Computer Science • 13 (3) 2012 http://dx.doi.org/10.7494/csci.2012.13.3.89

Bartosz Kryza
Dariusz Król
Michal Wrzeszcz
Lukasz Dutka
Jacek Kitowski

INTERACTIVE CLOUD DATA FARMING ENVIRONMENT FOR MILITARY MISSION PLANNING SUPPORT

Abstract *In a modern globalised world, military and peace keeping forces often face situations which require very subtle and well planned operations taking into account cultural and social aspects of a given region and its population as well as dynamic psychological awareness related to recent events which can have impact on the attitude of the civilians. The goal of the EUSAS project is to develop a prototype of a system enabling mission planning support and training capabilities for soldiers and police forces dealing with asymmetric threat situations, such as crowd control in urban territory. In this paper, we discuss the data-farming infrastructure developed for this project, allowing generation of large amount of data from agent based simulations for further analysis allowing soldier training and evaluation of possible outcomes of different rules of engagement.*

Keywords data farming, cloud, virtualisation, web 2.0, mission planning support

1. Introduction

Mission planning support in military applications is a very complex process, involving several issues, such as analysis of physical environment of the incident, social and cultural aspects of the location and definition of proper Measures of Effectiveness for assessment purposes. In particular, asymmetric threats in urban territory, which involve an operation of a relatively small group of soldiers or police forces within a city with civilian population ranging from neutral to hostile, require very careful planning in order to maximize the chances of positive outcome. Due to possibly large amount of civilians it is very difficult to predict the possible outcomes of different rules of engagement. This task can be supported by computer agent based simulation system, which allows the higher echelons to assess different strategies for the operation. However, in order to get meaningful data from the system, first a significant number of runs with different parameters must be performed. Obviously this is infeasible, requiring both an extreme amount of computing power and time, while in military applications it is often critical to have even rough assessment of the strategy and possible outcomes in predetermined amount of time.

In this paper we present a part of a system which was developed within the EUSAS (European Urban Simulation for Asymmetric Scenarios) project, which is financed by 20 nations under the Joint Investment Program Force Protection of the EDA. During the EUSAS project we developed a new modelling approach to human decision making [7] and proposed a combination of object-oriented and ontology-based approaches for real-time interworking of human behaviour models in the context of agent-based simulation systems [11]. Afterwards, we evaluated agent-based simulation platforms [12] to choose the one that best suits the project needs. The results of this work were used to create highly realistic agent based simulations of military missions. These simulations are the core of our system which is composed of two parts: serious game part which aim is support of the training of policemen and soldiers and data farming part which contains infrastructure and software that executes simulations to provide framework that supports military mission planning. In this paper we describe the data farming part.

The aim of the data farming part is to manage data farming experiments (DF experiments). DF experiment includes several executions of MASON [13] based agent simulation (of military operation in urban territory) with different values of parameters and gathering of logs and results. The implementation of the agents logic (both civilians and soldiers) and environment is based on advanced models, which require each run of the simulation to run for several minutes, taking on the input configuration file with all parameters set to proper values and providing on the output a log file with all events within the simulation relevant for further analysis. Our data farming systems goal is to provide the researcher with the ability to minimize the time (and computational cost) of the simulation by limiting the number of simulation runs necessary for obtaining relevant results as well as dynamically monitoring the intermediate results and fine tuning the parameter space on the fly.

The rest of the paper is organized as follows. In Section 2, we describe related work. Section 3 contains description of user requirements. In section 4, we present data farming overview from the user point of view. Then, in section 5, we show the architecture of our system and the user interface in section 6. We conclude this paper in section 7.

2. Related work

The concept of data farming was invented in 1998 [3, 8]. It may be used in any area where the research process contains following steps [8]:

- question/topic research and definition,
- model development and gaming,
- parameter space exploration,
- data exploration and analysis.

Since 1998, several tools have been developed. They can be split into three categories [8]:

- implementations of data farming environments,
- distillation modelling environments,
- data exploration tools.

Two very popular tools are the Maui High Performance Computing Parallel Execution System (PES) and OldMcData. They integrate following distillation modelling environments: ISAAC, Socrates, Pythagoras, Mana, PAX and NetLogo. To simplify the analysis, several visualization tools have been created. Three very popular ones are: the Playback Tool, the VizTool Landscape Plotter and Avatar.

The data farming is a powerful instrument, especially for military users. Several articles describing military application of data farming were created e.g. marine corps applications of data farming [5], SDF meta-technique [4] or "Data Farming and Defense Applications" by Horne and Meyer [9]. However, different military applications demand different techniques and tools. The main aspect that differentiates the EUSAS project from described solutions is that our systems offers users ability to interact with the system during the experiment to analyse partial results or to extend the range of investigated parameters. Attempts of using Grids to perform interactive and real-time applications have been described in [6], [14] and [10]. Another very important innovation of our system is the attempt to make the system totally independent from infrastructure to provide the user ability to use sources of computing power on demand (e.g. Clouds).

3. Requirements for mission planning support system

Users of different system parts have different requirements. The users of the serious game part (soldiers and policemen) need the system that will allow them to perform virtual missions using their computers. The environment shown on the screens should

realistically imitate actual locations. Furthermore, the system should allow users to cooperate during mission training. Users of the data farming part need the system that will allow them to simulate many possible mission scenarios to verify some rules of engagement. Although, requirements of serious game users and data farming users are very different, users of both parts of the system demand high reality of behaviour of computer agents so advanced psychological models have to be used. In this chapter we present the requirements of users of the data farming part of the system.

Military users involved in mission planning want to simulate the mission and check if soldiers are able to perform this mission with some given rules of engagement. The data farming allows users to achieve this goal successfully, since it gives them ability to check many factors which may influence the mission. The data farming experiment includes several executions of simulation with different values of parameters that represent these factors. If all simulations executed with different values of parameters end successfully, there is high probability that the real mission will also end with success. Otherwise, the users will know which factors are crucial and should be monitored. Furthermore, the users want to be able to choose ranges of tested parameters to determine the conditions under which the mission will be performed (e.g. to determine behaviour of civilians, number of agents etc.). The aim of the data farming is generation of large amount of data that may be analysed by external tools which are able to answer users questions. In the EUSAS project, the MASDA algorithm [2] is used to find which sequences of actions lead soldiers to success and after which sequences of actions soldiers may be hurt or killed.

Time is very important for the military users so they have to be able to check predicted time to finish the experiment. If this period of time is not satisfying, they require ability to speed up the experiment. Moreover, the users expect ability to check partial results as soon as possible. They also require possibility to modify the experiment during the runtime. Especially—to extend the parameters space when needed.

The user requirements imply some problems that have to be solved by providers of the infrastructure and software. The number of possible parameters combinations can be very large (ca. 10^{10}). Moreover, each run of the agent based simulation may take several minutes because one simulation may contain thousands of agents which have to use advanced psychological models. This is why one personal computer needs $o(10^5)$ years to execute one large data farming experiment. We use Cloud as a source of computing power accessible on demand. To give the user ability to speed up the experiment we use virtualisation of the resources and a user interface that allows creating new virtual machines anytime. However, even the most powerful infrastructure is not able to execute all possible combinations of a large data farming experiment in acceptable time by military users. Hence the reduction of parameters space have to be done before the experiment start, however, this reduction is a challenge because the number of executed combinations of parameters should be low while the loss of the information connected with the reduction should be minimal. For this purpose, we use specialized algorithms called “Design of Experiment” (DoE) methods [1] to

do the reduction. Other user requirements such as ability to check partial results, modify the experiment during the runtime etc. are addressed by a specialized user interface (see chapter 6).

4. Data farming overview

The data farming process performed by the EUSAS system is shown in Fig. 1. It addresses all user requirements described above. In the beginning of the experiment the initial parameter space has to be defined through selection of values of all parameters. Afterwards, the parameter space is reduced. Reduction size depends on a chosen design of experiment method—it is possible to obtain several orders of magnitude reduction. During the experiment execution, there is an ability to see the partial results and to do some basic analysis of these results, for instance to generate the regression tree. On the basis of this analysis, extension of experiment may be done. The user interface allows to do it interactively. For example, if the analysis shows that one parameter influences the output more than others, the system may sample it more densely. The speed up of the experiment may be done by increasing the computing power (i.e. adding additional nodes) taking cost limit into account. Hence the speed of the experiment should depend on circumstances.

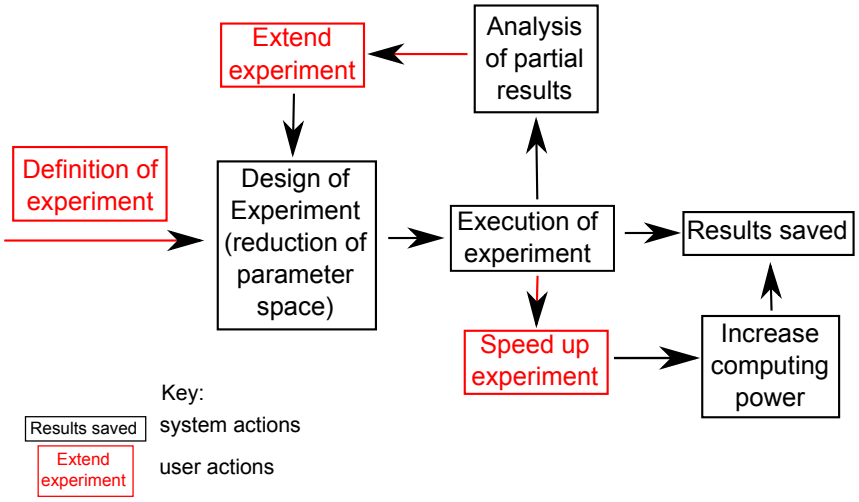

Figure 1. The data farming process performed by the EUSAS system.

5. Data farming architecture

The data farming architecture was designed to solve problems introduced in Section 3. Together with the user interface (see the next chapter) it creates the system that fulfils user requirements. Currently, DF experiments are performed on a computing

infrastructure, which is both dynamic and heterogeneous, the number of computing nodes can change depending on priorities of different tasks in a data centre and machines assigned to a particular data farming experiment can represent different hardware architecture and different operating systems, depending on current availability of resources.

Although a private data centre has several options regarding delegating parts of available resources to perform a concrete goal, for instance the physical worker nodes can be statically assigned and set up to compute DF experiments. However, this scenario requires substantial administrative effort related to installation and configuration of the operating system and software components. Another option is to use the virtualisation technology and start a number of virtual machines on the existing worker nodes, which can be stopped when not needed. Using virtualisation often increases utilization of the hardware infrastructure but also is much more flexible, e.g., allows running several different operating systems on the same worker node. Unfortunately, in many scenarios the private infrastructure is insufficient due to experiment or time limitations. To address this challenge, we intend to integrate our solution with publicly available commercial Clouds such as Amazon EC2, which in theory can be treated as unlimited source of computing power, depending only on the budget limit for a particular DF experiment. Depending on Cloud type, the computing power can be provided in various forms. In our case, the Infrastructure-as-a-Service (IaaS) model is most suitable, as it allows users to run virtual machines on third-party infrastructure. In most cases, IaaS Clouds provide an easy to use Application Programming Interface (API) to manage virtual machines. By using public Clouds, the user can speed up DF experiment when necessary.

A conceptual overview of the DF infrastructure is depicted in Fig. 2. A central point that dispatches experiment instances and provides user interface is called "simulation manager". We decided to use the "pull" mode to distribute the work among available resources. Using the "pull" mode means, the computing elements, e.g., virtual machines, ask for tasks to perform on their own. Thus, there is no need for simulation manager to know about available computing resources. Instead, the computing resources only have to know where simulation manager is located. Thus, it is trivial to scale the infrastructure only by running dedicated software on any available computing resources.

Regarding infrastructure management, we intend to provide a plug-in-based approach for extending simulation manager management capabilities. Using this approach, adding support for new type of computing resource will require only implementation a well-defined API for creating, running, stopping and destroying virtual machines for simulation manager and to provide a dedicated view for end users.

Simulations are implemented in Java language (using MASON framework) and log all actions within the simulation to a text file. The simulation instances are executed in virtual machines. The number of instances that are running simultaneously at each virtual machine is equal to the number of cores assigned to it. Each simulation in MASON uses one thread for its main activities. Although, it is possible to

perform some activities of agents by other threads simultaneously, the speed up is smaller than the number of additional cores available for the simulation. The user is able to start new virtual machines during data farming experiment. New virtual machines immediately ask for simulation instances what speeds up the experiment.

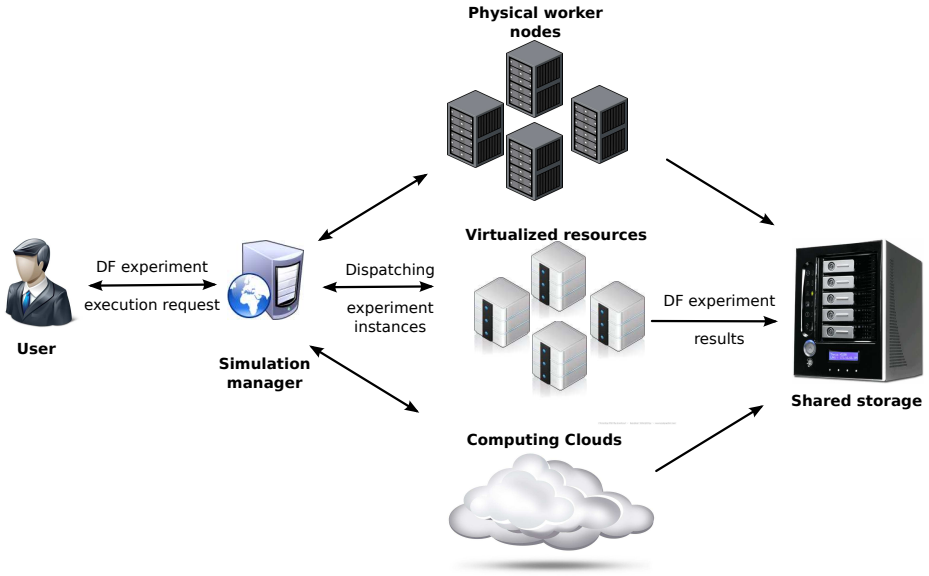

Figure 2. Data farming infrastructure overview.

The data farming manager schedules instances of experiment designed with experiment manager. It also cooperates with experiment manager during the definition of experiment providing Design of Experiment methods:

- Full factorial – all possible combinations of values of parameters are used,
- Fractional factorial – only a carefully chosen subset (fraction) of the experimental runs of a full factorial design is executed. Fedorov algorithm [15] is used to estimate how many combinations should be chosen,
- Orthogonal Latin Hypercubes – if all parameters have the same number of possible values, the number of executed instances is equal to the number of possible values of one parameter. In this case, it is guaranteed that each possible value of each parameter will be used exactly once. If some parameters have less possible values than others, some values of these parameters may be used more than once,
- 2^K – all possible combinations of minimal and maximal values of parameters are used.

The simulation manager also gathers data which are a base for analysis of partial results.

6. User interface

To manage data farming experiments in a user-friendly way, the EUSAS Data Farming module provides experimenters and administrators with a dedicated Graphical User Interface (GUI). It is web-based so it is accessible with various types of devices, which allows users to design and schedule DF experiments and to manage the Data Farming infrastructure. It contains two main panels, one dedicated to experiment configuration and monitoring and the second providing infrastructure management functionality.

6.1. Experiment management interface

"Experiment manager" supports all phases of running DF experiment. In particular, it allows researchers, e.g., users who run a DF experiment, to perform the following actions:

- schedule new DF experiments with custom parameterisation (several Design of Experiment methods are also supported),
- monitor DF experiments already running,
- analyse results of DF experiments either when an experiment is finished or during the execution of the experiment,
- download results of DF experiment.

The main view of the "Experiment manager" panel provides information about DF experiments, which are currently running, can be run or were ran in the past. The user can either go directly to a monitoring view of a currently running or historical experiment or decide to schedule a new experiment based on one of the available simulations. Starting a new DF experiment involves selecting parametrisation type of each input variable of an experiment, e.g., range of values, random value with different distributions or a concrete value, and then providing specific parameter values for each input variable regarding selected parametrisation types. Next, for parameters with range type of parametrisation, users can apply several Design of Experiment methods. Input parameters can be grouped into a number of sets, each can have different DoE method assigned. It gives the user ability to choose DoE which sample parameters more densely for more important parameters and DoE which reduce number of combinations better for other parameters.

After starting a DF new experiment, the user is redirected to a dedicated monitoring view depicted in Fig. 3. Using this view, the user can monitor the progress of the started DF experiment, analyse partial results, extend the experiment and download results after the experiment finishes.

In terms of analysing partial results of a DF experiment, we provide two types of charts, which intend to provide information about values of a selected Measure of Effectiveness (MoE). The first type of charts is histogram of MoE values, while the second type of charts is regression tree for a selected MoE.

Besides analysing partial results, the user can extend the set of input parameter values of a DF experiment. This functionality is embedded in the regression

tree chart. As in each node of a regression tree (except of leaves) a name of input parameter is given, the user can select an interesting input parameter and decide to generate additional experiment instances for new values of the selected input parameter. This functionality makes DF experiments interactive, i.e., the user can modify the experiment during the runtime without starting a new DF experiment.

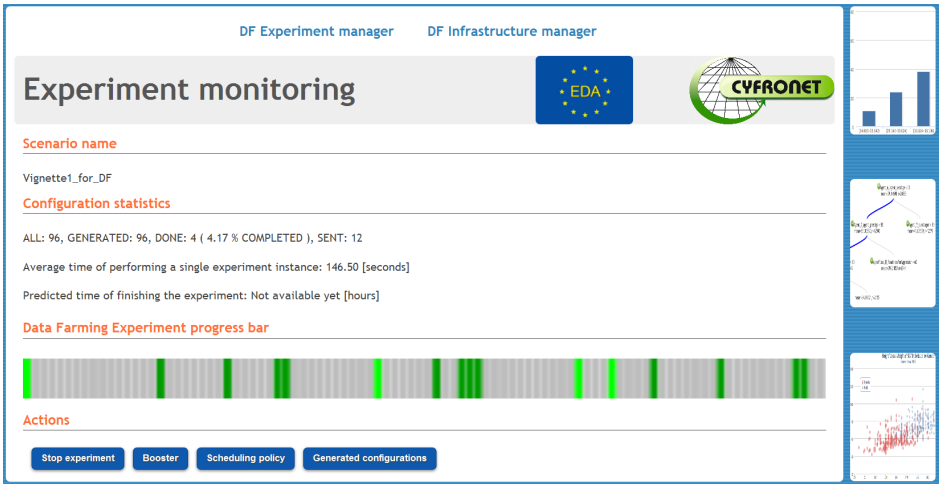

Figure 3. The monitoring view of a Data Farming experiment.

After the DF experiment is finished, the user can download a compressed package with results of the experiment. The package contains log files from each experiment instances and a CSV file with MoE and input parameter values for each experiment instance. The CSV file can be used, e.g., to analyze results with external, third party tools.

6.2. Infrastructure management interface

Besides management of DF experiments, "Simulation manager" allows users to manage the infrastructure, which is used to perform the DF experiments. "Infrastructure manager" is a panel (depicted in Fig. 4), which intends to facilitate the administration of the DF infrastructure. In the current version, it supports virtualised private infrastructure.

By using "Infrastructure manager", the user can register physical devices, which will be used to run virtual machines. Then, the user can create new virtual machines with a specified set of resources, e.g., number of CPU cores and amount of RAM. Each created virtual machine can be run, paused, stopped and deleted.

Figure 4. Data farming infrastructure management view.

7. Conclusions

In this paper we have presented a novel generic purpose solution for performing large scale data farming experiments on heterogeneous computing infrastructure consisting of privately owned and Cloud based resources. The system is currently being used for running data farming experiments related to military scenarios within the framework of European Defence Agency EUSAS project, where the simulation runs provide data about possible behaviour outcomes of large groups of civilians and blue forces in both military and urban crowd control scenarios. The data produced by the data farming is used to improve the psychological models of the civilian agents in the simulation as well as evaluate different rules of engagement for different situations.

The main future work for this system is improvement of the user interface in order to reflect the requirements of actual military analysts and integration of the system with existing Cloud infrastructures in order to extend the scalability of the system.

Acknowledgements

This work is supported by the EDA project A-0938-RT-GC EUSAS.

References

[1] Atkinson A. C., Donev A. N.: *Optimum Experimental Designs.* Clarendon Press, 1992.

[2] Bezek A., M. G., Bratko I.: Multi-agent strategic modeling in a robotic soccer domain. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 457–464, 2006.

[3] Brandstein A., Horne G.: Data farming: A meta-technique for research in the 21st century. In *Maneuver Warfare Science 1998*. Marine Corps Combat Development Command Publication, 1998.

[4] Choo C. S., Ng E. C., Ang C. K., Chua C. L.: Systematic data farming – an application to a military scenario. In *Proceedings of Army Science Conference*, 2006.

[5] Forsyth A., Horne G., Upton S.: Marine corps applicatons of data farming. In Kuhl M. E., Steiger N. M., Armstrong F. B., Joines J. A., editors, *Proceedings of the 2005 Winter Simulation Conference*, pp. 1077–1081, 2005.

[6] Funika W., Korcyl K., Pieczykolan J., Skital L., Balos K., Slota R., Guzy K., Dutka L., Kitowski J., Zielinski K.: Adapting a hep application for running on the grid. *Computing and Informatics*, 28:353–367, 2009.

[7] Hluchy L., Kvassay M., Dlugolinsky S., Schneider B., Bracker H., Kryza B., Kitowski J.: Handling internal complexity in highly realistic agent-based models of human behaviour. In *Proceedings of 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI 2011)*, pp. 11–16, 2011.

[8] Horne G., Meyer T.: Data farming: Discovering surprise. In Ingalls R. G., Rossetti M. D., Smith J. S., Peters B. A., editors, *Proceedings of the 2004 Winter Simulation Conference*, pp. 1082–1087, 2004.

[9] Horne G., Meyer T.: Data farming and defense applications. In *Proceedings of the MODSIM World 2010 Conference*, pp. 74–82, 2010.

[10] Korcyl K., Szymocha T., Funika W., Kitowski J., Slota R., Balos K., Dutka L., Guzy K., Kryza T., Pieczykolan J.: The atlas experiment on-line monitoring and filtering as an example of real-time application. *Computer Science, annual of AGH University of Science and Technology*, 9:77–86, 2008.

[11] Kvassay M., Hluchy L., Kryza B., Kitowski J., Seleng M., Dlugolinsky S., Laclavik M.: Combining object-oriented and ontology-based approaches in human behaviour modelling. In *Proceedings of IEEE 9th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp. 177–182, 2011.

[12] Laclavik M., Dlugolinsky S., Seleng M., Kvassay M., Schneider B., Bracker H., Wrzeszcz M., Kitowski J., Hluchy L.: Agent-based simulation platform evaluation in the context of human behavior modeling. In *Proceedings of the second international workshop on Infrastructures and Tools for Multiagent Systems*, pp. 396–410, 2011.

[13] Luke S., Cioffi-Revilla C., Panait L., Sullivan K., Balan G.: Mason: A multi-agent simulation environment. *Simulation: Transactions of the society for Modeling and Simulation International*, 82(7):517–527, 2005.

[14] Marco J., Campos I., Coterillo I., Diaz I., Lopez A., Marco R., Martinez-Rivero C., Orvis P., Rodriguez D., Gomes J., Borges G., Montecelo M.,

David M., Silva B., Dias N., Martins J. P., Fernandez C., Garcia-Tarres L., Veiga C., Cordero D., Lopez Cacheiro J., Lopez I., Garcia-Tobio J., Costas N., Mourino J. C., Gomez A., Bogacki W., Meyer N., Owsiak M., Plociennik M., Pospieszny M., Zawadzki M., Hammad A., Hardt M., Fernandez E., Heymann E., Senar M. A., Padee A., Nawrocki K., Wislicki W., Heinzreiter P., Baumgartner M., Rosmanith H., Kranzmuller D., Volkert J., Kenny S., Coghlan B., Pajak R., Mosurska Z., Szymocha T., Lason P., Skital L., Funika W., Korcyl K., Pieczykolan J., Balos K., Slota R., Guzy K., Dutka L., Kitowski J., Zielinski K., Hluchy L., Dobrucky M., Simo B., Habala O., Astalos J., Ciglan M., Sipkova V., Babik M., Gatial E., Valles R., Reynolds J. M., Serrano F., Tarancon A., Velasco J. L., Cstejon F., Dichev K., Keller R., Stork S.: The interactive european grid: Project objecives and achievements. *Computing and Informatics*, 27(2):161–171, 2008.

[15] Nguyen N., Miller A. J.: A review of some exchange algorithms for constructing discrete d-optimal designs. *Computational Statistics & Data Analysis*, 14(4):489–498, 1992.

Affiliations

Bartosz Kryza
AGH University of Science and Technology, ACC Cyfronet AGH, Krakow, Poland,
bkryzaagh.edu.pl

Dariusz Król
AGH University of Science and Technology, ACC Cyfronet AGH, Krakow, Poland,
dkrol@agh.edu.pl

Michal Wrzeszcz
AGH University of Science and Technology, ACC Cyfronet AGH, Krakow, Poland,
wrzeszcz@agh.edu.pl

Lukasz Dutka
AGH University of Science and Technology, ACC Cyfronet AGH, Krakow, Poland

Jacek Kitowski
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, ACC Cyfronet, AGH, Krakow, Poland

Received: 25.01.2012
Revised: 26.03.2012
Accepted: 9.07.2012