Hierarchical event-based control of multi-robot systems in unstructured environments

I Bychkov, S Ulyanov, N Nagul, A Davydov, M Kenzin, N Maksimkin
Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, 134, Lermontova str., Irkutsk 664033, Russia
E-mail: bychkov@icc.ru

Abstract. An essential requirement for achieving a high level of performance, autonomy, and reliability of a multi-robot system performing joint long-term operations in an unstructured environment is an advanced control system. In general, the robot’s control system is designed to be hierarchical and consists of several subsystems. To unify the interaction of individual components and reduce the load on computing and communication devices, we use the event-based methodology at different levels of the control system. In the paper, we demonstrate how this methodology can be applied to solve four challenging problems in robotics: cooperative formation control, path planning, missions scheduling, and action planning.

1. Introduction
Multi-robot systems are in the focus of modern research on robotics due to a wide range of actual and potential applications. To be capable of operating autonomously in unstructured environments for a long time, the robot should be equipped with an advanced control system that usually has several subsystems arranged in a hierarchy. Nowadays, considerable efforts are being made to reduce computing and communication resources required for the reliable functioning of the control system. Implementation of non-resource-intensive algorithmic and communication schemes in them allows increasing the level of robot autonomy.

One of the powerful ways to reduce the requirements to on-board equipment is to use the event-based approach, which implies that control commands of the robot are generated as a reaction to changes in the state of the group, mission conditions, and perception of the environment. To some extent, this approach is universal and can be used to solve many problems arising at different levels of hierarchy. In the paper, we consider some of them, namely, formation control, path planning, and mission scheduling. Following [1], we employ a three-level control system to solve these problems in the event-based context: formation control at the low level, path planning at the middle level, and mission scheduling at the upper level.

Formation control is the most studied problem in multi-agent systems, the research flow on which has not subsided so far. Not the last place in the studies is occupied by issues related to the reduction of computation and information exchange between robots. Traditionally, these issues are efficiently addressed by applying event-triggered control and communication strategies [2, 3, 4]. The paper proposes an event-triggered control strategy that provides a solution to the cooperative path-following problem [2, 5] in which robots are required to keep a desired
formation while following a given reference path. In contrast to the mentioned studies, we aim not only to derive control laws solving the problem, but also to optimize formation performance by synthesizing feedback gains, taking into account measurement errors, control saturation, communication delays, and variability of the reference speed. The purpose of synthesis is to minimize the steady-state pose errors of robots in formation.

Path planning in unstructured environments is a challenging problem that attracts considerable attention of specialists in robotics (see, for example, survey paper [6]). The emphasis in contemporary studies on path planning is made on developing effective algorithms with low computational costs that are applicable for real-time implementation. Despite the obvious progress in this area, not all methods are able to build paths online, taking into account robot’s dynamics and requirements to the generated paths. Most known path planning solutions indirectly or directly use iterative procedures to obtain a locally optimal path. However, when planning in an unknown environment, extensive optimization requiring significant computing resources is meaningless in many cases, since the resulting local path may not be optimal in the global sense. Thus, approaches that generate feasible paths in a short time are still in demand. In this paper, we introduce an event-based approach that allows generating safe smooth paths in an unknown environment without involving complex optimization procedures.

When a team of autonomous mobile robots has to perform a mission of a duration that exceeds their energy capacities, the mission can only be fulfilled if the robots are resupplied. The literature contains a vast amount of works on charging decision-making for automated electric vehicles and robots. Most of them focus on minimizing energy consumption while travelling through the area with several recharging stations [7, 8]. Those papers, concerning multiple vehicles competing for few recharging stations, are mainly devoted to individual behavioural approaches to ensure maximal survivability of each robot and does not utilize almost any information about the rest of the group [9]. Even those of them that are aimed to manage the whole group’s efficiency, usually consider only two parameters of each robot - current battery level and distance to the nearest charging dock [10, 11]. In this work, we are addressing the scheduling problem for the heterogeneous groups of robots, in which access priority for charging stations mostly depends on the functional capability of each robot.

To plan the actions of robots at the upper level, we are developing an approach presented at [12], which is based on automated theorem proving in the calculus of positively constructed formulas (PCFs). Tasks requiring planning are formalized in the form of non-Horn’s PCFs, which consists of a facts base that contains facts about the initial state of planning objects, as well as questions (rules) responsible for the distribution of goals for the robots and their movements to achieve these goals. Special heuristics and strategy are used to reduce the search space of the inference. It is assumed that the complete construction of the inference will accumulate facts in the refuted bases. Those facts will reflect successive changes in the state of the environment and robots and will be the basis of the plans.

Among other issues, the PCF calculus is applied to problems of designing and control of discrete-event systems (DES) which form a link between all levels of the hierarchical control system. Gaining increasing attention nowadays, the Ramage-Wonham’ theory of supervisory control (SCT) of DES is rapidly developed in the field of robots and robots group control. Supervisory control of DES makes system design more convenient by leveraging possibilities of model scalability and modifiability. Indeed, once the system model is built, various behaviours may be obtained by changing specifications and designing a corresponding supervisor that ensures a chosen specification satisfiability. Recent publications in this area concern robots cooperative exploration of an unknown environment [13], optimal planning for a set of ground robots [14], and warehouse multi-robots control [15].

The paper is organized as follows. In section 2 we present a logic-based approach to analysis and control of DES as a key component for solving a bunch of robotic control problems within
the event-based methodology. Then, we briefly describe new methods and algorithms for solving cooperative path-following (section 3), path planning (section 4), and mission scheduling problems (section 5). Finally, conclusions are given in section 6.

2. PCF-based approach to DES control

The main component of the proposed approach is DES that provides an appropriate response to detected events by analyzing data from sensors and information received from the robots. A change in the state of the DES can initiate both a recalculation of control actions at the lower level, and replanning of the path at the middle level. We employ DES in the form of finite automata to leverage explicit states representation and advantages of SCT. SCT considers an automaton as a generator of a formal language that is subject to control by forbidding some transitions to satisfy a specification. The specification is usually provided by another automaton. A detailed description of SCT can be found in [16].

We suggest application to supervisory control of DES the logical inference and automatic theorem proving (ATP) in the calculus of PCFs. The general form of PCF representing some automaton [17] consists of the single base $B = \{I(S), L(\varepsilon,S), L^m(\varepsilon,S), \delta(S_1,\sigma^j, S_2), \delta^m(S_1, \sigma^j, S_2), \Sigma_c(\sigma^i), \Sigma_{uc}(\sigma^i)\}$, $i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\}$, $n$ is the number of states, $k$ is the number of events. Here $L(s, S)$ denotes “$s$ is a current sequence of events in the state $S$” and $L^m(s, S)$ denote “$s$ is a current sequence of events in the state $S$, and $s$ is a marked string”. The first arguments of these atoms will accumulate the strings of languages generated and marked by the automaton. Predicate of the form $\delta(S_1, \sigma, S_2)$ will be interpreted as the automaton transition from state $S_1$ to state $S_2$ with an event $\sigma$. If the target state of a transition is marked, then delta atoms with an index $m$ are used, i.e., $\delta^m(S_1, \sigma, S_2)$ if $S_2$ is a marked state. The predicate $I(\_)$ denotes the initial state of the automaton. Controlled and uncontrolled events are represented in the base by predicates $\Sigma_c(\_)$ and $\Sigma_{uc}(\_)$, respectively. The function symbol “$\cdot$” denotes string concatenation, and the “$\varepsilon$” symbol corresponds to the empty string. During the inference of the PCF, languages, generated and marked by the DES, are accumulated in the PCF base. Thus, automaton as a generator of a formal language is simulated.

Using PCFs, parallel composition and product of automata may also be built. These structures are then applied for supervisory control theory problems solving. One of them is checking controllability of specifications on DES behavior. The concept of controllability plays a decisive role in SCT, since only controllable languages can be exactly achieved by the joint behavior of the controlled system and the supervisor. The way of using the PCF calculus to check if a given specification is controllable and the supremal controllable sublanguage constructing is presented in [17].

As an example, consider DES which describes the problem of robot’s path planning in an unknown environment. Suppose that the robot should follow a given reference path, leaving it to avoid collisions with encountered obstacles and returning to it after completing avoidance maneuvers (figure 1). The DES has the following states, corresponding to robot modes of operation: $PF$ (following the reference path), $DOL$ (detouring the detected obstacle from its left side), $DOR$ (detouring the detected obstacle from its right side), $NRP$ (navigation to the reference path), $SOL$ (searching for an obstacle on the left), $SOR$ (searching for an obstacle on the right). State $PF$ is marked to show that robot should always return to the reference path. Events of the DES are: $NOR$ – ‘there are no obstacles on the right’, $OF$ – ‘a detected obstacle is far’, $RPR$ – ‘the robot has reached the reference path’, and the sets of events $AL_i$, $AR_i$, $i = 1, 3, 5$, denoting switching to the obstacle avoidance modes from other modes. Some of the events $AL_i$, $AR_i$ are compositional, i.e. they determined by several atomic events, triggered by environmental conditions. For example, $AL_1$ as switching to the mode of obstacle avoidance from its left is the result of an event $eOLNF$ with one of the events $eORN$, $eORNF$, or $eON$, where $eOLNF$, $eORN$, $eORNF$, $eON$ are explained in table 1 (section 4). Let the specification
is provided by the automaton depicted in figure 2. The AUV behavior strategy that is provided by the specification language $K$ can be expressed, as follows: try not to get to a place where it is impossible to get out using standard obstacle avoidance algorithms; do not change once chosen obstacle avoidance direction until the AUV returns to the reference path (left or right-hand rule); rebuild the path only if it is vital [18]. Events $RPR$, $NOR$, $NOD$, $OF$ are uncontrollable. The prover, developed to implement the PCF calculus, verifies that $K$ is controllable, in 22 steps.

**Figure 1.** Generator $\mathcal{G}$.

**Figure 2.** Recognizer $\mathcal{H}$ for $K$.

Once the specification is proved to be controllable, the supervisor as a recognizer of the language $K$ may be designed. Its control action is then realized via parallel composition of the automata for the plant and the supervisor. Figure 3 shows the PCF that represents realization of the specification $K$ from the figure 2 for the DES from the figure 1. Here the base is \{$I_1(PF), I_2(PF), \Sigma_{uc}(OF), \Sigma_{uc}(NOL), \Sigma_{uc}(RPR), \delta_1(PF, AL_1, DOL), \delta_1(PF, AR_1, DOR), \ldots, \delta_2(PF, AL_1, DOL), \delta_2(PF, AR_1, DOR), \ldots$\} where we omit some atoms due to the lack of space. It consists of the atoms of the plant (the set $B$) and the supervisor (the set $B_S$). A distinctive feature of the PCF calculus is the ability to take into account the knowledge available in the system under consideration. The predicates $T_i$ in figure 3 are computable, and they are computed using the formulas corresponding to the transitions represented in figure 4. So the events $AL_i$, $AR_i$ may occur, i.e., appear in the following questions of the PCF, only if their triggering conditions are satisfied. This is how events are generated in the system described in section 4. For lack of space, only the first such questions are given. An asterisk used with the predicate $E$ in the last two questions shows that an occurred event should be removed from the base to exclude its reuse by the inference search machine. Thus, the PCF calculus semantics turns to a non-monotonic one what shows calculus’ variability.

3. Cooperative path-following
We consider the cooperative path-following problem for a team of $N$ vehicles whose dynamics in the horizontal plane is described [19] by

$$
\begin{align*}
\dot{x}_i &= u_i \cos(\psi_{Bi}) - v_i \sin(\psi_{Bi}), & F_i &= m_{ai}\dot{u}_i + d_{ai}, \\
\dot{y}_i &= u_i \sin(\psi_{Bi}) + v_i \cos(\psi_{Bi}), & 0 &= m_{vi}\dot{v}_i + m_{urt}u_i r_i + d_{vi}, \\
\psi_{Bi} &= r_i, & G_i &= m_r r_i + d_{ri}, & i = 1, N,
\end{align*}
$$

(1)

where $(x_i, y_i)$ are the coordinates of the $i$-th robot in a global reference frame, $\psi_{Bi}$ is the yaw angle; $u_i, v_i$ are the surge and sway speeds, $r_i$ is the yaw rate; $m_{\{i\}}$ define the mass-inertial
characteristics of the robot, which depend on the nominal mass $m_i$ and moment of inertia $I_{zi}$; $d_{i1} \cdots$ are the disturbances; $\mathcal{F}_i, \mathcal{G}_i$ are the control force and torque.

The task of the team is to maintain a desired formation while following a given reference path. We assume that the formation is organized according to the leader-follower scheme, and there is a robot in the group (called the formation leader) that is not a leader for any other robot. It must follow a virtual target (VT) which, as a point, moves along the reference path with a variable speed. Other robots as followers should hold a desired position relative to their leader(s). It should be noted that the VT is seen as a leader for the formation leader.

Let $q_i$ be a vector of variables that define the state of the robot in formation. As a rule, this vector includes relative position of the robot with respect to its leader(s) as well as its angular and progression speeds. The relative position of the robot can be specified in different ways: in a moving coordinate system associated with the leader [20] or with VT [5]; using distance and bearing angle [21]; and others. For vector $q_i(t)$, we specify a vector $\tilde{q}_i(t)$ that defines the desired motion of the robot in formation. In general, the state vector of each robot in the team in such a way that

$$\lim_{t \to \infty} || c_i^T (q_i(t) - \tilde{q}_i(t)) || < \tilde{q}_i^\infty, \quad c_i \geq 0 = \text{const},$$

(2)

where $\tilde{q}_i^\infty > 0$ defines the steady state errors.

Each robot, depending on its role in formation and mode of operation, tries to solve one of the three main tasks: keeping a desired position with respect to the leader, following the VT, and following the VT and keeping a given distance from the leader. The last one is typical when the group operates in an obstacle environment. Digital control laws with an event-triggered mechanism that solve the above mentioned tasks can be summarized as follows

$$\mathcal{F}_i(t) = \mathcal{F}_{ci}(t_k) + \mathcal{F}_{si}(t_k), \quad \mathcal{G}_i(t) = \mathcal{G}_{ci}(t_k) + \mathcal{G}_{si}(t_k), \quad t \in [t_k, t_{k+1}), \quad k = 1, 2, \ldots$$

$$\begin{bmatrix} \mathcal{F}_{si}(t_k) \\ \mathcal{G}_{si}(t_k) \end{bmatrix} = \begin{cases} \begin{bmatrix} \mathcal{F}_{si}(t_{k-1}) \\ \mathcal{G}_{si}(t_{k-1}) \end{bmatrix}^T & \text{if } \Theta(\tilde{q}_i(t_k), \tilde{q}_i^*(t_k), \ldots) = \text{false} \\ \text{sat} \left( K_i(t) (\tilde{q}_i(t_k) - \tilde{q}_i^*(t_k)) \right), \begin{bmatrix} \mathcal{F}_{si} \\ \mathcal{G}_{si} \end{bmatrix}^T & \text{otherwise}, \end{cases}$$

where $t_{k+1} = t_k + h, h$ is the control interval, $\mathcal{F}_{ci}, \mathcal{G}_{ci}$ are the feedforward terms aimed at canceling disturbances and tracking the heading direction of the team; $\mathcal{F}_{si}, \mathcal{G}_{si}$ are the feedback terms, $\Theta_i$ is the event-triggering condition which is fulfilled when the stabilization errors grow significantly, $\tilde{q}_i(t_k)$ and $\tilde{q}_i^*(t_k)$ are the latest estimates of vectors $q_i$ and $\tilde{q}_i^*$ available at $t_k$, $\overline{\mathcal{F}_{si}}, \overline{\mathcal{G}_{si}}$ are control resources in force and torque allocated for stabilization, $\text{sat}(\sigma, \bar{\sigma}) = \text{sign}(\sigma) \min(|\sigma|, \bar{\sigma})$ is the saturation function, $K_i$ is the matrix of feedback gains.

The robot as a follower can compute $\tilde{q}_i$ and $\tilde{q}_i^*$ based on either direct measurements of its relative position or information from its leader obtained via communication channels. To achieve high stabilization accuracy, the robot can also use the forward and angular speeds of the VT to determine $\tilde{q}_i^*$ [5]. The control law for the VT’s progression speed is built as a function of path
curvature when there are no obstacles detected, and as a function of the distance to the leader when the robot is detouring an obstacle.

We assume that feedback gains change in time depending on the robot’s desired angular and forward speeds, which, in turn, depend on the VT’s current speeds and the desired position of the robot in formation. Using some modifications of algorithms from [22, 23], which are based on sublinear vector Lyapunov functions, and the gain-scheduling methodology [24] allows us to synthesize feedback gains that ensure the minimum of the indicator \( q^\infty \) in (2). The interested reader is referred to [5] for a detailed description of the synthesis steps on the example of designing digital controllers (without event-triggered mechanisms) that solve the cooperative path-following problem for multiple autonomous underwater vehicles.

4. Event-based path planning

This section presents an event-based approach to path planning in an unknown environment for an individual robot. We consider a formulation of the problem in which the robot should follow a given reference path, leaving it to avoid collisions with encountered obstacles and returning to it after completing avoidance maneuvers. One needs to develop a path planning solution that generate paths that are first-order differentiable, meet kinematic constraints given by minimum turning radius \( R_{\min} \), and lie no closer than safe distance \( D_s \) from obstacles.

We assume that the robot is equipped with a multi-beam forward looking sonar (FLS) installed onboard in the horizontal plane in order to detect obstacles in the forward direction. The data obtained from the FLS can be presented as a set of pairs \((\alpha_i, \rho_i)\), where \( \alpha_i \) is the beam angle counted from the robot’s heading direction \((-\pi/2 < \alpha \leq \alpha_i \leq \pi < \pi/2)\), \( \rho_i \) is the distance to an obstacle in the beam direction, \( i = 1, N_b \), \( N_b \) is the number of beams. If no obstacles are detected in the direction of beam \( i \) or \( \rho_i \) is greater than the detection range \( \rho_d \) then \( \rho_i = \infty \).

Let \( I_L \) and \( I_R \) be the sets of left and right beams of the FLS respectively. For each obstacle point \( Q_i \), \( i = 1, N_b \), we define the maximum turning radius \( R_{\max} \) as

\[
R_{\max}^i = \rho_i \frac{\cos \alpha_i}{\sin 2\beta_i} - D_s, \quad \beta_i = \arctan \left( \frac{D_s}{\rho_i \sec \alpha_i + \tan \alpha_i} \right).
\]

It can be used to evaluate the robot’s ability to bypass the point \( Q_i \) at a safe distance \( D_s \), taking into account the kinematic constraint \( R_{\min} \): if \( R_{\min} \leq R_{\max}^i \), the robot can safely bypass the obstacle point \( Q_i \). Define also \( R_{\max}^L = \min_{i \in I_L} R_{\max}^i, \quad R_{\max}^R = \min_{i \in I_R} R_{\max}^i \), and \( \rho_{\min} = \min_{i \in I_L \cup I_R} \rho_i \).

The main component of the proposed approach is a discrete event system (DES) which is designed to detect situations that require updating the current path. Figure 4 shows a graph representation of the designed DES. Description of possible system states is given in section 2. The set of system events with their triggering conditions is shown in table 1. For certainty, we will accept \( R_{\min} = 20 \text{ m}, \quad D_s = 10 \text{ m}, \quad \Delta R = 10 \text{ m}, \quad \rho_n = 10 \text{ m}, \quad \rho_f = 70 \text{ m}, \quad N_b = 60, \quad \rho_d = 150 \text{ m}, \quad \alpha = -60^\circ, \) and \( \pi = 60^\circ \).

Changing the DES state entails the execution of actions related to building a new path. Action A1 (see figure 4) generates a path to avoid the detected obstacle using a modification [18] of the waypoint guidance algorithm [25]. Action A2 constructs a path that connects the robot’s current position with the reference path. For this purpose, we use the path planning algorithm from [18], which is based on Dubins curves [26]. The path-following controller designed in section 3 is employed to drive the robot along the generated paths.

Figure 5 shows simulation results for the robot operating in a complex obstacle environment. In the figure, the blue areas are obstacles scattered on the scene, and the red (blue) line correspond to the robot’s reference (actual) path. Analysis of the obtained results allows us to conclude that the proposed approach, together with the path planning algorithms, can generate feasible paths in real-time and has a potential to be implemented on board.
DOL
PF
DOR
NRP
SOR
SOL
MS
MC
\[eOLNF & (eORN or eORNF or eON)\]: A1
eEP
eRPR
\[eOLNF or eON\]: A1
\[eORNF & \ldots or eOLNF\]: A1
eNOR: A2
(eNOL): A2
\[eOLNF & (eORN or eORNF or eON)\]: A1
\[eORNF & (eOLN or eON)\]: A1

**Figure 4.** DES for the path planning solution

| Name   | Triggering Condition | Description                                   |
|--------|----------------------|-----------------------------------------------|
| eOLN   | \( R_{\text{max}}^L < R_{\text{min}} \) | The obstacle detected on the left is near     |
| eOLNF  | \( R_{\text{min}} \leq R_{\text{max}}^L < R_{\text{min}} + \Delta R \) | The obstacle detected on the left is not far  |
| eNOL   | \( R_{\text{max}}^L = \infty \) | There are no obstacles on the left            |
| eORN   | \( R_{\text{max}}^R < R_{\text{min}} \) | The obstacle detected on the right is near    |
| eORNF  | \( R_{\text{min}} \leq R_{\text{max}}^R < R_{\text{min}} + \Delta R \) | The obstacle detected on the right is not far |
| eNOR   | \( R_{\text{max}}^R = \infty \) | There are no obstacles on the right           |
| eON    | \( \rho_{\text{min}} < \rho_n \) | The detected obstacle is near                 |
| eOF    | \( \rho_{\text{min}} > \rho_f \) | The detected obstacle is far                  |
| eERP   |                       | The robot has reached the end of the reference path |
| eEAP   |                       | The robot has reached the end of the avoidance path |
| eRPR   |                       | The robot has reached the reference path      |

**Table 1.** Events of the DES.

Due to space limitations, we presented here a shortened and elaborated version of the DES from [18]. In the new DES, we preserved the most important aspects of the original system,
5. Mission scheduling scheme

Large-scale multi-vehicle robotic operations in different dynamic environments commonly involve long-stay activities of vehicles within the operational area. Since it is not always possible to replace low-battery robots on schedule, a number of available charging stations are required for the long-term mission success. Hence, an intelligent recharging schedule must be constructed to maintain efficient and uninterruptible mission implementation. A mission planning system capable of doing this should be both flexible and reliable to ensure a quick reaction on any unforeseen events. An additional significant limitation concerning the group strategy planning is a low range of inner-vehicle communication channel. Summarizing the above, two following problems have to be handled at once: to find the reasonable vehicle recharging schedule over the extended time period and to ensure the regularity of communication sessions within the group.

We denote by $T$ the mission length. Let $n$ be the overall size of robotic group committed to conduct the operation. Each vehicle $k \in \{1, 2, ..., n\}$ in the group is characterized by its speed $v^k$, battery capacity $b^k$, current battery level $b(t)^k \leq b^k$ and set of research equipment $u^k_j \in \{0, 1\}, j = 1, ..., l$. Hence, the working group is both parametrically and functionally heterogeneous.

Battery limitations force robots to periodically recharge at one of $w$ given charging stations. We denote the distance from the operational area to each station as $d_i, i = 1, ..., w$ and the average charging speed for all vehicles by the constant parameter $c \geq 1$. So, full charging time for an empty battery with a capacity of $b$ hours would be $b/c$ hours. Here we rate batteries not in energy units but as average run-time on cruising speed.

As robots are not obliged to leave the group only being completely discharged or to always recharge up to the entire full battery (although it is preferable), we can line up charging periods for each robot during the mission in order to obtain the desirable group rotation schedule:

1. The main requirement is to maintain each robot in good working order by organizing well-timed recharging for all vehicles in need with consideration of charging docks limitation.
2. Then, the working (non-charging) group should always be functionally able to perform the mission by having access to all available types of onboard equipment if possible.
3. Finally, we want to exclude as far as possible the simultaneous mass-charging of the group majority since it reduces the performance capability of the remaining team.

In order to ensure the event-responsiveness of the group as a whole, robots should coordinate in a way to periodically establish group communication sessions for inter-vehicle data exchange. As real communication channels are typically slow and limited in range, we propose using the rendezvous approach [27], when all robots periodically arrive at the specified location on a schedule. As we consider group rotation activities (robots leaving the group for recharging or joining back after) also as condition-changing events, we suggest jointing them with the group rendezvous to treat these moments as mission decomposition points. In that event, the group workflow can be represented as a sequence of the group operating periods with the group rendezvous at the end of each period (figure 6).

In case of any unexpected events, the decomposition scheme should be adjusted during the nearest rendezvous. The proposed scheme allows us to treat the action-planning problem for
each operating period as a problem for a robotic group with a static lineup. It also leads us to the two additional conflicting requirements for the group schedule:

(4) The period between successive rendezvous should never exceed the required periodicity $P$.
(5) Since each rendezvous distracts robots from their research work, we want to minimize the rendezvous frequency.

Thus, the upper-level mission planner strategy is to manage group operating periods by scheduling robots’ recharging cycles. As mission length $T$ is supposed to be a big value, we suggest using search space discretization $T = \langle T^1, ..., T^z \rangle$, $z = T/T_0$, where $T_0$ is a duration of each interval. Thus, the working schedule of a single robot can be denoted by a binary $z$-dimensional vector, where each attribute $h$ stands for a robot’s status on a corresponding time interval (0 if the robot is working, and 1 otherwise). A schedule of $i$-th vehicle is considered to be feasible if none working period (continuous sequence of $x_0$-attributes) lasts longer than that vehicle’s battery level at the beginning of that period $x\cdot T_0 \leq b(t_x)^i$. The group schedule at this point can be represented as $(n + 1)$-by-$z$ binary matrix $H = \{h_{ij}\}$, where first $n$ rows represent single vehicle’s schedules, and the last row encodes those rendezvous that are initiated by communication needs.

We regard requirements (1, 4) from the list above as the feasibility criteria and (2, 3, 5) as the efficiency criteria for the planning problem. For the efficiency criteria, we propose using a sum of two specific functions: $f(H) = f_G(H) + f_R(H) \rightarrow \min$. The first function $f_G(H)$ evaluates given schedule $H$ in respect of the group performance capability losses during the mission progress, while the second one $f_R(H)$ keeps track of the rendezvous frequency [28].

In general, the scheduling problem is NP-complete; therefore, the use of heuristic methods to solve it seems the most reasonable. As the mission scheduler should be simple, reliable, and fast, we propose using an evolutionary approach with few modified steps. At first, we use a compressed representation $G = \langle g_1, ..., g_z \rangle$ of the matrix schedule $H$ as the chromosome to drastically decrease the problem size. $G$ is a $z$-sized vector, where each attribute $g_i$ encodes a vehicle that initiates a rendezvous by either leaving or rejoining the group, depending on its current status. Then, we suggest using two different constructive heuristics to generate high-quality initial populations. To create new solutions, we use both random and swap mutation alongside with two competing crossovers: classical two-point and POX-crossover, which are widely recognized as the most efficient ones for a variety of scheduling problems. Also, we propose using a specialized neighbouring search operator designed to merge closely-spaced rendezvous points. An in-detail description of the proposed evolutionary algorithm may be found in [28].

The high efficiency of the suggested approach is shown through a series of simulation studies. It is proved to be fast, accurate, and reliable as it allows fast construction of group schedules within the 5% of the optimal value even for the large-sized cases. Figure 7 shows simulation results for a test instance with a heterogeneous group of four robots.
6. Conclusions
The paper addresses several multi-vehicle control problems arising at different levels of a hierarchical control system of autonomous robotic groups. All problems discussed in this work are treated as event-based to ensure their compatibility within the same event-driven environment. We have proposed a number of different methods to answer these issues and have individually validated every one of them to be reliable and efficient. We have also suggested a general concept of a three-level framework to combine the proposed approaches in an intelligent way. It is still open to various possible developments and could be easily integrated into the real-world robotic systems, which is supposed to be the next step in our study.

7. Acknowledgments
The research was partly supported by the RFBR, project no. 20-07-00397 (sections 2) and project no. 19-08-00746 (sections 3), and by the Basic Research Program of SB RAS, project no. IV.38.1.1.

References
[1] Bychkov I, Davydov A, Kenzin M, Maksimkin N, Nagul N and Ul’yanov S 2019 2019 International Siberian Conference on Control and Communications (SIBCON) pp 1–6
[2] Jain R P, Aguiar A P and de Sousa J B 2018 IEEE Robotics and Automation Letters 3 1941–1948
[3] Chu X, Peng Z, Wen G and Rahmani A 2018 Neurocomputing 275 121 – 131 ISSN 0925-2312
[4] Xu P, Li W, Tao J, Dehmer M, Emmert-Streib F, Xie G, Xu M and Zhou Q 2020 IEEE Access 8 167288–167299
[5] Ulyanov S and Maksimkin N 2019 Mathematics in Engineering, Science and Aerospace (MESA) 10 487–500
[6] Panda M, Das B, Subudhi B and Pati B B 2020 International Journal of Automation and Computing 17 321–352 ISSN 1573-7497
[7] Zhang S, Gajpal Y, Appadoo S and Abdulkader M 2018 International Journal of Production Economics 203
[8] Yi Z and Shirk M 2018 Transportation Research Part C: Emerging Technologies 86 37–58
[9] Sempé F, Muñoza A and Drogoul A 2002 Distributed Autonomous Robotic Systems 5 ed Asama H, Arai T, Fukuda T and Hasegawa T (Tokyo: Springer Japan) pp 91–100
[10] Kannan B, Marmol V, Bourne J and Dias M 2013 pp 3503–3510 ISBN 978-1-4673-5641-1
[11] Zhang Z, Pi B, Ma X, Wang Z and Xing L 2020 pp 232–236
[12] Davydov A and Larionov A 2020 2020 7th International Conference on Control, Decision and Information Technologies (CoDIT) vol 1 pp 727–732
[13] Dai X, Jiang L and Zhao Y 2016 Applied Intelligence 45 18–29 ISSN 1573-7497
[14] Hill R C and Laforetine S 2017 American Control Conference (ACC) pp 3840–3847 ISSN 2378-5861
[15] Tatsumoto Y, Shiraishi M, Cai K and Lin Z 2018 Control Engineering Practice 81 97 – 104 ISSN 0967-0661
[16] Wonham W M and Cai K 2019 Supervisory Control of Discrete-Event Systems (Springer International Publishing)
[17] Davydov A, Larionov A and Nagul N 2020 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO) pp 1151–1156
[18] Ulyanov S, Bychkov I and Maksimkin N 2020 Applied Sciences 10 7894 ISSN 2076-3417
[19] Lapiere L and Soetanto D 2007 Ocean Engineering 34 1734–1744 ISSN 0029-8018
[20] Millan P, Orihuela L, Jurado I and Rubio F R 2014 IEEE Transactions on Control Systems Technology 22 770–777 ISSN 1063-6536
[21] Ulyanov S and Maksimkin N 2017 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) pp 1032–1037
[22] Kozlov R I and Kozlova O R 2009 Journal of Computer and Systems Sciences International 48 262–271 ISSN 1555-6530
[23] Vassilyev S, Ulyanov S, Kenzin M and Maksimkin N 2017 AIP Conference Proceedings pp 020170(1)–020170(10)
[24] Leith D J and Leithead W 2000 International Journal of Control 73 1001–1025
[25] Yan Z, Li J, Zhang G and Wu Y 2018 Sensors 18 ISSN 1424-8220
[26] Dubins E L 1957 American Journal of Mathematics
[27] Zadeh S M, Powers D M W and Yazdani A M 2016 CoRR abs/1605.01824 (Preprint 1605.01824) URL http://arxiv.org/abs/1605.01824
[28] Kenzin M, Bychkov I and Maksimkin N 2020 2020 7th International Conference on Control, Decision and Information Technologies (CoDIT) vol 1 pp 745–750