Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A hybrid genetic algorithm to optimize device allocation in industrial Ethernet networks with real-time constraints*

Lei ZHANG[1], Mattias LAMPE[2], Zhi WANG‡[1]

([1]*State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China*)

([2]*Corporate Technology, Siemens Ltd., Beijing 100102, China*)

E-mail: zhangleijz@gmail.com; mattias.lampe@siemens.com; wangzhi@iipc.zju.edu.cn

Received Feb. 28, 2011; Revision accepted Aug. 1, 2011; Crosschecked Nov. 4, 2011

Abstract: With the advance of automation technology, the scale of industrial communication networks at field level is growing. Guaranteeing real-time performance of these networks is therefore becoming an increasingly difficult task. This paper addresses the optimization of device allocation in industrial Ethernet networks with real-time constraints (DAIEN-RC). Considering the inherent diversity of real-time requirements of typical industrial applications, a novel optimization criterion based on relative delay is proposed. A hybrid genetic algorithm incorporating a reduced variable neighborhood search (GA-rVNS) is developed for DAIEN-RC. Experimental results show that the proposed novel scheme achieves a superior performance compared to existing schemes, especially for large scale industrial networks.

Key words: Optimization, Real-time, Industrial Ethernet, Device allocation, Steady-state genetic algorithm, Variable neighborhood search

doi:10.1631/jzus.C1100045 **Document code:** A **CLC number:** TP393.11

1 Introduction

Industrial automation systems rely on the exchange of control and feedback messages between devices through a local communication network. In recent years industrial Ethernet has become an attractive candidate for such industrial networks as an alternative or complement to traditional fieldbuses, e.g., Controller Area Network (CAN) and Profibus. A considerable number of Ethernet based industrial communication standards, e.g., Ethernet for Plant Automation (EPA), Ethernet IP, and Profinet, have been defined in IEC 61784-2 (2005).

An important design target for industrial networks is to satisfy the timing requirements of the field devices, e.g., programmable logic controllers (PLCs), PC based controllers, sensors, and actuators. As the device number for factory automation systems is steadily growing and can range into tens of thousands, industrial networks are becoming more and more complex (Felser, 2005; Kjellsson *et al*., 2009). Guaranteeing real-time performance of these networks is therefore becoming an increasingly difficult task in the design process.

Nowadays network design is mostly done manually by experienced engineers. However, with increasing network complexity, engineering rules based on experience gained from smaller projects will not always lead to a cost-efficient solution for such large-scale projects. Identifying possible performance bottlenecks in manually planned networks is also a challenging task. Therefore, the development of suitable optimization algorithms and software tools to assist the design of industrial Ethernet networks is an increasingly important topic.

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 60873223 and 90818010), the State Key Laboratory of Industrial Control Technology (Nos. ICT0903, ICT1003, and ICT1103), and the Key Laboratory of Wireless Sensor Network & Communication of Chinese Academy of Sciences (No. 2011001)

The design of an industrial Ethernet network can be performed in two steps, designing a network topology and subsequently allocating devices to the individual switches (Elbaum and Sidi, 1996; Gen *et al*., 2008). Although network switches currently available for industrial Ethernet provide basic quality of service (QoS) capabilities, such as priority based scheduling (IEEE 802.1Q, 2003), they do not support more complex mechanisms, like dynamic QoS-based routing. Thus, careful optimization of network design at a physical level provides a static method to achieve a good real-time performance of the network without extra cost. However, in industrial applications the network topology typically cannot be chosen freely but is subject to many practical constraints such as cost and reliability considerations. In this case, the real-time performance of an industrial Ethernet network depends strongly on the way the devices are allocated to the individual switches in the network. Therefore, the problem of device allocation in industrial Ethernet networks with real-time constraints (DAIEN-RC) remains an important topic.

To obtain the best performance of DAIEN-RC, it is important to jointly consider the optimization criteria suitable for industrial applications and the optimization algorithm operating on them. Recently numerous publications have addressed the DAIEN-RC problem based on this holistic consideration. The DAIEN problem was mathematically modeled as a *k*-way graph partitioning problem in Krommenacker *et al.* (2002). This problem was further analyzed and formulated as a bi-objective optimization problem in Zhang and Zhang (2007). Pure genetic algorithms (pGA) were applied in Krommenacker *et al.* (2002) and Zhang and Zhang (2007). A switch-device encoding was proposed in Carro-Calvo *et al*. (2010) to improve the efficiency of pGA. A graph partitioning strategy was developed in Li *et al.* (2007) for the problem. The objective of the aforementioned references was to minimize the traffic between subnetworks and cause the traffic to be evenly distributed in the network, which could implicitly improve the average delay of the overall network. However, this approach cannot adequately deal with the stringent and diverse real-time requirements of typical industrial applications. Instead of globally optimizing average delay, it is more meaningful in industrial applications to consider the potentially different real-time requirements of the individual connections between the devices. Towards this aim, Georges *et al*. (2006) developed a delay model to determine delay bounds in switched networks, and evaluated if end-to-end delays were below their real-time requirements in the objective function. The optimization was performed using a pGA.

It must be pointed out that explicit optimization of the end-to-end delay distribution greatly increases the difficulty of optimization as compared to the traffic optimization discussed in Krommenacker *et al.* (2002), Georges *et al.* (2006), Li *et al.* (2007), Zhang and Zhang (2007), and Carro-Calvo *et al*. (2010). Algorithms known from graph partitioning as described in Li *et al.* (2007) are no longer applicable since the computation of the end-to-end delay of any communication pair requires knowledge of the complete device allocation plan.

pGAs as in Krommenacker *et al.* (2002), Georges *et al.* (2006), Zhang and Zhang (2007), and Carro-Calvo *et al*. (2010), on the other hand, have their limitations if the search space becomes very large. As mentioned before, the size of industrial communication networks increases, and thus the search space for DAIEN-RC can become very large. The difficulty of pGA in dealing with such large scale problems provides the motivation to propose a new approach particularly suitable for dealing with large scale networks.

In this paper, a novel scheme is proposed for explicitly optimizing the end-to-end delay distribution in large scale Ethernet networks. First an improved objective function representing the optimization criterion is proposed that describes the real-time requirements of typical industrial applications. Second, a hybrid genetic algorithm with reduced variable neighborhood search (GA-rVNS) is developed with the aim to find a good device allocation within a reasonable computation time. A steady state GA (ssGA) is used to explore the entire search space and to find promising areas, whereas rVNS exploits a given search region. A novel strategy for guiding the search is applied to increase the efficiency of rVNS.

2 Problem statement

Real-time requirements of industrial automation applications are characterized by the maximal

tolerable delays of all traffic flows between devices. The goal of DAIEN-RC is to guarantee that the end-to-end delays of all traffic flows stay below their maximal tolerable limits by optimizing the allocations of devices in the network. An example of optimizing DAIEN-RC is shown in Fig. 1.

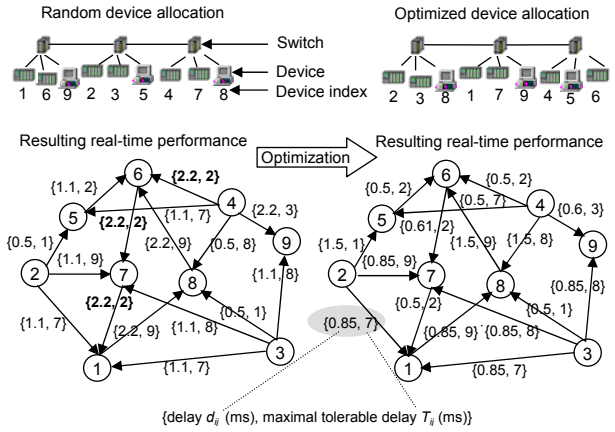

Fig. 1 An illustrative example of optimizing device allocation in industrial Ethernet networks with real-time constraints (DAIEN-RC)

Nine devices are connected to three switches arranged in a linear topology. The set {delay, maximal tolerable delay} is used to represent the real-time performance and requirement of each traffic flow. With a random device allocation three highlighted traffic flows violate their delay limits, while no violation is found after the optimization

N devices are assumed to be connected to M switches in an industrial Ethernet network. The traffic flows between the devices are denoted by an $N\times N$ matrix $\boldsymbol{C}$ as

$$\boldsymbol{C}=\begin{bmatrix} 0 & c_{12} & \cdots & c_{1N} \\ c_{21} & 0 & \cdots & c_{2N} \\ \vdots & \vdots & & \vdots \\ c_{N1} & c_{N2} & \cdots & 0 \end{bmatrix}, \tag{1}$$

where the element c_{ij} represents a traffic flow from device i to j. c_{ij} can be described by the average data rate r_{ij} and maximum frame size b_{ij}. For each traffic stream c_{ij}, let T_{ij} be the maximal tolerable delay, and d_{ij} be an upper bound of the actual end-to-end delay. This upper bound can be derived from delay models based on the known or assumed behavior of switches using such mathematical tools as queuing theory and network calculus. Delay modeling for switched Ethernet with a given input traffic has been addressed in many works (Song *et al*., 2002; Wang *et al*., 2002). Here d_{ij} for each traffic flow is obtained from the delay model derived in our previous work (Zhang and Wang, 2010) using network calculus. The goal of DAIEN-RC is to guarantee that $d_{ij}<T_{ij}$, $\forall(i,j)|c_{ij}\neq 0$ by optimizing the allocations of the N devices to the M switches arranged in a given topology.

Note that the maximal tolerable delays for different communication pairs within a network may be different whereas for any given pair the tolerable delay typically does not change over time. Tolerable delays in industrial networks can generally range from milliseconds to minutes. In most practical networks, the delay requirements cover only part of that huge range, but nevertheless can vary considerably among the communication pairs. This variety of real-time requirements within the network is a notable difference between industrial applications and other real-time applications such as multimedia streaming. This should be considered in modeling the problem.

To account for this diversity of real-time requirements, 'relative delay' is introduced as

$$d_{ij}^{\mathrm{rel}}=d_{ij}/T_{ij},\quad \forall(i,j)\,|\,c_{ij}\neq 0. \tag{2}$$

Relative delay normalizes the end-to-end delay of a given communication pair according to its specific real-time requirement. Based on relative delay, an objective function of DAIEN-RC can be formulated as

$$f=\sum_{\forall i,j\in\{(i,j)|c_{ij}\neq 0\}}\beta_{ij}d_{ij}^{\mathrm{rel}},\ \beta_{ij}=\begin{cases}\beta_p, & d_{ij}^{\mathrm{rel}}>1,\\ 1, & d_{ij}^{\mathrm{rel}}\leq 1.\end{cases} \tag{3}$$

This objective function aims at minimizing the average relative delay in the whole network. β_{ij} is a penalty coefficient, which introduces penalties for those communication links where messages cannot be delivered within the required transmission deadline, in the following denoted as 'bad links'. A larger constant β_p puts a higher weight on avoiding violations of the real-time requirements.

In practical applications the following constraints need to be considered:

1. Each device must be allocated to exactly one switch.

2. The number of devices assigned to each switch must not exceed its number of available ports.

3. The traffic flowing to/from each switch port must not exceed the wire speed of the switch.

To formulate the constraints above, the following notations are given. Let matrix $\boldsymbol{S}$ as given in Eq. (4) specify which device is assigned to which switch: if device i is connected to switch k, s_{ik}=1; otherwise, s_{ik}=0.

$$\boldsymbol{S}=\begin{bmatrix} 0 & s_{12} & \cdots & s_{1M} \\ s_{21} & 0 & \cdots & s_{2M} \\ \vdots & \vdots & & \vdots \\ s_{N1} & s_{N2} & \cdots & 0 \end{bmatrix}. \tag{4}$$

Denote with p_k the number of ports available for devices in switch k. Let A be the set of all the ports of all switches. Q_a denotes the maximum supported data rate on a given port $a\in A$. Let matrices $\boldsymbol{W}^a$ and $\boldsymbol{V}^a$ as given in Eqs. (5) and (6) describe which traffic flows to/from port a. If traffic c_{ij} flows to port a, $w_{ij}^a=1$; otherwise, $w_{ij}^a=0$. If traffic c_{ij} flows from port a, $v_{ij}^a=1$; otherwise, $v_{ij}^a=0$.

$$\boldsymbol{W}^a=\begin{bmatrix} 0 & w_{12}^a & \cdots & w_{1N}^a \\ w_{21}^a & 0 & \cdots & w_{2N}^a \\ \vdots & \vdots & & \vdots \\ w_{N1}^a & w_{N2}^a & \cdots & 0 \end{bmatrix}, \tag{5}$$

$$\boldsymbol{V}^a=\begin{bmatrix} 0 & v_{12}^a & \cdots & v_{1N}^a \\ v_{21}^a & 0 & \cdots & v_{2N}^a \\ \vdots & \vdots & & \vdots \\ v_{N1}^a & v_{N2}^a & \cdots & 0 \end{bmatrix}. \tag{6}$$

The constraints can then be formulated as

$$\sum_{k=1}^{M} s_{ik}=1, \quad \forall i\in\{1,2,...,N\}, \tag{7}$$

$$\sum_{i=1}^{N} s_{ik}\le p_k, \quad \forall k\in\{1,2,...,M\}, \tag{8}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^a r_{ij}\le Q_a,\ \sum_{i=1}^{N}\sum_{j=1}^{N} v_{ij}^a r_{ij}\le Q_a,\ \forall a\in A. \tag{9}$$

3 Steady state genetic algorithm

An ssGA is used in the proposed hybrid heuristic algorithm. It is a variant of GA that generates only one offspring in each generation.

3.1 Representation and decoding

The potential solutions to a problem are encoded with strings of numbers called chromosomes. As opposed to the integer encoding for DAIEN-RC proposed in Krommenacker *et al.* (2002), Georges *et al.* (2006), and Zhang and Zhang (2007), the chromosome representation in this paper is based on random keys. More details of random key representation can be found in Bean (1994). Each device with index i is assigned a random real value $x_i\in[0, 1]$, $i\in\{1, 2, \ldots, N\}$, and the real string $x_1, x_2, \ldots, x_N$ forms the chromosome.

The decoding of a chromosome into a device assignment scheme can be accomplished as follows. Devices are sorted in ascending order of the values in the real string and then divided into M coherent groups. Each group corresponds to a switch, and the size of each group equals the number of ports P of each switch. Finally the M groups of devices are assigned to the M switches. Fig. 2 shows an example.

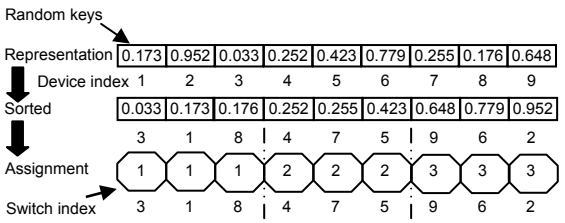

Fig. 2 An illustration of random key representation and decoding

Devices 1–9 are assigned to switches 1–3. Assuming each switch has P=3 ports, the given chromosome corresponds to an assignment of devices {1, 3, 8} to switch 1, devices {4, 5, 7} to switch 2, and devices {2, 6, 9} to switch 3

As opposed to integer encoding, the advantage of random keys is that any random key vector can be interpreted as a feasible solution, i.e., a solution that meets all constraints. By elimination of nonfeasible solutions, the size of the search space is decreased from M^N for integer representation to $N!/[(P!)^{M-1}(N-(M-1)P)!]$ for random key based representation. Therefore, random key representation can dramatically increase the efficiency of the search.

3.2 Genetic operators

In an ssGA, a population is maintained containing a fixed number of individuals represented by different chromosomes. The individuals representing better solutions to the given problem have a larger

chance to survive and contribute to the next generation. The quality of each chromosome is evaluated by a fitness function. We simply use the objective function in Eq. (3) as the fitness function. For each newly generated individual, determining its fitness value requires the recalculation of end-to-end delays d_{ij} of all traffic flows.

In each generation exactly one offspring is created by selection, crossover, and mutation operations. Two parents are selected with the selection mechanism described in Tang and Yao (2007). One parent is selected by wheel selection (Gen and Cheng, 1997), and the other is selected randomly from the population. A child is created from the parents' chromosomes by uniform crossover (Michalewicz, 1994), possibly followed by swap mutation (Krommenacker *et al.*, 2002).

In an ssGA a replacement strategy is necessary to decide which member of the population will be replaced by the child. In this study, the child replaces the worst individual in the population. Thus, the next generation is formed, differing from the previous generation by only one individual.

4 Reduced variable neighborhood search

In our hybrid GA-rVNS, rVNS is used to strengthen the search ability of ssGA. ssGA is used to globally explore the entire search space, while rVNS exploits the promising regions more thoroughly.

rVNS proposed by Mladenovic and Hansen (2001) is a simplified version of variable neighborhood search (VNS) (Mladenovic and Hansen, 1997). Starting from a given potential solution of a problem, rVNS further optimizes the solution by exploring different neighborhoods in a stochastic way rather than exhaustively searching the neighborhoods. rVNS is quite suitable for DAIEN-RC, since comprehensive local search for this problem is computationally expensive. Especially for large scale networks, the size of neighborhoods is typically very large, and the cost for evaluating the correspondingly large number of fitness values in an exhaustive local search scheme would be prohibitively large.

In adopting the rVNS for the hybrid GA-rVNS, we propose a novel scheme for guiding the search in rVNS. The generation of a neighborhood is no longer done exclusively at random, but restricted to a subset of the solution space. This helps to keep the computational effort low. In the following subsections, we will introduce the proposed neighborhood structures, the control scheme of generating neighbors, and the framework of the proposed rVNS.

4.1 Neighborhood structures

The neighborhood of a solution is the set of all solutions that can be derived from a given solution by executing a specific type of operation on the chromosome. Let us denote with N_k (k=1, 2, ..., $k_{\max}$) a set of neighborhood structures, each defined by a different type of operation, and with $N_k(x)$ the set of solutions in the kth neighborhood of x. In the following, two alternative operations ($k_{\max}$=2) are presented for defining the neighborhood:

'Device swap' swaps two devices connected to different switches. It is frequently used in a local move in combinatorial optimization problems. The time complexity of a complete search within a neighborhood defined by device swap is $O((N-P)^2/2)$.

'Switch swap' swaps the placements of two switches together with the connected devices. Switch swap is a group operation that exchanges the positions of two groups of devices at one time. The complexity of a complete search within a neighborhood defined by switch swap is $O((M-1)^2/2)$.

4.2 Control scheme for neighbor generation

Before presenting the control scheme for neighbor generation, we first introduce the notion of 'swap distance'. For a solution x and a neighbor solution x' generated from x by a single device swap operation, the swap distance $d_s(x, x')$ denotes the number of switches that a message has to pass through between the two exchanged devices. Correspondingly, for switch swap operation, swap distance $d_s(x, x')$ denotes the number of switches between the swapped switches. For example, in Fig. 3 the swap distance of swapping devices 1 and 2 is 5.

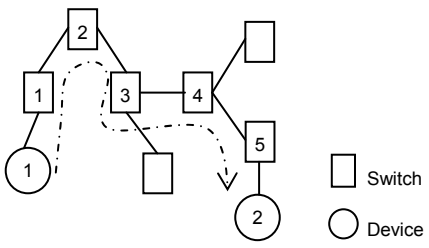

Fig. 3 An illustration of swap distance
The swap distance of swapping devices 1 and 2 is 5

The potential impact of a swap operation on network performance is not identical for all swap pairs, but is related to the relative placement of the devices to be swapped. The transmission delay of any message passing through any switch along the path between the swapped devices may be influenced. A swap operation with a larger swap distance therefore generally impacts a larger number of end-to-end delays of messages.

As concluded in Hansen and Mladenovic (2001), for many problems local minima are relatively close to each other with respect to one or several neighborhoods. This implies different swap distances are preferred in different phases of evolution. At the very beginning of evolution, moves with larger swap distances will be acceptable since a larger improvement of network performance may be obtained within one move. When the search has proceeded closer to a local optimum, the search should focus more on the vicinity of the current solution, and therefore smaller swap distances are preferred.

A neighbor x' of solution x is generated by choosing x' randomly from the set $\{x_0|d_s(x_0, x)=d_\lambda, x_0\in N_k(x)\}$, where the swap distance d_λ is generated randomly according to the probability function

$$P(d_\lambda)=\frac{1-\sigma}{1-\sigma^D}\sigma^{d_\lambda-1},\ \sigma\in(0,\ 1),\ d_\lambda\in\{1,2,...,D\},\quad(10)$$

where D is the maximum possible swap distance. According to this probability function, swap pairs with longer distances generally have lower probability to be generated. The described adjustment of the swap distance distribution with progressing evolution is achieved by varying σ according to

$$\sigma=\begin{cases}\dfrac{\alpha}{\ln\Delta f}, & \Delta f<\mathrm{e}^{\alpha},\\ 1-\varepsilon_1, & \Delta f\geq\mathrm{e}^{\alpha},\end{cases}\quad(11)$$

where ε_1 is a very small positive value, $\Delta f(t_{i-1}, t_i)=(f(t_{i-1})-f(t_i))/f(t_{i-1})$ is the observed improvement of the fitness value over a fixed time window $[t_{i-1}, t_i]$, and $\alpha\in(-\infty, 0)$ defines a threshold of the control scheme. Fig. 4 illustrates the neighbor generation scheme with $\alpha=-2$, $D=10$. Since Δf converges to zero as evolution proceeds to convergence, σ will gradually become reduced as soon as Δf falls below the given threshold, thus increasingly giving preference to smaller swap distances.

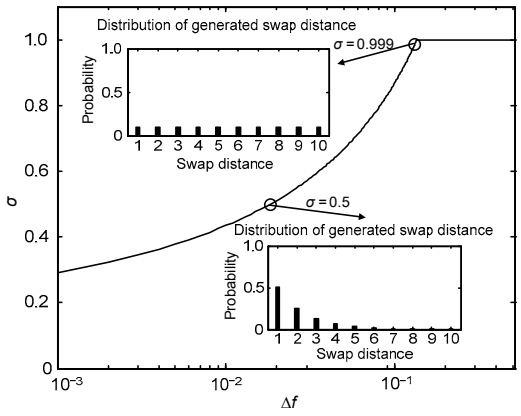

Fig. 4 An illustration of the neighbor generation scheme with $\alpha=-2$, $D=10$

4.3 Framework of rVNS

In GA-rVNS, rVNS is terminated when the improvement Δf becomes smaller than a pre-defined value ε_1. The framework of rVNS is shown in Fig. 5.

Input: initial x.
Output: optimized x.
$t_0\leftarrow 0$; $i\leftarrow 0$; $\Delta f(t_0, t_1)\leftarrow 1$;
while $\Delta f>\varepsilon_2$ **do**
  $k\leftarrow 1$;
  while $k\leq k_{\max}$ **do**
    Generate swap distance d_λ according to Eqs. (10) and (11);
    Choose x' randomly from $\{x_0|d_s(x_0, x)=d_\lambda, x_0\in N_k(x)\}$;
    Calculate all end-to-end delays d_{ij} for solution x';
    if $f(x')<f(x)$ **then**
      $x\leftarrow x'$;
    end if
    $k\leftarrow k+1$;
  end while
  if running time $t=t_i$ **then**
    Update $\Delta f(t_{i-1}, t_i)\leftarrow(f(t_{i-1})-f(t_i))/f(t_{i-1})$;
    $i\leftarrow i+1$; $t_i\leftarrow t_{i-1}+\Delta t$;
  end if
end while
return x

Fig. 5 Framework of the proposed rVNS

4.4 Summary of GA-rVNS

An initial population is randomly generated. Then GA-rVNS repeatedly improves the population by alternately applying the GA operators and the rVNS scheme described above until a predefined time limit has been reached. Fig. 6 shows the framework of the proposed hybrid heuristic.

Input: the communication traffic matrix.
Output: the final solution of DAIEN-RC.
1 $t \leftarrow 0$;
2 Initialize population $P(0)$ randomly with size p_{size};
3 Evaluate $P(0)$ by fitness function f;
4 **while** a preset computation time has not been reached **do**
5 $t \leftarrow t+1$;
6 Select parent P_1 by wheel selection, and P_2 randomly from $P(t-1)$;
7 Generate a child $C(t)$ by mating parents with uniform crossover;
8 Apply swap mutation on $C(t)$ with a probability $p_{mutation}$;
9 Optimize $C(t)$ with the rVNS method;
10 Update $P(t)$ by deleting the worst individual in $P(t-1)$ and inserting $C(t)$ into $P(t-1)$;
11 **end while**
12 **return** the best individual in population $P(t)$ as the final solution of DAIEN-RC

Fig. 6 Framework of the proposed hybrid GA-rVNS

5 Computational results

Test cases are generated considering different application scenarios. The test results are analyzed to empirically study the performance of the proposed approach. All algorithms are implemented in Matlab.

5.1 Test cases

5.1.1 Network model

It is assumed in the test scenarios that the switches in the network are arranged in a linear topology. This network structure is common at field level for its cost efficiency and convenience of engineering and installation. Although linear topology is the main focus throughout this paper, the proposed approach is suitable for arbitrary topologies.

The wire speed of each switch is 100 Mb/s full duplex. A maximum of four end devices can be connected to each switch.

5.1.2 Traffic model

Communication relationships between pairs of devices in this study are based on the cyclic exchange of messages (Jasperneite *et al*., 2002). The maximal tolerable delay normally equals the cycle time of message exchange between pairs of devices. The traffic for the individual traffic flows is randomly generated following the principle given in Zhang and Zhang (2007). The cycle time is chosen randomly in the interval [10, 100] ms, the frame length is fixed at 100 bytes, and the average data rates per traffic flow are uniformly distributed in the interval [8, 80] kb/s. Although frame length does affect the end-to-end delays and thus the network performance, the qualitative result of the algorithm comparison is not affected by the frame length, so a detailed analysis for different frame lengths is not given here.

Two basic kinds of communication structures typical in industrial applications are considered: centralized communication between a controller device (e.g., a PLC) and a number of I/O devices and peer-to-peer communication between pairs of controller devices or I/O devices. Fig. 7 is an illustration of these two communication structures.

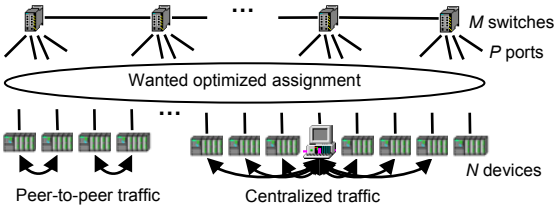

Fig. 7 Illustration of two communication structures

Based on these basic types of communication, two sets of tests are generated:

Test set 1: These tests assume only peer-to-peer traffic with each device communicating with an average of four other devices.

Test set 2: A mix of centralized and peer-to-peer traffic is assumed. This is a model for more complicated traffic scenarios in industrial communication networks.

For both test sets, the device number is varied from 40 to 250. Table 1 provides an overview of all generated test cases in this study.

Table 1 Generated test cases

Test case	Number of devices, N	Number of switches, M	Number of traffic streams	
			Test set 1	Test set 2
1	48	12	192	240
2	100	25	400	500
3	148	37	592	740
4	200	50	800	1000
5	248	62	992	1240

5.2 Methods and parameter settings

The performance of the following algorithms will be compared in this subsection.

pGA: The genetic algorithm in Georges *et al.* (2006) is implemented as a baseline for comparative evaluation of other algorithms. The parameters are chosen as follows: population size=15; integer chromosome representation; two-point crossover with probability $p_{\text{crossover}}$=0.6; swap mutation with probability p_{mutation}=0.2.

rVNS: Pure rVNS is implemented for comparison with hybrid GA-rVNS. Pure rVNS is terminated when a predefined time limit has been reached. The parameters are α=−0.5 and ε_1=10^{-3}.

GA-rVNS: The parameters for the ssGA part are: population size=20; p_{mutation}=0.1, β_p=100. For the rVNS part, the parameter α=−0.5, and ε_1, ε_2=10^{-3}.

Each algorithm is executed 20 times for each test case for a fair comparison of the average performance. Besides the average, the best and worst out of the 20 results are recorded to assess the scattering of the results among individual optimization runs.

The complexity of the considered algorithms is determined largely by the number of fitness evaluations. Therefore, the fixed number of function evaluations, n_{eval}, is used as the termination condition for comparing all the algorithms based on approximately equal computational effort (Hart *et al.*, 2000). The number of required fitness evaluations is related to the problem scale. For each test case all the algorithms are terminated after the same number of function evaluations.

5.3 Performance of the proposed optimization criterion

To show the effectiveness of the proposed optimization criterion, the objective function f is compared with the objective function

$$f^* = \max\left\{d_{ij} - T_{ij}\right\} \tag{12}$$

applied in Georges *et al.* (2006) for DAIEN-RC.

Real-time performance under different objective functions is measured by two metrics: average relative delay ($\bar{d}_{\text{r}}$) and the number of bad links. Fig. 8 shows the results of pGA with objective functions f and f^* on test set 1. In addition to the average results, the best and worst results obtained from the 20 optimization runs are indicated as error bars in the plots.

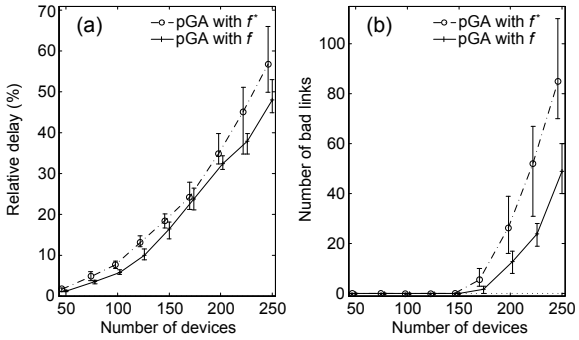

Fig. 8 Average relative delay (a) and the number of bad links (b) of pGA with objective functions f and f^* on test set 1

As can be seen in Fig. 8, pGA with the proposed objective function f consistently leads to better performance than with f^*. For the test case with N≤148, optimized device allocation based on either objective function can guarantee zero bad link. When the network scale gets larger, pGA with f can guarantee real-time performance, while pGA with f^* cannot. When network traffic becomes too large, neither of the objective functions can ensure that all delays are below their respective tolerable limits. However, pGA with the proposed f can lead to many fewer bad links than that with f^*.

Any improvement or degradation of relative delay on any link will have an impact on the proposed f during evolution, whereas a search algorithm operating on f^* will base its actions at any time only on the consideration of the links still violating the maximal tolerable delays, and ignore the other links. Regarding the convergence of the search algorithm, it is preferable to include quantitative information about all links in the objective function rather than having the objective value depend only on a varying subset of all links.

5.4 Effectiveness of the control scheme for neighbor generation

To show the effectiveness of the proposed control scheme for rVNS, three variants of GA-rVNS are implemented (Table 2). The GA-rVNS with a uniformly random neighbor generation serves as the baseline of comparison. For the second GA-rVNS variant, an optimized but fixed value of σ is chosen in

Eq. (10), giving preference to smaller swap distances in the neighbor generation. The third variant implements the dynamic adaptation of σ over the evolution process.

Table 2 Three versions of GA-rVNS

GA-rVNS variant	Neighbor generation	σ
1	Uniformly random	
2	Proposed guiding strategy Eq. (10)	0.5
3	Proposed guiding strategy Eq. (10)	Adaptive*

* According to Eq. (11)

Fig. 9 shows the results of the three versions of GA-rVNS when applied to test set 1. Shaping the distribution of the swap distances with a fixed σ gives a significant advantage over the uniformly distributed neighbor selection. The proposed strategy of gradually changing from a uniform distribution of swap distances towards increasingly smaller values of σ as the evolution proceeds leads to a further performance improvement, especially for large networks. As the search starts to converge and the observed fitness improvements per time are gradually becoming smaller, the decreasing σ will put an increasing weight on neighbors with small swap distances. The search is thereby focused on the vicinity of the current solution. The results obtained show that this adaptive strategy is clearly superior to the non-adaptive schemes and can thus be considered a very effective means to improve the efficiency of the rVNS.

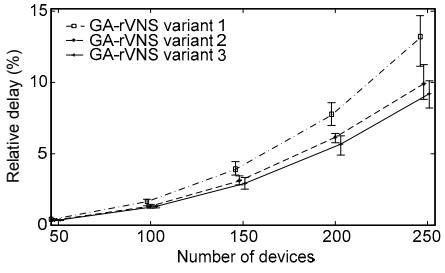

Fig. 9 Effectiveness of the proposed scheme for neighbor generation on test set 1

5.5 Comparative analysis of algorithms

The quality of the algorithms can be evaluated by the average relative delay $\overline{d}_r$ and the number of bad links when the network becomes overloaded. Figs. 10 and 11 summarize the results of pGA, rVNS, and GA-rVNS with the proposed f for each of the two test cases. The following observations can be made:

1. pGA results always have the largest (i.e., worst) fitness value. As the network scale increases, the performance of pGA degrades so much that some control messages cannot be delivered within the required cycle times.

2. GA-rVNS yields significantly better solutions than pGA in all the tests. Fig. 12 depicts the convergence of the 20 trials for pGA and GA-rVNS with device number N=100 in test set 1, showing that the evolutionary search behavior of the proposed GA-rVNS is better than that of pGA.

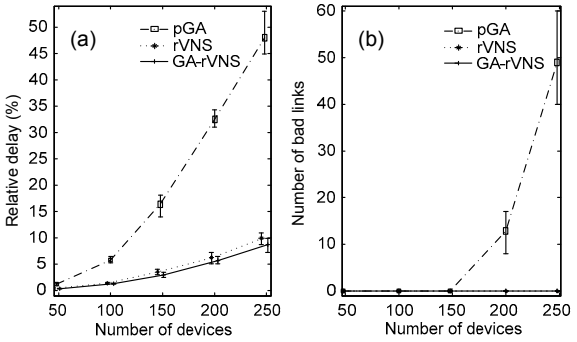

Fig. 10 Average relative delay (a) and the number of bad links (b) of different algorithms on test set 1

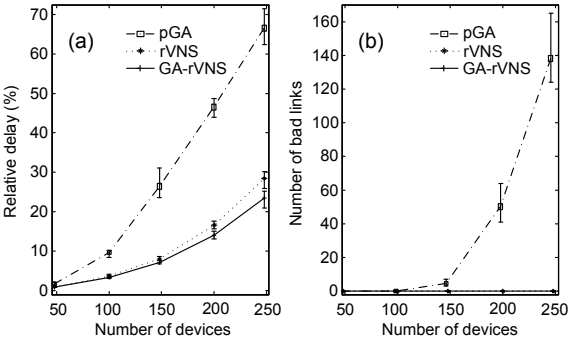

Fig. 11 Average relative delay (a) and the number of bad links (b) of different algorithms on test set 2

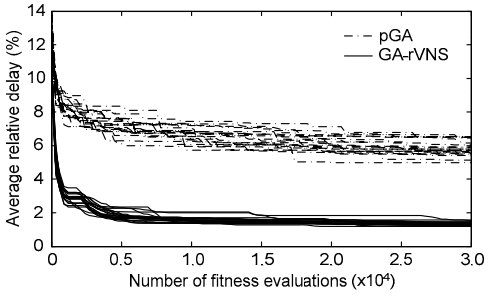

Fig. 12 Convergence behavior of pGA and GA-rVNS on test set 1 with N=100

3. In both test sets, compared with a pure rVNS, the improvement of GA-rVNS becomes more significant as the network size increases.

4. Lower average relative delay is obtained by GA-rVNS than by other algorithms, which means GA-rVNS yields a network that has more 'reserves' than the solutions found using other algorithms. Therefore, the robustness of the designed system is increased with respect to sudden changes in input traffic or unforeseen addition of devices.

5. The capability of GA-rVNS for optimizing larger scale problems is superior to that of pGA. Even for very large network sizes, GA-rVNS can guarantee the required real-time performance, while the other algorithms generate solutions with a significant number of bad links.

6 Conclusions

We propose a hybrid genetic algorithm incorporating a reduced variable neighborhood search (GA-rVNS) algorithm for finding an optimized device allocation in industrial Ethernet networks with regard to the guarantee of real-time performance. The objective of minimizing average relative delay explicitly considers the real-time performance on the individual data connections, which suits the inherent diversity of real-time requirements in industrial applications. The results for various test cases reflecting different network sizes and traffic types show that the proposed scheme achieves a better performance than the existing scheme without increasing the computational complexity. The proposed combination of an objective function based on relative delay and the hybrid GA-rVNS algorithm has proven particularly suitable for large size problems.

Issues to be further investigated include alternative neighborhood structures for the rVNS part and alternative schemes for dynamically adapting the parameter σ.

References

Bean, J.C., 1994. Genetic algorithms and random keys for sequencing and optimization. *INFORMS J. Comput.*, **6**(2): 154-160. [doi:10.1287/ijoc.6.2.154]

Carro-Calvo, L., Salcedo-Sanz, S., Portilla-Figueras, J.A., Ortiz-Garcia, E.G., 2010. A genetic algorithm with switch-device encoding for optimal partition of switched industrial Ethernet networks. *J. Network Comput. Appl.*, **33**(4):375-382. [doi:10.1016/j.jnca.2010.03.003]

Elbaum, R., Sidi, M., 1996. Topological design of local-area networks using genetic algorithms. *IEEE/ACM Trans. Network.*, **4**(5):766-778. [doi:10.1109/90.541324]

Felser, M., 2005. Real-time Ethernet-industry prospective. *Proc. IEEE*, **93**(6):1118-1129. [doi:10.1109/JPROC.2005.849720]

Gen, M., Cheng, R., 1997. Genetic Algorithm and Engineering Optimization. Wiley, New York.

Gen, M., Cheng, R.W., Lin, L., 2008. Network Model and Optimization Multiobjective Genetic Algorithm Approach. Springer Verlag Berlin Heidelberg, p.274-283.

Georges, G.P., Krommenacker, N., Divoux, T., Rondeau, E., 2006. A design process of switched Ethernet architectures according to real-time application constraints. *Eng. Appl. Artif. Intell.*, **19**(3):335-344. [doi:10.1016/j.engappai.2005.09.004]

Hansen, P., Mladenovic, N., 2001. Variable neighborhood search: principles and applications. *Eur. J. Oper. Res.*, **130**(3):449-467. [doi:10.1016/S0377-2217(00)00100-4]

Hart, W.E., Rosin, C.R., Belew, R.K., Morris, G.M., 2000. Improved Evolutionary Hybrids for Flexible Ligand Docking in AutoDock. *In*: Floudas, C.A., Pardalos, P.M. (Eds.), Optimization of Computational Chemistry and Molecular Biology. Kluwer, the Netherlands, p.209-230.

IEC 61784-2, 2005. Digital Data Communications for Measurement and Control - Part 2: Additional Profiles for ISO/IEC 8802-3 Based Communication Networks in Real-Time Applications. IEC, Switzerland.

IEEE 802.1Q, 2003. Virtual Bridged Local Area Networks. IEEE, New York, USA.

Jasperneite, J., Neumann, P., Theis, M., Watson, K., 2002. Deterministic Real-Time Communication with Switched Ethernet. Proc. 4th IEEE Int. Workshop on Factory Communication Systems, p.11-18. [doi:10.1109/WFCS.2002.1159695]

Kjellsson, J., Vallestad, A.E., Steigmann, R., Dzung, D., 2009. Integration of a wireless I/O interface for Profibus and Profinet for factory automation. *IEEE Trans. Ind. Electron.*, **56**(10):4279-4287. [doi:10.1109/TIE.2009.2017098]

Krommenacker, N., Divoux, T., Rondeau, E., 2002. Using Genetic Algorithm to Design Switched Ethernet Industrial Network. Proc. IEEE Int. Symp. on Industrial Electronics, **1**:152-157. [doi:10.1109/ISIE.2002.1026057]

Li, F., Zhang, Q., Zhang, W., 2007. Graph partitioning strategy for the topology design of industrial network. *IET Commun.*, **1**(6):1104-1110. [doi:10.1049/iet-com:20060677]

Michalewicz, Z., 1994. Genetic Algorithms+Data Structures= Evolution Programs. Springer Verlag, New York.

Mladenovic, N., Hansen, P., 1997. Variable neighborhood search. *Comput. Oper. Res.*, **24**(11):1097-1100. [doi:10.1016/S0305-0548(97)00031-2]

Song, Y., Koubaa, A., Simonot, F., 2002. Switched Ethernet for Real-Time Industrial Communication Modelling and

Message Buffering Delay Evaluation. Proc. 4th Int. Workshop on Factory Communication Systems, p.27-30. [doi:10.1109/WFCS.2002.1159697]

Tang, M.L., Yao, X., 2007. A memetic algorithm for VLSI floorplanning. *IEEE Trans. Syst. Man Cybern. B*, **37**(1): 62-69. [doi:10.1109/TSMCB.2006.883268]

Wang, Z., Song, Y.Q., Chen, J.M., Sun, Y.X., 2002. Real-time Characteristics of Ethernet and Its Improvement. Proc. 4th World Congress on Intelligent Control and Automation, **2**:1311-1318. [doi:10.1109/WCICA.2002.1020794]

Zhang, L., Wang, Z., 2010. Real-Time Performance Evaluation in Hybrid Industrial Ethernet Networks. Proc. 8th World Congress on Intelligent Control and Automation, p.1842-1845. [doi:10.1109/WCICA.2010.5554500]

Zhang, Q., Zhang, W.D., 2007. Using genetic algorithm to design switched Ethernet industrial network. *Eng. Appl. Artif. Intell.*, **20**(1):79-88. [doi:10.1016/j.engappai.2006.03.004]