

A Recovery Method for the Robotic Decentralized Control System with Performance Redundancy

Iakov Korovin¹, Eduard Melnik², and Anna Klimenko³(✉)

¹ Southern Federal University, Rostov-on-Don, Russia

² Southern Scientific Center of the Russian Academy of Sciences (SSC RAS),
Rostov-on-Don, Russia

³ Scientific Research Institute of Multiprocessor Computing Systems,
Southern Federal University, Taganrog, Russia
anna_klimenko@mail.ru

Abstract. The fault of the robotic control system is critical and leads to the general system failure, while autonomous robots have to gain their aims without any maintenance. Contemporary academic studies propose decentralized control systems as prospective from the robustness point of view. On the other hand, a performance redundancy allows to optimize resource utilization and improve the fault-tolerance potential of the control system. This paper is devoted to the recovery method of the robotic decentralized control system with performance redundancy. A reconfiguration problem has been formalized, decentralized method of the solution obtaining is represented. Also some simulation results are given and discussed.

Keywords: Decentralized control system · Robustness · Fault-tolerance · Simulated annealing · Autonomous robots control

1 Introduction

Fault-tolerance is extremely important for autonomous robotic systems. The large amount of them is performing their tasks in hazardous and aggressive environments, where a man is not supposed to be located. Besides, autonomous robots perform and must achieve their goals without any repair for a long terms of time.

In robotics, some classifiers of failures are proposed. For example, in [1] the general failure levels are concerned: mechanical level (a joint becomes lock); hardware level (sensor does not perform properly); controller level; controlling computer level.

Another classification of failures is described in [2]: sensors; effectors; communications; power system; control system.

Human and physical faults as a cause of failure are distinguished in [3], where the detailed taxonomy (Fig. 1) also can be found.

In the scope of this paper the robot control systems are under consideration. Contemporary control systems are the software and hardware complexes, where distributed computing paradigm is used widely. The monitoring and control tasks (MCTs)

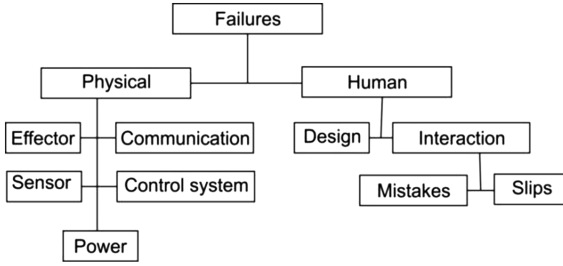


Fig. 1. Failures taxonomy

within the control system are performed by computational units (CUs). Each MCT allocates some computational resources, and each CU performs more than one MCT. The control system architecture from the management point of view can be centralized, hierarchical or decentralized. Some works show that the decentralized control system is potentially more fault-tolerant than others [4], but needs additional research in the fields of cooperative problem solving, multiagent systems, etc.

Author of [5] defines a fault-tolerant control system as a control system, which is able to automatically maintain the system stability and an acceptable performance when component failures occur. To gain these objectives, following principles must be implemented: fault detection; fault isolation; fault identification; fault recovery.

Usually, passive or active recovery methods are used [2]. Within the active recovery methods two main reconfiguration strategies are used. The first one proposes using of some pre-defined control laws, the second one is oriented to the on-line synthesis of the system controller with respect to the fault identification.

In the scope of this paper recovery method for the decentralized control system with the performance redundancy will be considered. The next section contains the brief explanation of the performance redundancy in comparison to the structural one. Section 3 is devoted to the reconfiguration problem formalization with the graceful system degradation objective. Section 4 contains reconfiguration method, and, at last, Sect. 5 presents some experimental results and discussion.

2 Performance Redundancy and Decentralized Dispatching

Structural redundancy is widely used nowadays [4]. As mentioned in [5], redundancy is the key ingredient in any fault-tolerant systems. Almost all of modern aircraft such as Boeing 777 and Airbus A320/330/340 have used triplex- or quadriplex-redundant activation systems, flight control computer and databus systems [6, 7].

Performance redundancy considers all CUs as performing elements with some performance reserve. Advantages of performance redundancy are explained in details in [8–10].

The way of ICS dispatching is important too: the centralized dispatching has multiple drawbacks, in particular, the main dispatcher fault is the cause of the entire system failure without the possibility to recover.

Decentralized dispatching of the ICS operates with equal control elements. Each CU is controlled by its own software agent (Fig. 2), which operates as a kind of MCT. In the case of CU failure (software or hardware) agents of operational nodes begin a recovery procedure via reconfiguration: MCTs from the faulted node can be launched by the operational ones.

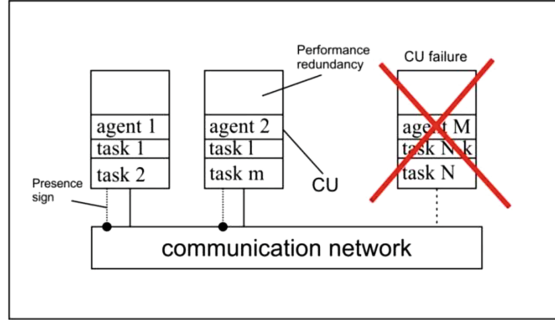


Fig. 2. Performance redundancy and software agents representing the CUs

ICS with decentralized dispatching and performance redundancy has good recover possibilities, but requires the design and implementation of the cooperative recovery methods and algorithms.

3 Reconfiguration Problem Formalization

Let the input data be the following:

- A set of MCTs $G = \{x_{ij}\}$, $i = 1 \dots N$, where x_i – the size of task i , N – the number of tasks.
- Let $G = G_c \cup G_{nc}$, $G_c \cap G_{nc} = \emptyset$, where G_c – a subset of critical MCTs, G_{nc} – a subset of non-critical ones. Non-critical MCTs can be eliminated from the system during reconfiguration. The number of critical MCTs is N_c , and the number of non-critical MCTs N_{nc} ;
- Let G_f be the set of MCTs from the faulted CU. $G_f \subseteq G$, G_p is the performing tasks, $G_p \subseteq G$, $G_p \cap G_f = \emptyset$.
- A planned completion time for the set G is T_{plan} .
- Number of CUs is M with the performance p .

Let's take into consideration that we have to allocate the MCTs from the set G_f within the system of operational CUs, on which the tasks from the set G_p are allocated with the constraint of completion time T_{plan} . Let the resource allocated by CU j for the subtask i be λ_{ij} . The tasks allocation before the failure is described by matrix R :

$$R = \begin{vmatrix} r_{11} & r_{12} & r_{1M} \\ \dots & \dots & \dots \\ r_{N1} & \dots & r_{NM} \end{vmatrix}, \quad (1)$$

where $r_{ij} = f(\frac{x_i}{\lambda_{ij}p})$, $f(\frac{x_i}{\lambda_{ij}p}) = \begin{cases} \frac{x_i}{\lambda_{ij}p}, & \text{if } x_i \text{ is running on CU } j, \\ 0, & \text{otherwise.} \end{cases}$

Let the failure occurred on the CU with number d . The column d of matrix R is deleted, so there is $M - 1$ columns and $N - |G_f|$ lines in the new matrix R_f . Renumber the elements of R_f in the following way, saving the indexes from the matrix R in the upper positions:

$$R_f = \begin{vmatrix} r_{11}^{ij} & r_{12}^{ij} & r_{1(M-1)}^{iM} \\ \dots & \dots & \dots \\ r_{11}^{Nj} & \dots & r_{(N-G_f)(M-1)}^{NM} \end{vmatrix}. \quad (2)$$

R_f describes the system state before the reconfiguration and contains the allocation of the operational tasks among the operational CUs. R_r will be the allocation of the task set G on the $M - 1$ CUs. Formally, the subset G_f will be added to the G_p with the number of $CU_s = M - 1$:

$$R_r = \begin{vmatrix} r_{11}^{ij} & r_{12}^{ij} & r_{1(M-1)}^{iM} \\ \dots & \dots & \dots \\ r_{11}^{Nj} & \dots & r_{(N)(M-1)}^{NM} \end{vmatrix}; \quad (3)$$

$$r_{ij}^{kl} = f(\frac{x_i}{\lambda_{ij}p})g(x_i);$$

$$g(x_i) = \begin{cases} 0, & \text{if } x_i \in G_{nc} \text{ and eliminated from the system,} \\ 1, & \text{otherwise.} \end{cases} \quad (4)$$

Let's consider matrix Ψ :

$$\Psi = [\varphi(x_{1j}^{kl}) \quad \varphi(x_{2j}^{kl}) \quad \dots \quad \varphi(x_{Nj}^{kl})] \quad (5)$$

where $\varphi(x_{1j}^{kl}) = \begin{cases} 0, & l = j, \\ \xi, & \text{otherwise} \end{cases}$ k, l — the saved indexes of matrix R , j — the number of CU in matrix R_r , ξ — the integer number.

The matrix Ψ describes if the MCT x_i was relocated from CU l to CU j .

The first objective function can be written in the following manner:

$$F_1 = \sum_{i=1}^N \varphi(x_{ij}^{kl}) \rightarrow MIN. \quad (6)$$

The next objective function component is load balancing which can be written in the following way.

$$F_2 = \left(\sum_{i=1}^N \lambda_{ik} - \sum_{i=1}^N \lambda_{il} \right) \rightarrow \text{MIN}, \forall k, l, \lambda_{ik} \in R_r. \quad (7)$$

In other words, we need to find MCTs allocation with respect to load balancing objective function. While the desirable option is to deliver the graceful system degradation, it is useful to keep running as much MCTs as possible. The maximum number of non-critical tasks running equals to the maximum summa of all $g(x_i)$ in the matrix R' .

The last objective function component will be as following:

$$F_3 = - \sum_i^N g(x_i) \rightarrow \text{MIN}. \quad (8)$$

Herewith the time constraint must be satisfied:

$$\forall j : \sum_{i=1}^N r'_{ij} \leq T_{plan}, j \in [1 \dots M]. \quad (9)$$

Let's put the current multicriteria optimization problem to the following form:

$$F = \sum_{i=1}^N \varphi(x_{ij}^{kl}) \rightarrow \text{MIN}, \quad (10)$$

$$\left(\sum_{i=1}^N \lambda_{ik} - \sum_{i=1}^N \lambda_{il} \right) \leq \gamma; \quad - \sum_i^N g(x_i) \leq \mu; \quad \forall j : \sum_{i=1}^N r'_{ij} \leq T_{plan}, j \in [1 \dots M],$$

$x_i > 0$; $0 < \lambda_{ij} < 1$, where $0 < \gamma < 1$ is the assumed level of load dispersing, μ is the integer number.

4 Cooperative Problem Solving

The problem formalized earlier contains a kind of k-partition problem (or bean-packing problem) which is NP-hard, so there is no polynomial algorithms for the solution obtaining. In the scope of this research the simulated annealing (SA) with the “quenching” temperature is used [11] to reach an acceptable solution in a reasonable time.

With the shortage of time and the using of decentralized control, it is appropriate to initiate a search for a new system configuration at all operational nodes (which are represented by the agents, Fig. 2).

After CU or MCT fault detection and identification (which are out of this paper's scope), the reconfiguration is initialized, and every performing agents launches the new configuration search.

As soon as one of the agents finds allowable configuration, solving, in fact, a constraint satisfaction problem, it notifies other agents, which take the solution found as a new configuration proposed to perform.

Here we have to make some assumptions. The system of agent is synchronous in terms of work [12]. If one agent sends a message, agent-addressee receives it without delay. There is no message losses in the communication network. The model of communication network is fully connected graph. Then, the next assumption takes place: some agents can broadcast incorrect solution as a result of search. So, the cooperative configuration search method must have a kind of mechanism to prevent the further usage of unviable solution.

Each agent also must have a queue (Q) for the incoming messages and a queue (S) for the viable solution. Besides, we assume every agent know the constraints for the MCTs: launching time spans, data transfer interconnections, etc.

Generalized method based on a simulated annealing for one agent is represented below:

1. Set the initial parameters: temperature, quenching ratio, etc.
2. Generate solution R in a random manner.
3. If Q contains any solutions, go to the 4.
4. Beginning of the cycle
 - 4.1. If Q contains any solutions, go to the 5:
 - 4.2. Generation of new solutions: R.
 - 4.3. Calculate the value of F.
 - 4.4. Check the admissibility of F
 - 4.5. If the current solution is acceptable, go to 5.
 - 4.6. Temperature correction. Go to step 4.
5. Broadcast the solution reached.
 - 5.1. Range the solutions in the Q.
 - 5.2. Verify the best solution. If the verification is successful, go to 6.
 - 5.3. If the solution is unviable, delete it from Q. Go to 5.2.
6. Broadcast the verified solution.
7. Choose the most frequent viable solution from S for the execution.
8. End.

The cooperative method described above allows every agent to have all solutions in Q after at least one agent found a solution. Verification process contains the check of constraints for the MCTs. The S queue contains solutions estimated as "viable". We assume that if solution S_1 was accepted by N_1 agents, and solution S_2 was accepted by N_2 agents, S_1 is "viable" if $N_1/N_2 = 2$ [12].

5 Simulation Results and Brief Discussion

Taking into consideration the shortage of time, the first simulation study is in the field of solution obtaining speed. For this study SA with Boltzmann generation rule and quenching ratios 0.9; 0.8; 0.7 was used (Fig. 3).

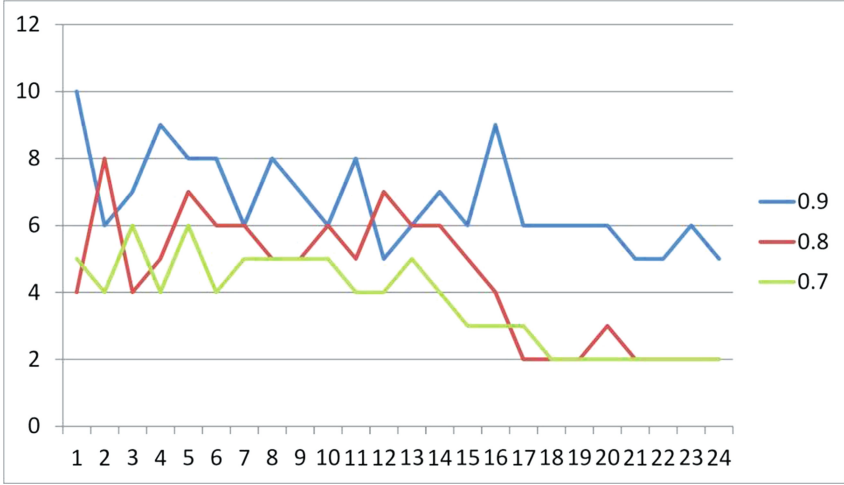


Fig. 3. SA with “quenching” temperature scheme convergence speed

The results of pilot simulation allow to affirm that some local minimas can be reached fast enough (10–20 iterations). It makes SA a perspective search method even in the time shortage circumstances (2 eliminated non-critical tasks as a result).

Next simulation is made for the different number of calculating agents (5;10) with the initial number of MCTs = 50 (Fig. 4).

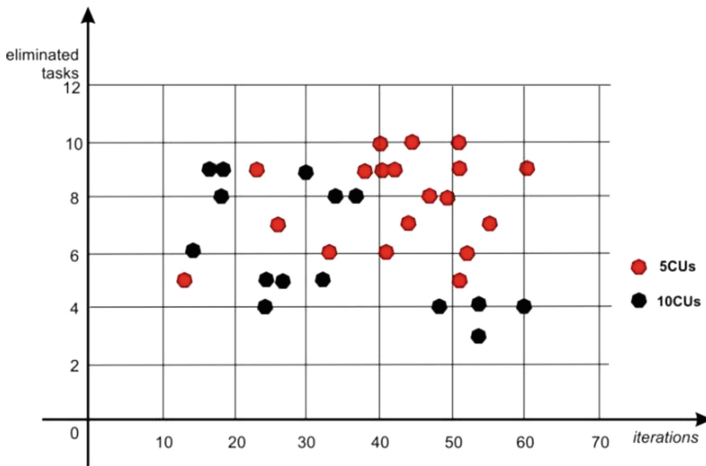


Fig. 4. The number of eliminated MCTs within 5 and 10CUs cooperative problem solving

It is seen, that with the increasing of agents number the quality and the speed of the solution obtaining becomes better. The best result for the 5 agents is 5 eliminated non-critical tasks, while the best result for the 10 agents is 3 lost tasks.

6 Conclusions

Fault-tolerance is one of the important aspects for the autonomous robots. An enormous field of tasks is performed without man's assistance, so the robots should be reliable, viable and fault-tolerant.

In the scope of this paper the decentralized robotic control system recovery is considered. A reconfiguration problem is formalized, and a method for the cooperative problem solving is proposed. A method represented bases on the parallel multistart SA and provides some degree of robustness in the circumstances of incorrect agent behavior. It must be noted, that SA with "quenching" temperature scheme can be "fast" enough to find local minimas. Also, a group of searching agents improve solution quality significantly. For example, if one agent can find a local minima of unaccepted quality, another one, with different initial computational point, can find the acceptable solution. It must be noted, that the redundancy must be sufficient.

The future work is proposed to be directed to the field of robust distributed algorithms and one's modelling and efficiency estimation.

Acknowledgements. The reported study was funded by SSCRAS project 0256-2014-0008 within the task 007-01114-16 PR and by RFBR projects 14-08-00776-a and 15-37-20821-mol-a-ved.

References

1. Gini, M., Smith, R.: Monitoring robot actions for error detection and recovery. In: Proceedings of the Workshop on Space Telerobotics, vol. 3, p. 67 (1987)
2. Crestani, D., Godary-Dejean, K.: Fault tolerance in control architectures for mobile robots: fantasy or reality?. In: 7th National Conference on Control Architectures of Robots, Nancy, France (2012)
3. Carlson, J., Murphy, R.R.: Reliability analysis of mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2003), pp. 274–281 (2003)
4. Kalyaev, I.A., Melnik, E.V.: Decentralized Systems of Computer Control. Publishing SSC RAS, Rostov-on-Don (2011)
5. Zhang, Y., Jiang, J.: Bibliographical review on reconfigurable fault-tolerant control systems. *Ann. Rev. Control* **32**(2), 229–252 (2008)
6. Bartley, G.F.: Boeing B-777: fly-by-wire flight controls. In: The Avionics Handbook. CRC Press, Boca Raton (2001)
7. Biere, D., Favre, C., Traverse, P.: Electrical flight controls, from Airbus A320/330/340 to future military transport aircraft: a family of fault-tolerant systems. In: The Avionic Handbook. CRC Press, Boca Raton (2001)
8. Melnik, E.V.: Simulation options for redundancy in distributed information and control systems with a decentralized organization. In: Proceedings of SFU, SER Technical science, no. 3, pp. 184–193 (2013). (In Russian)

9. Melnik, E.V.: Principles of organization of the decentralized network-centric information management systems. Herald Comput. Inf. Technol. **4**, 25–30 (2013). (In Russian)
10. Melnik, E.V.: Effect processor computational load balancing devices in highly distributed information management system. In: Mechatronics, Automation, Control, pp. 29–35 (2012). (In Russian)
11. Ingber, L.: Simulated annealing: practice versus theory (1993). <http://citeseer.uark.edu:8080/citeseerx/viewdoc/summary?doi=10.1.1.15.1046>
12. Tel, G.: Introduction to Distributed Algorithms, pp. 1–608. Cambridge University Press, Cambridge (2000)

Interactive Collaborative Robotics

First International Conference, ICR 2016, Budapest,

Hungary, August 24-26, 2016, Proceedings

Ronzhin, A.; Rigoll, G.; Meshcheryakov, R. (Eds.)

2016, IX, 254 p. 126 illus., Softcover

ISBN: 978-3-319-43954-9