# ALGORITHMS AND A TOOL SYSTEM FOR SCHEDULING OF DATA EXCHANGE OVER CHANNELS WITH CENTRALIZED CONTROL

V.V. Balashov, V.A. Balakhanov, V.A. Kostenko, R.L. Smeliansky, V.A. Kokarev
*Laboratory of Computer Systems, Department of Computational Mathematics and Cybernetics*
*Lomonosov Moscow State University*
GSP-1, 1-52, Leninskiye Gory, Moscow, 119991, Russia
E-mail: hbd@lvk.cs.msu.su, baldis@lvk.cs.msu.su, kost@lvk.cs.msu.su, smel@lvk.cs.msu.su, matrix@lvk.cs.msu.su

In this paper we address the problem of automatic scheduling of data exchange over a channel with centralized control in real-time avionics systems. Examples of such channels are MIL STD-1553B multiplex data bus and Fibre Channel FC-AE-1553 channel. Scheduling algorithms implementing greedy strategy and ant colony method are presented and experimentally evaluated. These algorithms support customization for specific requirements of the target onboard real-time avionics system. Results of experimental evaluation demonstrate that the ant colony algorithm outperforms greedy algorithms on the considered classes of input data. The proposed technology for data exchange scheduling is implemented in a tool system.

## 1 INTRODUCTION

Most modern onboard real-time avionics (RTA) systems are distributed systems. They include sensors, control nodes, computational nodes, actuators, data storage and display devices connected by communication channels. An architecture based on channels with centralized control (CC channels) is widely used in RTA systems. Examples of CC channels are MIL STD-1553B [**1**], STANAG 3910 [**2**], and Fibre Channel FC-AE-1553 [**3**]. The main focus of this paper is scheduling of data exchange for MIL STD-1553B bus.

Data exchange over a CC channel constitutes a sequence of application and service data transfers between devices attached to the channel. From scheduling point of view, each transfer is a *job*. Data exchange schedule defines start time for each job. Static scheduling strategy is typically used to schedule data exchange over CC channels in RTA systems. According to this strategy, scheduling is performed offline and the resulting schedule cannot be changed in runtime. The schedule is executed by the controller, which is one of the devices attached to the channel.

In RTA systems, the number of jobs to be scheduled reaches several hundreds. Technological requirements defined by specifics of RTA system's hardware and software are applied to the schedule. These requirements define a set of constraints on the schedule. A combination of high number of jobs, high channel load, and complicated constraints makes manual schedule construction infeasible. Construction of a correct schedule, which includes all jobs and meets all constraints, is a complex task and needs automation. Existing tool systems for static scheduling of computations and/or data exchange in RTA systems (e.g. [**4**, **5**]) provide insufficient support for constraints on the schedule.

In this paper a technology is presented for automatic scheduling of data exchange for CC channels. The technology includes a workflow a and scheduling algorithms. Section 2 introduces the main principles for organization of data exchange over a CC channel and gives examples of constraints on the data exchange schedule. Section 3 presents the proposed workflow for static scheduling of data exchange and states the requirements for a tool system to support this workflow. Section 4 describes the algorithms developed for scheduling of data exchange and presents results of their experimental evaluation.

# 2 ORGANIZATION OF DATA EXCHANGE OVER A CHANNEL WITH CENTRALIZED CONTROL

## *2.1 Cyclic scheme of data exchange*

Data exchange over a CC channel constitutes a sequence of application and service data transfers between devices attached to the channel (terminal devices). CC channel can have a bus topology (MIL STD-1553B [**1**]) or a ring topology (FC-AE-1553 [**3**]). Each device can operate as a data source and/or a data receiver.

One of the terminal devices is the channel controller. It manages the data exchange and monitors the state of other terminal devices. Only the controller can initiate data exchange over the channel, other terminal devices execute the commands issued by the controller (request/response scheme). This guarantees absence of collisions. The controller operates according to the exchange schedule defined during development of the RTA system. Data exchange is performed asynchronously with execution of primary functions of the terminal devices.

An RTA system can support several modes of operation, with different corresponding sets of data transfer jobs. In this paper, data exchange scheduling problems for different modes are considered separately. For each mode, a static schedule is constructed which cannot be changed in runtime. The schedule executed by the controller changes only when the RTA system operation mode is changed. In the schedule, a particular start time is assigned to each job.

Cyclic scheme of computation is typical for modern RTA systems. According to it, each RTA system device executes its software in a loop, on each iteration of which a block of output data is prepared for sending through the channel, in form of one or several *messages*.

Cyclic scheme of data exchange corresponds to the cyclic scheme of computation. For this scheme of exchange, the workload for a CC channel is a set of messages, each of which is intended for periodic transfer through the channel. For each message, its duration and required frequency of transfer is known. Several exchange jobs correspond to a single message. Data to be transferred in a message is computed during the RTA system operation.

Duration $l_{int}$ of the *scheduling interval*, for which the data exchange schedule must be constructed, equals the least common multiple of the messages' periods (period is a reciprocal value of frequency). In the RTA system runtime, after the end of the scheduling interval, the schedule is executed again. Number of data exchange jobs corresponding to a given message equals quotient of $l_{int}$ and the message's period. Section 2.2 shows how the jobs are formed for a message and how their deadlines are defined.

For cyclic scheme of data exchange, the scheduling interval is divided into intervals of equal duration, known as *subcycles* (see Figure 1). In each subcycle, one *job chain* can be executed (job chain is a sequence of jobs executed without intermediate delays). In the beginning and the end of the subcycle there are intervals on which no job can be scheduled.

## *2.2 Relationship between periodic messages and data exchange jobs*

From scheduling point of view, each message is described by its duration and period. For some messages, *phase offsets* can be defined that narrow down deadline intervals of corresponding jobs.

A set of data exchange jobs corresponds to each message. Deadline intervals for the jobs are defined by period and phase offsets of the message. Let us consider a message m with duration d, period T, phase offsets $\varphi_1$ and $\varphi_2$. Let $v_1$, $v_2$ and $v_3$ be the first three jobs of this message (total number of its jobs equals $l_{int}/T$). Attributes of these jobs are calculated as follows (i=1, 2, 3):

- left deadline: $s_i = (i-1)*T + \varphi_1$;
- right deadline: $f_i = (i-1)*T + \varphi_2$;

- deadline interval: $[s_i; f_i]$;
- duration: $t_i = d$;

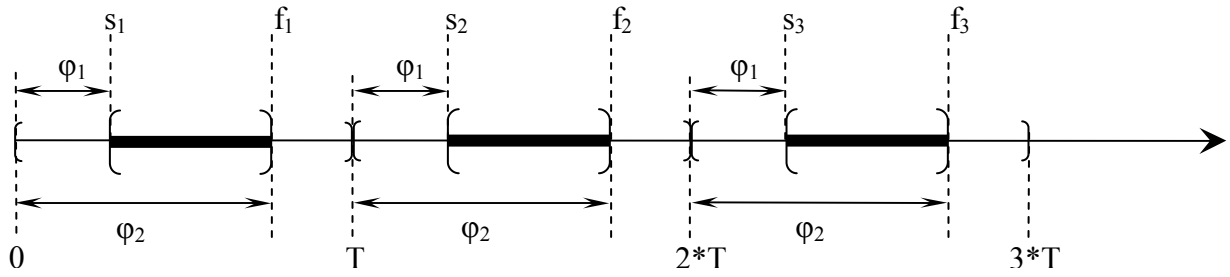



**Figure 1.** Deadline intervals for jobs

Figure 1 shows deadline intervals (bold horizontal lines) for jobs $v_1$, $v_2$, $v_3$. Each job must be executed within its deadline interval.

It should be noted that a set of *independent* jobs can be modeled within the cyclic scheme by a set of messages, each of which has period equal to $l_{int}$, and phase offsets that define the deadline interval for the only job of the message. In this case, $l_{int}$ is given as input data, not defined by messages' periods.

Let $V = \{v_i\}_{i=1}^{N_V}$ be a set of exchange jobs. An important characteristic of V is *channel load*, expressed by the formula:

$$E = \frac{1}{l_{int}} * \sum_{i=1}^{N_V} t_i * 100\%$$

### *2.3 Constraints on the schedule*

A correct data exchange schedule must meet a number of constraints, including:

1) general constraints:
   a) absence of collisions (execution intervals of jobs must not intersect);
   b) each job must be executed within its deadline interval;
2) constraints corresponding to technological requirements to data exchange, for instance:
   a) subcycle duration;
   b) duration of the reserved time interval in the beginning of the subcycle;
   c) duration of the reserved time interval in the end of the subcycle;
   d) time increment, by which the schedule can be shifted without violation of deadlines and other constraints;
   e) maximum allowed number of jobs in a chain;
   f) constraints on ordering of jobs within a subcycle (e.g. a job of message $M_2$ must be scheduled exactly after a job of message $M_1$);

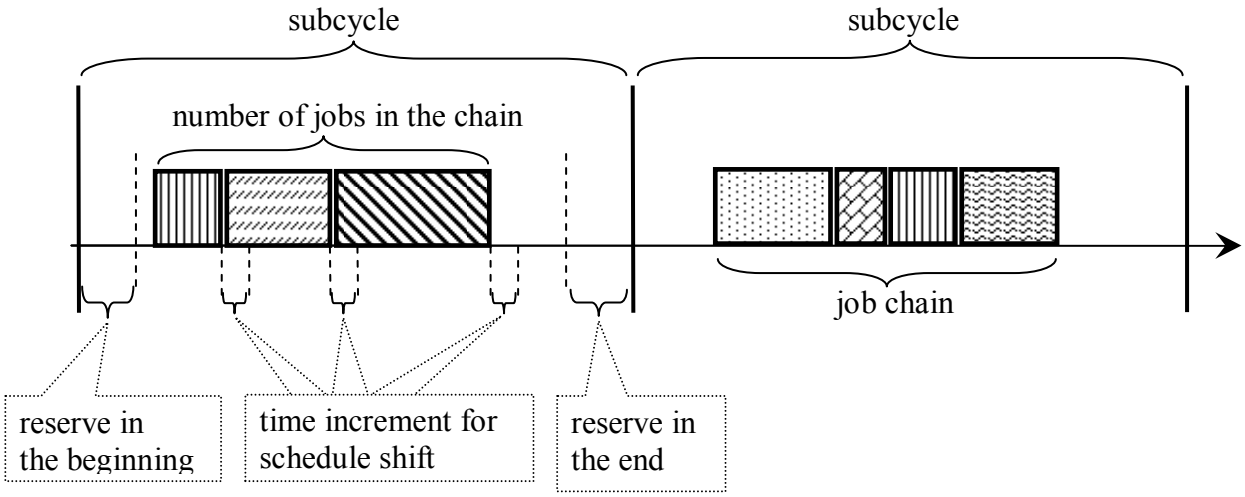



**Figure 2.** Cyclic scheme of data exchange

Figure 2 illustrates constraints 2.a – 2.e. A particular set of constraints on the schedule depends on specifics of hardware and software used in the target RTA system. As a result, an important requirement to the scheduling algorithm is its ability to be customized to support different sets of constraints.

All constraints listed above take place in real RTA systems based on MIL STD-1553B channels. However, these constraints are defined not by specifics of this particular standard, but by cyclic scheme of data exchange and specifics of its implementation. For instance, the reserved interval in the beginning of the subcycle (see 2.b) is necessary to reprogram the controller's adapter to run the next job chain. Time increment for schedule shift (2.d) guarantees that if some exchange job is executed unsuccessfully (e.g. due to noise on the channel), its execution can be instantly repeated without violation of constraints on the schedule, resulting only in a tolerable shift of the job chain's tail.

Therefore, the listed constraints can take place in RTA systems based on different CC channels with cyclic scheme of data exchange. For instance, FC-AE-1553 is designed to simplify migration from legacy MIL STD-1553B systems with *minimum changes* of data exchange logic (including the cyclic scheme if it is implemented in the legacy system) and data exchange supporting software. Cyclic scheme can be applied in the control bus of STANAG 3910 channel, since the control bus is a MIL STD-1553B.

Some constraints require specific interpretation depending on type of the CC channel. For instance, FC-AE-1553 is a pipelined channel, and execution intervals of consequent jobs may partially overlap.

## 3 WORKFLOW FOR DATA EXCHANGE SCHEDULING

### *3.1 Scheme of the workflow*

The proposed technology for data exchange scheduling assumes the following workflow:

1. Creation of the project: filling the database with information on structure of the onboard network and characteristics of workload for data exchange channels.
2. Automatic construction of a data exchange schedule which is *complete* (includes all data exchange jobs), and *correct* (meets all constraints, including those imposed by technological requirements); optional manual correction of the schedule.

3. In case a complete and correct schedule cannot be constructed: automatic correction of technological requirements to data exchange, so that such schedule can be constructed with updated requirements.
4. Generation of software code which defines the schedule for the devices attached to the channel.
5. Generation of reports on input data (see step 1) and constructed schedules, for inclusion in documentation on the RTA system.

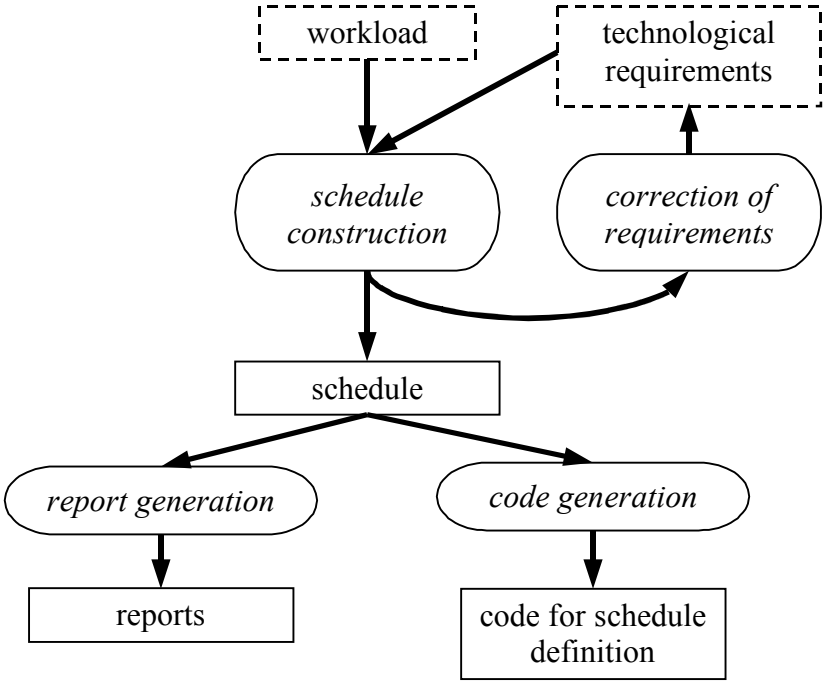



**Figure 3.** Exchange scheduling for a single channel

Figure 3 shows a workflow diagram for scheduling of data exchange over a single CC channel. Main categories of data are shown as rectangles (dashed for input data, solid for output data). Workflow steps are shown as rounded rectangles.

### *3.2 Requirements for tool support*

According to the workflow described in Section 3.1, a tool system for data exchange scheduling must support the following activities:

- entering and storing the input data;
- automatic construction and storing of data exchange schedules;
- automatic correction of technological requirements to data exchange (in case it is impossible to construct a complete and correct schedule with current requirements);
- visualization and manual correction of the schedule;
- generation of schedule definition code for devices of the target RTA system;
- generation of reports on input data and constructed schedules.

Data exchange scheduling tools can be applied on different stages of RTA system development, from conceptual phase to industrial prototyping. Applying the tools on the conceptual phase enables development of consistent specifications of information interchange for different devices attached to CC channels. Applying the tools on prototyping phase automates the development of system software for data exchange control. The tools can also support upgrading the RTA system by constructing updated data exchange schedules in case new devices are connected to the channel, existing devices are replaced by new ones, or application software is modified.

To be applicable to development of RTA systems with different basic hardware and software, or different subsystems of a complex RTA system, scheduling tools must support customization according to specifics of the target class of RTA systems. This includes support for:

- considering different technological requirements during schedule construction;
- customization of the template for generated source code;
- customization of the templates for reports;
- schedule construction for different standards of CC channels [**1, 2, 3**].

## 4 PROPOSED SCHEDULING ALGORITHMS

### *4.1 Greedy scheme for data exchange scheduling*

The problem of scheduling the data exchange over a CC channel belongs to the NP-hard complexity class. This justifies the use of heuristic algorithms to solve the problem. Basic scheduling algorithm applied within the proposed technology is the greedy algorithm. It takes the following data as input:

- set of jobs $V = \{v_i\}_{i=1}^{N_V}$, for each of which is given:
  - duration $t_i$;
  - deadline interval $[s_i; f_i]$;
- technological requirements to data exchange.

Output of the algorithm includes:

- schedule: $S \subseteq V$, with start time assigned to each job $s \in S$;
- list (set) of excluded jobs: $U = V \setminus S$.

Let us denote by *t* the earliest point of time on which scheduling of jobs is allowed. The scheme of the greedy scheduling algorithm is as follows:

1) t := 0;
2) select, according to a given deterministic criterion, a job which can be started at *t* without violation of constraints on the schedule;
   if there are no such jobs, set t equal to the earliest moment of time on which some job can be started (only increase of t is allowed), and go to step 5;
3) put the selected job on schedule, with start time t;
4) set t equal to the finish time of the scheduled job;
5) take account of constraints on the schedule defined by technological requirements to data exchange:
   5.1) update the value of t;
   5.2) update the deadline intervals of jobs not put on schedule;
6) put on the list U of excluded jobs those jobs that are not put on schedule and have deadline interval shorter than duration;
7) if there are any jobs not put on schedule and not excluded, go to step 2, else **stop**.

This scheme guarantees that the constructed schedule is correct, i.e. meets all constraints. However, the schedule can be incomplete (some jobs are excluded, $U \neq \varnothing$). The objective function to be maximized by the greedy algorithm is the number of scheduled jobs.

Customization of the greedy scheme to support specific technological requirements is performed by including procedures, which take account of corresponding constraints on the schedule, to steps 2 and 5. For instance, after the current job chain reaches maximum number of jobs, on step 5 variable t is set to start time of the next subcycle. In this way the set of supported technological requirements can be extended without modification of the greedy scheme.

Specifics of the channel workload (sets of jobs) for target RTA system can be considered in the criterion for job selection on step 2. Examples of greedy criteria are:

- $H_{EDF}$: prefer jobs with minimum finish deadline (*earliest deadline first* [**6**]);

- $H_{EFT}$ : prefer jobs with minimum earliest finish time [**7**];
- $H_{RM}$ : prefer jobs of messages with maximum frequency (*rate monotonic* [**6**]);
- $H_{LSF}$ : prefer jobs with minimum slack (slack is the temporal difference between job's deadline and job's earliest finish time).

Computational complexity of a greedy scheduling algorithm depends on the criterion for job selection on step 2. For $H_{EDF}$ and $H_{RM}$ criteria, the algorithm's complexity is $O(N_V^2 * \log(N_V))$.

The scheme described above, and an algorithm based on it [**7**], were initially developed for solving the problem of scheduling data exchange over MIL STD-1553B bus. The same scheme can be applied to solving similar problems for other CC channels. In particular, a version of this algorithm was developed for FC-AE-1553 channel with ring architecture [**8**]. The main specific feature of this algorithm is support for pipelined transfer of data.

### *4.2 Ant colony algorithm for data exchange scheduling*

Although the greedy scheme described above enables support for a wide variety of technological requirements to data exchange, and can be tuned for specific problems, the algorithm based on this scheme requires the criterion for job selection (see step 2) to be defined in advance. For each heuristic criterion there is a set of input data for which a correct and complete schedule exists but cannot be constructed by greedy algorithm based on this criterion. This is caused by deterministic nature of the greedy algorithm: on each iteration, the criterion deterministically selects one job to be put on schedule. Scheduling of this job on current iteration may lead to impossibility to schedule some other jobs later.

There are at least two ways to fix this disadvantage:

- apply limited enumeration of jobs on step 2 instead of greedy heuristic or in combination with it;
- apply a nondeterministic criterion for job selection, which can select with some probability any job from the set of jobs not put on schedule.

We propose to follow the second way by using a hybrid scheme which combines the greedy scheme described above with the ant colony method [**9**, **10**]. Ant colony method is used to find the ordering of jobs in the schedule, and steps 3-5 of the greedy scheme are invoked to put jobs on the schedule.

Ant colony algorithm (ant algorithm) searches a given graph for paths that meet certain criteria and have maximum value of the objective function. During the execution of the algorithm, partial paths (sequences of adjacent vertices) are built based on value of a heuristic criterion (local objective function, similar to a greedy heuristic criterion) and on amount of pheromone on edges ("experience" from previous iterations). Next vertex of a path is selected by the roulette scheme with probability proportional to the value of local objective function and to amount of pheromone on the edge leading to the vertex.

Ant algorithm supports automatic adaptation to specifics of the problem by marking up the graph's edges. The markup is used for construction of the solution on each iteration, and is refined while the number of iterations grows.

To apply an ant algorithm to construction of data exchange schedule, the scheduling problem must be converted to a problem of finding a path with certain properties in a graph. Input and output data for the algorithm are the same as for the greedy scheme (see Section 4.1). Let us denote the number of subcycles in the scheduling interval as $N_{SC}$. The proposed ant algorithm will deal with a fully connected graph, each vertex of which falls into one of the categories:

1) vertices $r_1,...,r_{N_V}$, corresponding to jobs;

*2)* vertices $w_1,...,w_{N_{sc}}$, corresponding to subcycles.

The ant algorithm searches this graph for paths meeting the following constraints:

- the path passes through each vertex exactly once;

- the path begins with $w_1$;
- the path can pass through $w_i$ only after passing $w_1,...,w_{i-1}$.

The objective function to be maximized by the algorithm is quotient of the number of scheduled jobs and the number of jobs in the input set V. The schedule is build along with construction of the path. Passing through the vertex $w_i$ corresponds to starting a job chain for the i-th subcycle. Passing through the vertex $r_j$ corresponds to adding the job $v_j$ to the end of job chain of the current subcycle (if the resulting partial schedule is correct) or to putting $v_j$ on the list U of excluded jobs (otherwise). A job is put on schedule by a procedure that includes steps 3-5 of the greedy scheme for schedule construction (see Section 4.1). Local objective function guarantees that probability of selecting a vertex for a job that can be correctly scheduled (added to the current partial schedule) is higher than probability of selecting a vertex for a job that cannot be correctly scheduled.

Scheme of the proposed ant algorithm is as follows:

1) construct several paths according to the following procedure:
   - 1.1) set the first vertex: $w_1$;
   - 1.2) determine the set of vertices to each of which a pass from the current vertex is allowed according to constraints on the path;
   - 1.3) calculate the probability of pass to each of the vertices from this set, according to value of the local objective function and amount of pheromone on corresponding edges;
   - 1.4) choose the next vertex of the path by the roulette scheme, according to calculated probabilities;
   - 1.5) update the partial schedule:
     - (a) if the chosen vertex corresponds to a job:
       - (i) put the job on schedule (if this job can be correctly scheduled);
       - (ii) put the job on the list U of excluded jobs (otherwise);
     - (b) if the chosen vertex corresponds to a subcycle: switch to the next subcycle;
   - 1.6) if the path construction is not finished, go to step 1.2;
2) calculate the objective function for constructed paths;
3) update the amount of pheromone on the edges belonging to constructed paths, depending on values of the objective function on these paths;
4) if the termination condition is not met, go to step 2, else **stop**.

Termination condition for the ant algorithm is usually exceedance of given maximum number of iterations without improving the value of the objective function.

Ant algorithm's ability of automatic adaptation to a specific problem comes at the expense of higher computational complexity than that of an algorithm based on the greedy scheme. Computational complexity of constructing a single path (and corresponding schedule) is close to complexity of a greedy scheduling algorithm.

### *4.3 Experimental evaluation of scheduling algorithms on synthetic sets of jobs*

Computational experiments were performed to compare the ant algorithm introduced in Section 4.2 to algorithms based on greedy scheme with $H_{EFT}$ and $H_{EDF}$ criteria for job selection (see Section 4.1).

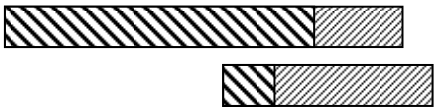
The aim of these experiments was to verify the ability of the ant algorithm to schedule synthetic sets of jobs which are designed to exploit drawbacks of greedy algorithms based on heuristics $H_{EFT}$ and $H_{EDF}$. The essence of such a drawback is that the greedy heuristic chooses an obviously wrong order of jobs in the schedule, forcing some jobs to be unscheduled. Such issue also takes place on real data, causing practical issues, however to make the effect most explicit it is necessary to use custom generated job sets. The hypothesis to be checked was that nondeterministic ant algorithm iteratively finds the proper ordering of jobs.

In experiments described in this section, schedules were constructed with 10 subcycles, with 10% reserved time in the end of each subcycle (requirement 1.c from Section 2.3). Duration of each subcycle was 1000 time units. In each experiment, the number of jobs was 80, and jobs were independent.

Two classes of input data were considered. First class contains pairs of jobs such that the deadline interval for the second job begins closely to the middle of deadline interval for the first job, and finishes after the end of the deadline interval of the first job; earliest finish time for the first job is greater than that for the second job. For example:

$t_1 = 70, s_1 = t , f_1 = t + 90;$

$t_2 = 10, s_2 = t + 50 , f_2 = t + 100.$

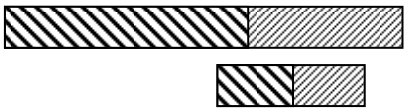


where $t_i$ is job's duration, $s_i$ is its left deadline, $f_i$ is its right deadline. Values for $t_i$ were randomly generated, and values for $s_i$ and $f_i$ were calculated as shown above.

The second class of input data contains pairs of jobs such that the deadline interval for the first job contains the deadline interval for the second job; earliest finish time for the first job is lesser than that for the second job. For example:

$t_1 = 60, s_1 = t , f_1 = t + 100;$

$t_2 = 20, s_2 = t + 50 , f_2 = t + 90.$



Values for $t_i$, $s_i$, $f_i$ were computed in a similar way as for the first class.

From each class of input data, sets of jobs were considered with different values of channel load E (introduced in Section 2.2). Values in tables 1 and 2 refer to the quotient of the number of jobs put on schedule and total number of jobs in the input set. Values are taken for different algorithms and different channel loads.

The experiments demonstrate that fraction of scheduled jobs for greedy algorithms considerably depends on the class of input data. At the same time the ant algorithm constructs almost complete schedules on different classes of data.

It must be noted that in practice of RTA system development it is essential to put *all* data exchange jobs on schedule, resulting in a complete and correct schedule. Manual fine tuning of greedy algorithms to specifics of a class of input data enables construction of complete schedules for high channel loads (value of E up to 70%).

| Channel load E, % | Fraction of scheduled jobs | | |
|---|---|---|---|
| | Ant algorithm | Greedy algorithm, $H_{EFT}$ | Greedy algorithm, $H_{EDF}$ |
| 10 | **1** | **1** | 0.6 |
| 20 | **1** | **1** | 0.59 |
| 30 | 0.99 | **1** | 0.57 |
| 40 | 0.99 | 0.99 | 0.57 |
| 50 | 0.97 | 0.96 | 0.56 |
| 60 | 0.95 | 0.96 | 0.56 |
| 70 | 0.94 | 0.93 | 0.55 |
| 80 | 0.92 | 0.92 | 0.54 |
| 90 | 0.91 | 0.9 | 0.52 |

**Table 1.** Experimental results, data from class 1

| Channel load E, % | Fraction of scheduled jobs | | |
|---|---|---|---|
| | Ant algorithm | Greedy algorithm, $H_{EFT}$ | Greedy algorithm, $H_{EDF}$ |
| 10 | **1** | 0.56 | **1** |
| 20 | 0.99 | 0.55 | **1** |
| 30 | 0.99 | 0.56 | 0.98 |
| 40 | 0.97 | 0.55 | 0.97 |
| 50 | 0.95 | 0.54 | 0.95 |
| 60 | 0.94 | 0.53 | 0.93 |
| 70 | 0.93 | 0.53 | 0.92 |
| 80 | 0.92 | 0.51 | 0.91 |
| 90 | 0.92 | 0.5 | 0.92 |

**Table 2.** Experimental results, data from class 2

### *4.4 Approach to correction of requirements to data exchange*

In case the scheduling algorithm does not construct a correct and complete schedule for a given set of jobs and technological requirements to data exchange, there are the following alternatives:

1) use another scheduling algorithm;
2) selectively exclude some jobs from the set of jobs, or modify deadline intervals for jobs;
3) modify the technological requirements to data exchange.

The proposed approach is based on the third alternative; the scheduling algorithm and the set of jobs are *not* changed. We consider modifications of technological requirements that alter *values* of the requirements, e.g. increase the maximum allowed number of jobs in a chain (to allow putting more jobs in one subcycle). To minimize the impact of changing the requirements on the RTA system (e.g. longer chains require more memory on the adapter card), the changes of requirements' values must be *minimal*. So a measure of total variation of requirements is introduced which is a weighted sum:

$$cost(\bar{x}^{new}) = \sum_{i=1}^{N} c_i * \left| x_i^{old} - x_i^{new} \right|,$$

where vector $\bar{x}^{old}$ keeps initial values of requirements, and $\bar{x}^{new}$ keeps the altered values; $c_i$, $i = 1..N$, are positive cost multipliers. For each requirement's value $x_i$ there is a range of allowed variation, defined by specifics of the RTA system under development.

The problem is to find, in the given ranges, the requirements' values $\bar{x}^{new}$, with which the given scheduling algorithm constructs a correct and complete schedule for the considered set of jobs. The objective function to be minimized is $cost$. A heuristic algorithm was developed for solving this problem [**11**]. Experimental evaluation [**12**] of this algorithm shows that value of the objective function on the resulting solution deviates from the optimal value by no more than 15%, on problems with up to 5 variable requirements to data exchange.

## 5 CONCLUSION

In this paper a technology was presented for scheduling of data exchange through a channel with centralized control (CC channel). MIL STD-1553B was considered as an example of CC channel. One of the basic requirements to algorithms and tools for scheduling of data exchange is support for customization according to specifics of the target class of RTA systems. A greedy scheme of exchange scheduling was presented that meets this requirement. A new hybrid algorithm was proposed that combines the greedy scheme and ant colony method, and enables automatic

adaptation to the specifics of a particular set of input data. Experimental evaluation of greedy algorithms and this ant algorithm justifies the adaptiveness feature of the latter which allows the ant algorithm to outperform greedy algorithms on the considered classes of input data. Future work on the ant algorithm includes improving the algorithm in order to enable construction of complete schedules with high channel loads, for different classes of input data (see Section 4.3);

The presented technology was implemented in the Scheduler CAD tool system **[13]** which was successfully used for generation of data exchange schedules for CC channels in RTA systems. Future work on the technology as a whole includes its extension for advanced standards of CC channels, including Fibre Channel FC-AE-1553.

REFERENCES


**1** U.S. Standard MIL STD-1553B "Aircraft internal time division command/response multiplex data bus". U.S. Department of Defense, Washington D.C., 1978.

**2** Guide to Digital Interface Standards for Military Avionics Applications. Technical report ASSC/110/6/2-Issue 3, Avionics Systems Standardization Committee, 2006.

**3** Information Technology – Fibre Channel – Avionics Environment – Upper Layer Protocol and Profile based on MIL-STD-1553B Notice 2 (FC-AE-1553). Technical Report INCITS/TR-42:2007. INCITS, 2007.

**4** Lee, Y.-H., Kim, D., Younis, M., Zhou, J. Scheduling tool and algorithm for integrated modular avionics systems. Proc. 19th Digital Avionics Systems Conference, Philadelphia, PA, USA, 2000.

**5** Goltz, H.-J., Pieth, N. A Tool for Generating Partition Schedules of Multiprocessor Systems. Proc. 23rd Workshop on (Constraint) Logic Programming, Potsdam, Germany, 2009.

**6** Liu, C.L., Layland, J.W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 1973, 20(1), 46–61.

**7** Kostenko, V.A., Guryanov, E.S. An Algorithm for Scheduling Exchanges over a Bus with Centralized Control and an Analysis of its Efficiency. *Programming and Computer Software*, 2005, 31(6), 340–346.

**8** Bychkov, I.A., Kostenko, V.A. Problems of data exchange scheduling and network topology selection for arbitrated loop. Proc. 3rd Conference for Methods and Tools for Information Processing, Moscow, Russia, 2009, 204–214.

**9** Dorigo, M. Optimization, Learning and Natural Algorithms. Ph.D. Thesis, Politechnico di Milano, Milano, 1992.

**10** Shtovba, S.D. Ant Algorithms: Theory and Applications. *Programming and Computer Software*, 2005, 31(4), 167–178.

**11** Balashov, V.V. Recommendation Generation Algorithms for Scheduling of Data Exchange through a Channel with Centralized Control. *Journal of Computer and Systems Sciences International*, 2007, 46(6), 887–894.

**12** Balashov, V.V., Shestov, P.E. Recommendation Generation for Providing Compatibility of Requirements to Data Exchange over a Bus with Centralized Control in Real-Time Embedded Systems. Proc. 4th International Conference on Parallel Computations and Control Problems, Moscow, Russia, 2008, 1385–1404.

**13** Balashov V.V., Kostenko V.A., Smeliansky R.L. A Tool System for Automatic Scheduling of Data Exchange in Real-Time Distributed Avionics Systems. In Proc. of the 2nd EUCASS European Conference for. Aerospace Sciences, Brussels, Belgium, 2007.