is used to globally search the network load balancing problem, so that the solution of the problem is quickly in the vicinity of the global optimal region. Then the simulated annealing algorithm is used to further local optimization near the global optimal region, and the optimal solution for the network load balancing problem is found. The simulation results show that the method improves the global search speed, improves the network resource utilization, and significantly improves the network load imbalance.

1. Introduction
With the rapid development of the network, network congestion is becoming more and more serious. In the face of high burst and high real-time network characteristics[1], it is necessary to continuously improve the utilization of network link resources and reduce the probability of network congestion. In response to this problem, scholars have proposed various solutions for network load balancing. The most commonly used network load balancing method is the shortest path routing method[2], but this method makes the network load distribution extremely uneven, and some network nodes are loaded due to frequent processing tasks. Too large, while other network nodes are idle, causing wasted network resources. Therefore, this paper proposes a solution to load balancing problem for the problems of network congestion, low network resource utilization and unbalanced load on modern network links. The method combines the genetic algorithm and the simulated annealing algorithm, and makes full use of the advantages of the simulated annealing algorithm's local search ability and the genetic algorithm to search for the global optimal speed. It forms a network load balancing algorithm and is applied to network load balancing.

2. Introduction of genetic algorithm and simulated annealing algorithm
2.1. Genetic Algorithm
Genetic Algorithm (GA) is a process of heuristic natural selection and belongs to evolutionary Algorithm[3]. Genetic algorithms usually use biological heuristic operators such as mutation, crossover and selection to generate high-quality optimization and search problem solutions[3].
Referring to the theory of biological evolution, genetic algorithm simulates the problem as a biological evolution process, generates the next generation of solutions through selection, crossover, mutation and other operations, and gradually eliminates the solutions with low fitness function value and increases the solutions with high fitness function\[4\]. In this way, after N generations of evolution, individuals with high fitness function value are likely to evolve. The following is the process of genetic algorithm:

1. In the initial stage of the algorithm, a set of feasible solutions, namely the first generation chromosomes, are generated randomly.
2. Fitness function was used to calculate the fitness of each chromosome, and the probability of each chromosome being selected in the next evolution was calculated according to the fitness.
3. By "crossing over," you make some chromosomes.
4. The chromosomes produced by crossing over perform the mutation operation.
5. Chromosomes with a high fitness value in the parent generation are inherited by "replication".
6. Repeat step (2) until the optimal solution is satisfied.

2.2. Simulated annealing algorithm
Simulated Annealing algorithm (SA) is a heuristic algorithm with strong local search ability. It Simulated the Annealing process of solids, which is suitable for solving large-scale problems with combination and optimization\[1\]. In general, SA uses the Metropolis criterion to accept the probability p of the optimal solution, which can be expressed as the following formula:

\[
p = \begin{cases} 
1, & f(i) \leq f(i') \\
\exp \left( \frac{f(i) - f(i')}{kT} \right), & f(i) > f(i')
\end{cases}
\]  

(1)

And t represents the control parameter, f(i) represents the objective function value of the problem, and f(i') represents the objective function value under the new state. Suppose the temperature at time t is represented by T(t), then the simulated cooling mode of SA is as follows:

\[
T(t) = \frac{T_0}{\ln(1 + t)}
\]  

(2)

Due to the GA and SA have different advantages and disadvantages when solving the load problem, GA has strong global search ability, but poor local optimization ability. SA has strong local optimization ability and slow convergence speed after search. Therefore, this paper combines GA and SA to form a combination optimization algorithm with better performance -- simulated annealing genetic algorithm. It combines the advantages of the two algorithms and overcomes the disadvantages of the single algorithm.

3. Modeling of network load balancing problem
In general, the network topologic structure is represented by digraph. In this paper, the following network topologic structure G=(V,E,C) is used to describe the network topology. V is the convergence of network nodes, E is the convergence of network links, and C is the maximum capacity of network links and other constraints\[5\].

Fig.1 Network topologic structure
If set $M$ is used to represent the load in the network, then for any $m$ and $m \in M$, there is $(S_m, T_m, y_m)$, where $S_m$ represents the entry network node, $T_m$ represents the exit network node, $y_m$ represents the link bandwidth requirement of $(S_m, T_m)$. $X^m_{ij}$ represents whether the load $M$ passes through the link $(i, j), (i, j) \in E$, and $h^m$ represents the routing hops of the link. So

$$X^m_{ij} = \begin{cases} 0, & \text{No link (i, j)} \\ 1, & \text{Over link (i, j)} \end{cases} \quad (3)$$

The objective of network load balancing optimization in this paper is to make the network link utilization more balanced and the network nodes on the network link have the shortest processing time. At the same time, the network traffic on the link with heavy load is transferred to the link with light load. Thus, the load on each link is relatively balanced and the processing speed is fast, reducing the probability of network congestion. The mathematical model of network load optimization in this paper is as follows:

Let $\alpha$ be the maximum utilization rate of network link, $C_{ij}$ represents the capacity of link $(i, j)$, namely, the maximum number of tasks on network link, and the task length of all tasks on network link $(i, j)$ is represented by matrix $T_{ij}$. Suppose that there are four tasks running on the link $(V_5, V_7)$ in Fig.1, the length of task 1 is 2, the length of task 2 is 4, the length of task 3 is 6, and the length of task 4 is 8, then $T_{ij} = \{2, 4, 6, 8\}$. $K$ in $T_{ij}[k]$ represents the task number, while $T_{ij}[k]$ represents the task length of task $k$. The processing speed of all server nodes on the network link $(i, j)$ is expressed by matrix $N_{ij}[s]$. The link $(V_1, V_2)$ in Fig.1 contains four network nodes. Assuming that the processing speed of $V_1$ node is 2, that of $V_2$ node is 1, that of $V_3$ node is 2, and that of $V_4$ node is 3, then $N_{ij}[s] = \{2, 1, 2, 3\}$. $s$ in $N_{ij}[s]$ represents the number of nodes, while $N_{ij}[s]$ represents the processing speed of node $s$.

After the task matrix $T_{ij}[k]$ and node matrix $N_{ij}[s]$ are determined, the task processing time of all tasks assigned to all nodes on the link $(i, j)$ is represented by the matrix $time_{ij}[k][s]$, $time_{ij}[k][s]$ represents the time required to allocate task $k$ to nodes for processing, and its calculation formula is as follows:

$$time_{ij}[k][s] = \frac{T_{ij}[k]}{N_{ij}[s]} \quad (4)$$

Therefore, the network load balancing objective function and constraint conditions to be optimized are defined as follows:

The objective function: $f = \min \alpha + \min time[k][s] \quad (5)$

The requirement of constraint:
\[
\begin{split}
X^m_{ij} - X^m_{ji} &= 0 \quad m \in M, i \neq S_m, j \neq T_m \\
X^m_{ij} - X^m_{ji} &= 1 \quad m \in M, i = S_m \\
X^m_{ij} - X^m_{ji} &= -1 \quad m \in M, i = T_m \\
X^m_{ij} \cdot y_m &\leq C_{ij} \cdot \alpha \quad (i, j) \in E \\
X^m_{ij} &\leq h_m \quad m \in M \\
X^m_{ij} \in (i, j) &\quad \alpha \geq 0
\end{split}
\] (6)

4. Simulated annealing genetic algorithm (SA) is used to optimize the load balancing problem

According to the above equation (5) and (6), network load balancing optimization is under the constraint of the above equation (6), selected from multiple paths an optimal path, if uses the shortest path routing method, time consuming and difficult to obtain the optimal solution, so this article uses the method of combining the simulated annealing algorithm and genetic algorithm to solve. In view of the randomness of high task concurrency on the network link, this paper adopts the random natural number encoding method. Since the network structure is known, the number of network links is known. And each link is a chromosome in the genetic algorithm, so the length of chromosome encoding is determined, and its value is the same as the total number of links between network nodes. In the simulation network structure diagram in Fig.2 below, the length of chromosome encoding is 22. First, each route is coded from 1, so that the routing code arrangement is a feasible solution to the original problem, that is, a chromosome in the genetic algorithm. In Fig.2, AH is a link in the network, and \{1,2,7\} is a chromosome according to coding rules.

![Fig.2 Simulation network structure diagram](image)

After the coding is completed, simulated annealing genetic algorithm is used to solve the problem. Firstly, genetic algorithm is used to solve the optimal solution interval of the load problem, which requires setting parameters, initializing the population, determining the fitness function, genetic operation, etc. Secondly, on the basis of genetic algorithm, simulated annealing algorithm is used to search the local optimal solution in the optimal solution interval. The specific steps are as follows:
(1) set initial parameters

Let the initial population number of genetic algorithm adopted be 10, and the convergence condition is that formula (5) above satisfies \( f \leq 0.8 \), set \( T_0 = 0.8 \) as the initial fire temperature of the simulated annealing algorithm, where the link crossover probability and mutation probability in the genetic algorithm are shown as follows:

\[
pv = pc = \begin{cases} 
0 & \text{The fitness value of the link } (i,j) \geq \text{Average fitness} \\
1 & \text{The fitness value of the link } (i,j) < \text{Average fitness}
\end{cases}
\] (7)

Where \( pv \) is the mutation probability and \( pc \) is the crossover probability.

(2) initialize the population

In this paper, the initial population is generated in a random manner. If the maximum routing number of the network feasible routing set \( Q_1 \) is set as \( N_1 \), then a random number from 1 to \( N_1 \) is generated to determine the \( y_1 \) in the \((y_1, y_2, y_3, \ldots, y_m)\) of each individual in the initial population. As shown in Fig.2, each link in the network structure is a feasible route, and the maximum route number is the number of passable links in the network structure diagram, from which 10 links are randomly selected to form the initial population.

(3) determine the fitness function

Since the network load problem is a process of finding the optimal solution of the objective function under multiple constraints[6], the optimization objective designed in this paper is that the load on the network link is relatively balanced and the processing time of the network node is the shortest. Therefore, this paper adopts the following fitness function to calculate the advantages and disadvantages of each link.

\[
f(x) = \frac{1}{\max \alpha_{ij} + \max \text{time}_{ij}[k][s]} \quad (i,j) \in E \quad s \in V \quad k \leq C_{ij} \] (8)

(4) genetic manipulation

After the fitness function is determined, the fitness values of the links in the initial population are respectively calculated, while the links with low fitness values need to evolve through genetic manipulation to generate new links and replace the parent links to generate new populations. GA genetic manipulation is generally through selection, crossover and mutation. The specific process is as follows:

1) selection: the links with higher fitness value in the parent population are directly copied to the next generation population, while other links in the child population are firstly selected for cross operation by the selected links. In the initial population of links, links with low fitness are selected as follows:

The probability of link \((i,j)\) being selected = the fitness of link \((i,j)\) / the sum of the fitness of all links.

2) crossover: after the selection operation, the probability of the link crossover is calculated according to the above formula (7). If \( pc > 0.5 \) is crossed, the link will be disconnected from a network node in the link and recombined into a new link; otherwise, the mutation operation will be carried out.

3) Mutation: after cross operation, calculate the probability of the link to carry out mutation operation. If \( pv > 0.5 \), then carry out mutation operation. Select a node randomly in the link and change it into other nodes to generate a new link. Otherwise, the fitness value of the link is calculated by directly replacing the chromosome with low fitness of the parent generation.

(5) Perform annealing operations

Let the initial temperature of annealing be \( o=0.8 \). According to Meteopolis criteria, it is necessary to judge whether to accept the current link as the optimal solution according to the results of formula (1), in which \( f(i) \) is the fitness function value of the current link and \( f(i') \) is the fitness function value in the new state. If the fitness value of the link in the new state is higher than that of the link in the current state, the link in the new state is accepted, making \( t=t+1 \); otherwise, the link in the new state is accepted with a certain probability, and the link with the lowest fitness in the population is replaced with the new link currently generated. The specific algorithm flow is shown in Fig.3.
5. Simulation experiment and performance analysis

In order to detect the performance of simulated annealing genetic algorithm in dealing with network load problem, the simulation network structure diagram in Fig.2 was used for load balancing experiment in this paper. Select 14 network nodes and set the maximum link hops as 6, that is, each link has a maximum of 7 network nodes, each link capacity is 155[7], and each task length is 32.

In the simulation network structure diagram in Fig.2, it is assumed that the flow constraint conditions of each link are shown in table 1, and the CPU performance of each network node is shown in table 2.

| Link | BM | AH | AN | AL | CM |
|------|----|----|----|----|----|
| flow constraint | 33 | 27 | 32 | 40 | 34 |

| Link | MH | DI | DN | DM | HI |
|------|----|----|----|----|----|
| flow constraint | 31 | 48 | 21 | 25 | 15 |

| Network node | A | B | C | E | F | G |
|--------------|---|---|---|---|---|---|
| CPU performance(GHz) | 2.8 | 2.5 | 2.0 | 2.8 | 2.8 | 2.0 |

| Network node | H | I | J | L | M | N |
|--------------|---|---|---|---|---|---|
| CPU performance(GHz) | 1.5 | 1.8 | 2.3 | 2.8 | 3.0 | 1.5 |

In the above simulation network environment, this paper compared the genetic algorithm, shortest path algorithm and simulated annealing genetic algorithm in handling network load, conducted 100 experiments, and finally took the average value as the final result. Its performance is shown in table 3, and the scheduling results of simulated annealing genetic algorithm are shown in Fig.4.

| algorithm | Maximum link utilization $\alpha$ | The convergence algebra |
|-----------|---------------------------------|-------------------------|
| Simulated annealing genetic algorithm | 0.525 | 100 |
| Genetic algorithm | 0.653 | 200 |
| Shortest path algorithm | 0.715 | 500 |
As can be seen from table 3, when using the shortest path algorithm to deal with the network load, the maximum link utilization rate $\alpha = 0.715$, the link utilization rate is large, the load is unbalanced, there is no load on some links in the network, and the network resource utilization rate is low, which is easy to cause network congestion. Compared with shortest path algorithm, genetic algorithm has improved performance, but it has higher computational complexity and slower convergence. Simulated annealing genetic algorithm has the best performance, with the maximum link utilization rate $\alpha = 0.525$, which makes each link load relatively balanced, network congestion probability reduced, and network resource utilization rate better. The convergence speed is faster than genetic algorithm.

As can be seen from figure 4, when the simulated annealing genetic algorithm evolves to $T=20$, the task processing time of the network node gradually converges, and when $T=40$, it already converges.

6. Conclusion
In view of the network load balancing problem under the characteristics of modern networks, the single genetic algorithm or simulated annealing algorithm has the disadvantages of falling into the local optimal solution and slow convergence speed when solving the network load. In this paper, genetic algorithm and simulated annealing algorithm are combined to optimize network load. Considering the processing time of network nodes and other factors, further optimization of network load can well prevent the search process from falling into the local optimal solution and other advantages. In addition, through experiments in the simulation environment, the performance of solving network load is compared with other methods. The results show that the simulated annealing genetic algorithm has the shortest time, faster convergence and better average performance than the genetic algorithm and the shortest path algorithm. Effectively reduce the probability of network congestion, make the network load relatively balanced, and improve the network resource utilization. Therefore, the simulated annealing genetic algorithm is a feasible algorithm to solve the problem of network load balancing.

Acknowledgments
We are grateful to the National Natural Science Foundation China Project for the financial supported under Grant No.51574194 for this paper.

References
[1] Tan shenglan. Application of simulated annealing genetic algorithm in network load balancing [D]. Dongguan university of technology, 2011. 111-114 (in Chinese).
[2] Cao LAN. Application of genetic algorithm in load balancing system [J]. Journal of sichuan university of science and technology, 2011 (01). 83-84 (in Chinese).

[3] Liu congjun. Application research of load balancing mechanism based on improved genetic algorithm [D]. Jiangsu university of science and technology, 2018 (08). 115-119 (in Chinese).

[4] Wang chao. Application of genetic algorithm based on load balancing in network resource scheduling [J] 2012 (6): 36-34 (in Chinese).

[5] Zhang s f, wu g x. asymmetric DHT and load balancing in P2P networks [J], journal of communications, 2007(9). 60-67 (in Chinese).

[6] Yang Chunyong, Chen Shaoping. Network load balancing routing optimization based on improved ant colony algorithm [J]. Computer engineering, 2010, 36 (8): 4-7 (in Chinese).

[7] X P Xiao. Traffic engineering with MPLS in the Internet [J]. IEEE Networking, 2000, 14 (2): 28-33.