Security-Aware Distributed Job Scheduling in Cloud Computing Systems: A Game-Theoretic Cellular Automata-based Approach

Jakub Gąsior and Franciszek Seredyński

Department of Mathematics and Natural Sciences, Cardinal Stefan Wyszyński University, Warsaw, Poland
j.gasior@uksw.edu.pl f.seredynski@uksw.edu.pl

Abstract. We consider the problem of security-aware scheduling and load balancing in Cloud Computing (CC) systems. This optimization problem we replace by a game-theoretic approach where players tend to achieve a solution by reaching a Nash equilibrium. We propose a fully distributed algorithm based on applying Iterated Spatial Prisoner's Dilemma (ISPD) game and a phenomenon of collective behavior of players participating in the game. Brokers representing users participate in the game to fulfill their own three criteria: the execution time of the submitted tasks, their execution cost and the level of provided Quality of Service (QoS). We experimentally show that in the process of the game a solution is found which provides an optimal resource utilization while users meet their applications' performance and security requirements with minimum expenditure and overhead.

Keywords: Collective behavior; Multi-agent systems; Spatial prisoner's dilemma game; Second order cellular automata.

1 Introduction

An increasing popularity of the CC paradigm [2, 5] is lately accompanied by another phenomenon, namely fast developing of the Internet of Things (IoT). A large number of IoT devices will require automatic access to the resources of CC and, together with human clients, will create a huge number of potential CC users. The users sending their computational requests to CC will require fulfilling at least two criteria: predefined efficiency and predefined level of security. These criteria potentially can be fulfilled by solving in a centralized way a huge multi-criteria optimization problem with a request of full information about the system resources and users demands, what seems to be intractable for realistic problems.

Within the proposed simulated framework, we analyze a novel approach to considered multi-objective job scheduling problem aiming to optimize both aforementioned performance criteria. Firstly, we apply algorithmic game-theory to extend the notions of independence and selfishness of each user submitting his jobs to the cloud. We give to the users the freedom to choose the best strategy for their jobs. The proposed users' strategies and resulting job allocations are used as an input in our distributed scheduling scheme employing a non-cooperative game-theoretic approach. The whole process is realized in the two-dimensional Cellular Automata (CA) space where individual agents are mapped onto a regular square lattice. Main issues that are addressed here are: a) incorporating the global goal of the system into the local interests of all agents participating in the scheduling game, and b) formulation of the local interaction rules allowing to achieve those interests.

Moreover, we propose to develop a fully distributed approach based on converting an optimization problem into a game-theoretic problem where brokers representing users demands will search a solution as a Nash equilibrium. For this purpose we will use a variant of ISPD game proposed by [6] in the context of CA space. The game has many interesting features. It has built-in "soft security" mechanisms called membranes and it has potential of providing a collective behavior. Similar to [8, 14], we will combine this approach with a traditional Multiobjective Genetic Algorithm (MOGA) approach in order to account for different trade-offs between QoS, computation time, and its associated costs.

The remainder of this paper is organized as follows. In Section 2, we describe the proposed CC system model and define the scheduling problem. Section 3 contains some background concerning CA, details of the studied ISPD game and some results of the experimental study of the game from positions of collective behavior. Section 4 presents the proposed agent-based game-theoretic scheduling scheme and its application. Section 5 demonstrates the performance metrics, the input parameters and experimental results. Finally, Section 6 concludes the paper.

2 Multi-objective Scheduling in Cloud Environment

2.1 Cloud Datacenter Model

The system model is an extension of the architecture introduced in [13] and further developed in our earlier paper [4]. A system consists of a set of geographically distributed Cloud nodes $M_1, M_2, ..., M_m$, which are specified by several characteristics, including their processing capacity s_i in million floating point operations per second (MFLOPS), cost per hour c_i, memory and storage space. For the purpose of this paper, we follow the specification of Compute Optimized Virtual Machine (VM) series provided by Amazon EC2.

Users $(U_1, U_2, ..., U_n)$ submit to the system workflow application J_k^j for execution. Each application is the set of n tasks or jobs. Users are expected to pay appropriate fees to the Cloud provider dependent on the Service Level Agreement (SLA) requested. Job (denoted as J_k^j) is jth job produced (and owned) by user U_k. J_k stands for the set of all jobs produced by user U_k, while $n_k = |J_k|$ is the number of such jobs. Each task has varied parameters defined as a tuple $< r_k^j, size_k^j, t_k^j, d_k^j >$, specifying its release dates $r_k^j \geq 0$; its size $1 \leq size_k^j \leq m_m$, that is referred to as its processor requirements or *degree of parallelism*; its workload t_k^j defined in MFLOPs and a deadline d_k^j.

Rigid jobs require a fixed number of processors for parallel execution: this number is not changed until the job is completed. We assume that job J_k^j can only run on machine M_i if $size_k^j \leq m_i$ holds, that is, we do not allow multi-site execution and co-allocation of processors from different machines. We assume a space sharing scheduling approach, therefore a parallel job J_k^j is executed on exactly $size_k^j$ disjoint processors without preemptions.

In mapping jobs onto cloud resources, we have to tackle a number of security-related problems [7]. The first step is for a user to issue a Security Demand (SD) to all submitted jobs. When setting up the SD values, users should be concerned about issues related to job sensitivity, job execution environment, access control and data integration [12], etc. On the other hand, the Security Level (SL) of a machine can be attributed to the available intrusion detection mechanisms, firewalls, and anti-virus capabilities, as well as prior job execution success rates. This defense capability is evaluating the risk existing in the allocation of a submitted job to a specific machine.

Thus, a job is expected to be successfully carried out when SD and SL satisfy a security-assurance condition ($SD \leq SL$) during the job mapping process. The SD is a real fraction in the range [0,1] with 0 representing the lowest and 1 the highest security requirement. The SL is in the same range with 0 for the most risky resource site and 1 for a risk-free or fully trusted site. Specifically, we define a *Job Failure Model* as a function of the difference between the job security demand and a resource trust (Equation 1):

$$P_{i,j}^{Failure} = \begin{cases} 0, & SD_j \leq SL_i, \\ 1 - exp^{-(SD_j - SL_i)}, & SD_j > SL_i. \end{cases} \tag{1}$$

Meeting the security assurance condition ($SD_j \leq SL_i$) for a given job-machine pair guarantees successful execution of that particular job. Such scheduling will be further called as a *Secure Job Allocation*. On the other hand, successful execution of the job assigned to machine without meeting this condition ($SD_j > SL_i$), will be dependent on the calculated probability and further referred to as a *Risky Job Allocation*.

Our goal is to design a dynamic and fully decentralized scheduling architecture oriented to on-line CC platforms. In this paper, we make the following assumptions about the characteristics of the modeled CC system and workload scheduler:

- The model is *on-line*;
- The scheduler is *decentralized* and *clairvoyant*;
- Each resource has a different number of processing elements;
- Jobs are *parallel* and *rigid*. Job J_k^j must be executed in parallel on $size_k^j$ processors of exactly one resource.
- Jobs are characterized by their release dates ($r_k^j \geq 0$);
- *Preemption* is not allowed;
- Processors are *time-shared* between jobs. Resources are *space-shared* and *time-shared* between jobs.

Our main scheduling objective is to ensure fairness among multiple users requesting access to cloud resources by distributing user's U_k jobs among the available machines. A detailed explanation of this process is provided in Section 4.

3 Collective Behavior in Iterated Spatial Prisoner's Dilemma Game

3.1 Cellular Automata

CA are spatially and temporally discrete computational systems originally proposed by S. Ulam and J. von Neumann in the late 1940s. Today they are a powerful tool used in computer science, mathematics and natural science to model different phenomena and develop parallel and distributed algorithms [15].

We will consider two dimensional (2D) CA which is 2D lattice consisting of $n \times n$ elementary cells. For each cell a local 2D neighborhood of a radius r is defined (see, Fig. 1). At a given discrete moment of time t each cell (i, j) is in some state $q^t_{i,j}$, in the simplest case it is a binary number 0 or 1.

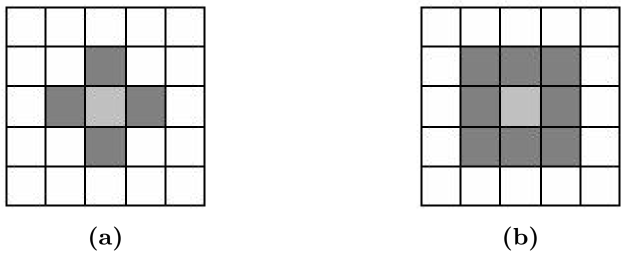


Fig. 1: 2D CA of the size 5×5. Examples of a local neighborhood of the size $r = 1$ of a central cell (in light gray): a) von Neumann neighborhood consisting of 4 neighbors, b) Moore neighborhood consisting of 8 neighbors.

At discrete moments of time, all cells synchronously update their states according to the assigned local rules (transition functions), which depend on states of their neighborhood. A cyclic boundary condition is applied to finite size of 2D CA. If two or more different rules are assigned to CA to update cells, such CA is called non-uniform.

3.2 Iterated Spatial Prisoner's Dilemma Game

We consider 2D CA of the size $n \times n$. Each cell of CA has a Moore neighborhood of radius r, and also a rule assigned to it which depends on the state of its neighborhood.

Each cell of 2D CA will be considered as an agent (player) participating in the ISPD game [6]. Each player (a cell of CA) has two possible actions: C (cooperate) and D (defect). It means that at a given moment of time each cell is either in the state C or the state D. Payoff function of the game is given in Tab. 1.

Table 1: Payoff function of a row player participating in ISPD game.

Action	Cooperate (C)	Defect (D)
Cooperate (C)	$R = 1$	$S = 0$
Defect (D)	$T = b$	$P = 0$

Each player playing a game with an opponent player in a single round (iteration) receives a payoff equal to R, T, S or P, where $T > R > P > S$. Values of these payoffs used in this study are specified in Tab. 1, where $P = S = 0$, $R = 1$, and $T = b\,(1.1 < b < 1.8)$. If a player takes action C and the opponent player also takes action C than the player receives payoff equal to $R = 1$. However, if a player takes action D and the opponent player still takes action C, the defecting player receives payoff equal to $T = b$. In two remaining cases, a player receives a payoff equal to 0.

In fact, each player associated with a given cell plays in a single round a game with each of his 8 neighbors and this way collects some total score. After a q number of rounds (iterations of CA) each cell (agent) of CA can change its rule (strategy). We assume that considered 2D CA is non-uniform CA, and to each cell one of the following rules can be assigned: *all-C* (always cooperate), *all-D* (always defect), and *k-D* (cooperate until not more than $k\,(0 \leq k \leq 7)$ neighbors defect). The strategy *k-D* is a generalized strategy TFT, and when $k = 0$ it is exactly the strategy TFT.

A player changes its current strategy into another one comparing collected during q rounds total score with scores collected by his neighbors. He selects as his new strategy the strategy of the best performing neighbor, i.e. the player whose the collected total score is the highest. This new strategy is used by a cell (player) to change its current state, and the value of the state is used in games during the next q rounds.

It is worth to notice that the considered 2D CA differs from a classical CA, where rules assigned to cells do not change during evolving CA in time. A CA with a possibility of changing rules is called a second-order CA. In opposite to a classical CA, a second-order CA has potential to solve various optimization problems.

The main research issue in [6] was to study conditions of appearing of specific structures called membranes created by cells using *k-D* rules which protected cooperated players from defected players. In this study, we are interested in the collective behavior of a large team of players, in particular in conditions of emerging global cooperation in such teams.

4 Game-theoretic Approach to Cloud Scheduling

This section provides a complete overview of our proposed agent-based game-theoretic distributed scheduling scheme. To develop a truly distributed multi-objective scheduling framework we propose a four-stage procedure consisting of:

– *Stage 1*: For each user U_k submitting his batch of jobs J_k to the cloud we assign a broker B_k, responsible for allocation of user's U_k jobs in the system;
– *Stage 2*: A Pareto frontier is calculated for each broker B_k and batch of jobs J_k submitted by user U_k under an assumption that all cloud resources belong exclusively to the broker B_k using the MOGA approach;
– *Stage 3*: Specific job allocation solutions (*Scheduling Strategies*) from the Pareto frontier are selected according to the Pareto *Selection Policies* characterizing user's preferences (namely, *Maximum Reliability*, *Minimum Cost* and *Minimum Cost with Deadline*). These solutions will be subsequently used by a broker B_k representing user's U_k interests;
– *Stage 4*: Individual brokers employ previously selected *Scheduling Strategies* in the ISPD game realized in a two-dimensional CA space, trying to find a compromise global scheduling solution (Nash Equilibrium (NE) point) in their selfish attempts to obtain cloud resources.

An equilibrium is reached when none of the participating brokers is interested in changing its own task allocation strategy. This is experienced when no improvement is achieved in the individual scores and a valid global problem solution $\overline{S}$ is obtained. A more detailed explanation of this process was provided in our earlier work [3].

In this paper, we are more interested in the problem of incorporating this global goal of the system into the local interests of individual brokers. To study a possibility of emergence of a global collective behavior of players in the sense of the second class of the collective behavior classification [1, 10] we will introduce a number of local mechanisms of interaction between brokers, which can be potentially spread or dismissed by the brokers during the evolution of system in the process of the iterated game and study their impact on the scheduling performance.

The first mechanism is the possibility of sharing locally profits obtained by players participating in the game. Some kind of hard local sharing was successfully used [11] in the context of LA games. Here we will be using a soft version of sharing, where a player decides to use it or not. It is assumed that each player has a tag indicating whether he wishes (on) or not (off) to share his payoff with players from the neighborhood who also wish to share their payoffs. The sharing works in such a way that if two players play a game and both wish to share their payoffs, each of them receives half of the payoff from the sum of payoffs received by both players in the game. Before starting the iterated game each player turns on its tag with a predefined probability $p_{sharing}$.

The second mechanism which will be used is a mutation of a number of the system parameters. With some predefined value of probability, a CA-based agent of the first type can change the currently assigned strategy (rule) to one of the two other strategies. Similarly, a CA-based agent of the second type can increase/decrease its probability of cooperation.

The third mechanism which can be used is a competition which is a generalization of the idea proposed in [9]. In fact, each player associated with a given cell plays in a single round a game with each of his neighbors and this way collects some total score. If the competition mechanism is in turned on, after a q number of rounds (iterations) each agent compares its total payoff with total payoffs of its neighbors. If a more successful player exists in the neighborhood of a given player this player is replaced by the most successful one.

5 Experimental Analysis and Performance Evaluation

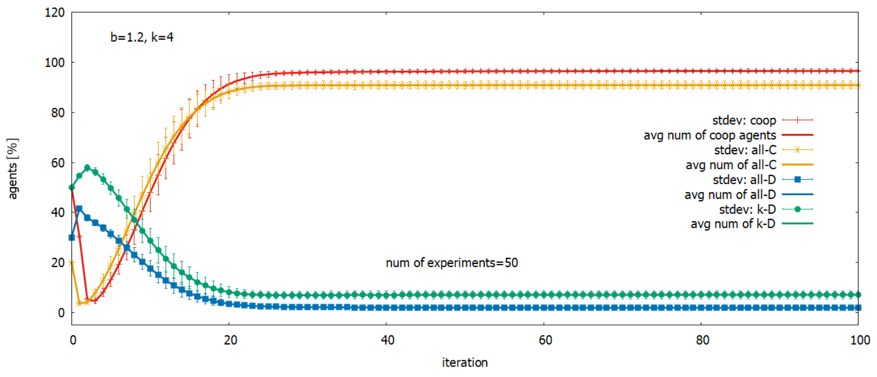
5.1 Parameter settings

A number of experiments with the proposed system have been conducted with the following settings of parameters. A 2D array of the size of 50×50 cells (players) was used, with an initial state C or D (player action) set with the probability equal to 0.5. Initially, the rule *k-D* was assigned (if applied) to CA cells with the probability equal to 0.7, and the remaining three rules with the probability equal to 0.1. When the competition mechanism was turned on, updating the array cells (by a winner in a local neighborhood) was conducted after $q = 1$ iterations.

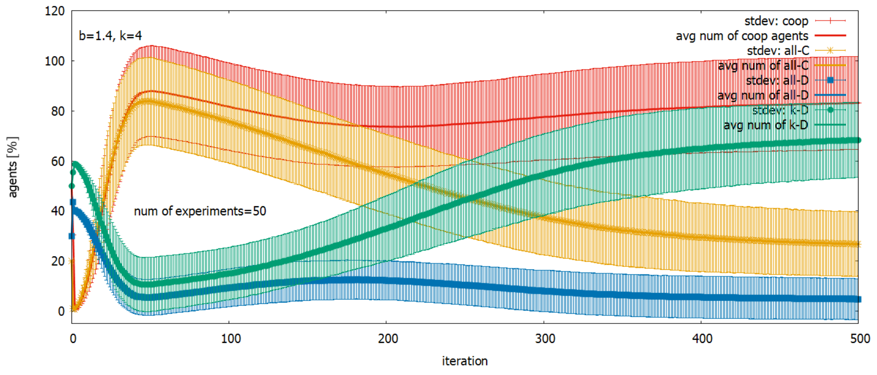
5.2 Experiment #1. Typical runs of ISPD game with variable values of parameter b.

The purpose of this set of experiments was to observe the dynamics and conditions of appearing in the game a collective behavior of players measured by a total number of cooperating players in the game. Initially conducted experiments have shown that the ability of emergence of collective behavior depends mainly on values of parameter b and only a little bit on a value of k. Therefore, we will show some experimental results for the same value $k = 4$ and different values of b.

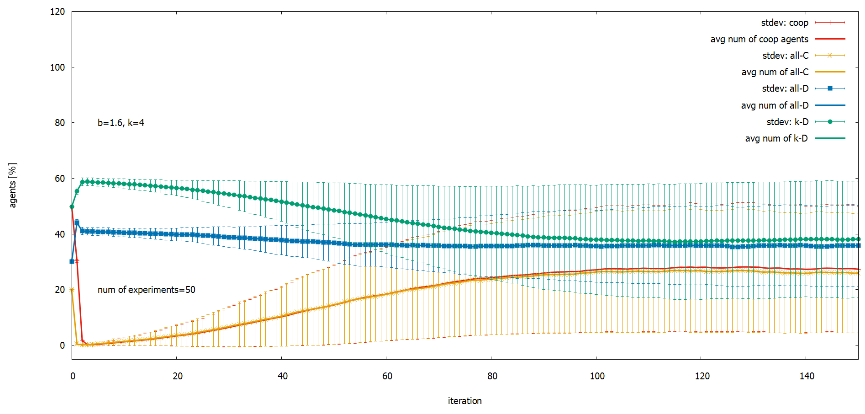
The experiments were conducted for 2D CA of the size $n \times n = 10000$ and Fig. 2 shows results averaged over 50 runs of the game. Depending on the value of b we can notice four regions of behavior resulting in different types of equilibria. When b is in the range $[1.1, \ldots, 1.3]$ about 96% of players (see, Fig. 2a) decides quickly (after about 25 iterations) to cooperate (red color). Among cooperative players around 91% of players uses strategy *all-C*, (yellow color) and around 5% uses the strategy *k-D*, (green color). Remaining about 4% of players use strategy *all-D*.



(a)



(b)



(c)

Fig. 2: Averaged runs of ISPD game with $k = 4$ for: a) $b = 1.2$, b) $b = 1.4$ and c) $b = 1.6$.

When parameter b is the range $[1.4, \ldots, 1.5]$ (see, Fig. 2b) the behavior of players is more complex, and the system achieves an equilibrium after about 450 iterations. During about first 50 iterations global cooperation of the level of around 90% is reached (red color), and next it is slowly decreasing during around 200 iterations to return back to the previous level after around 450 iterations. We can observe the changing strategies structure of cooperating players.

While at the beginning of the game about 80% of coopering players use the strategy *all-C* (yellow) and about 10% of cooperating players use the strategy *k-D*, after around 450 iterations we can see a new stable proportion of cooperating players: around 65% of them use strategy *k-D* and around 25% use the strategy *all-C*. Still, a few percents of players use the strategy *all-D*. Comparing to the previous experiment we can see the increasing standard deviation of cooperation level and strategies applied by players.

When the value of b is close to the value 1.6 (see, Fig. 2c) a global cooperation grows very slowly and reaches the level of around 20% (red color) after about 100 iterations, and it is supported by about 18% of players applying the strategy *all-C* (yellow color). The remaining players use either *k-D* strategy (about 41%, green color) or *all-D* (about 39%, blue color). About 2% of players applying *k-D* strategy support a global operation.

When $b \geq 1.7$ (not shown) global cooperation is not observed.

5.3 Experiment #2: Patterns of agents' strategies.

For the presented above dependencies between the level of global cooperation and the parameter b, we can also observe accompanying spatial patterns of strategies used by players. Fig. 3 (left) shows corresponding spatial pattern of players' strategies used in the game with the parameter $b = 1.1$. One can see that dominating strategy used by players is strategy *all-C* (white color) and some players use strategies *all-D* (black color) and *k-D* (tones of orange color which depends on the value of k). Strategies of players which use either *all-D* or *k-D* strategies create short, isolated, sometimes vertical or horizontal regular structures.

For games with values $b = 1.2$ (not shown) and $b = 1.3$ (see, Fig. 3 (right)) spatial structures of strategies are similar to the game with $b = 1.1$, but we can observe an increasing length of regular vertical and horizontal structures, which are still isolated for $b = 1.2$, but more connected with $b = 1.3$. We can see also an increasing number of strategies *k-D* which replace the strategy *all-C*.

For $b = 1.4$ (see, Fig. 4 (left)) spatial structure of strategies are created by isolated small clusters of *all-C* strategies and rarely clusters of strategies *all-D*, both surrounded by strategies *k-D*. When $b = 1.5$ (see, Fig. 4 (right)) spatial structures of strategies are created by long irregular isolated chains with two types of granularity of *k-D* strategies and *all-D* strategies. When $b = 1.6$ (see, Fig. 5 (left)) the structure of spatial strategies is similar to that observed for $b = 1.4$.

However, in this case we can see appearing new small clusters of *all-D* strategies. Finally, when $b = 1.7$ (see, Fig. 5 (right)) we can notice that none of players uses *all-C* strategy but dominating strategy selected by them is the strategy *k-D* with different values of k and some clusters of players use the strategy *all-D*.

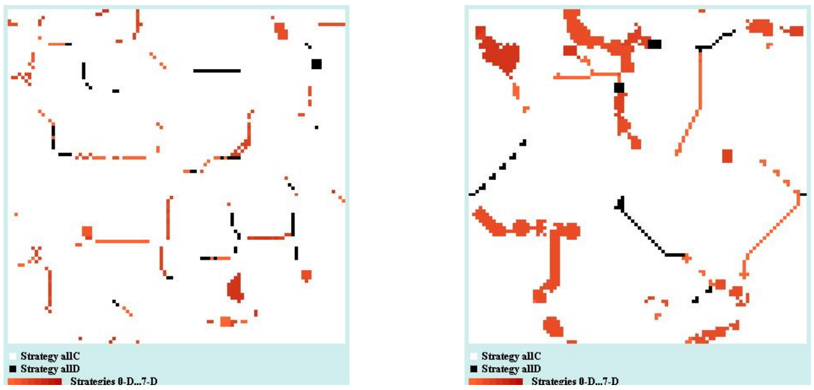


Fig. 3: Strategies distribution for: $b = 1.1$ (left) and $b = 1.3$ (right).

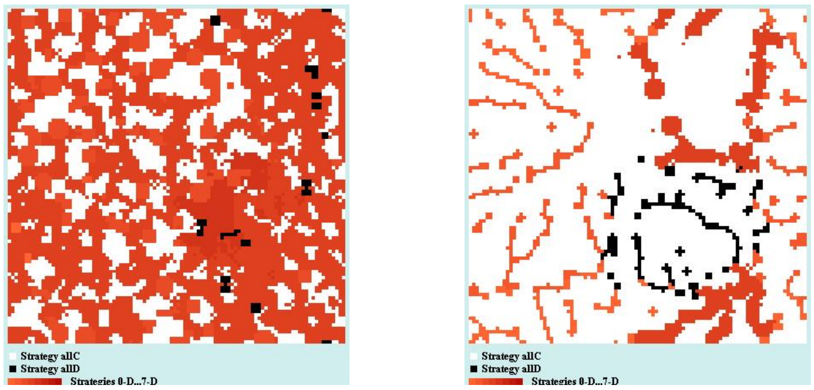


Fig. 4: Strategies distribution for: $b = 1.4$ (left) and $b = 1.5$ (right).

5.4 Experiment #3: A distributed scheduling game.

The last series of experiments incorporated our previous findings into a fully decentralized scheduler capable of determining a *Competitive Scheduling Profile* that minimizes job completion times and failure probabilities by exploiting agents' selfish needs to maximize their own payoff scores. For that purpose, 8

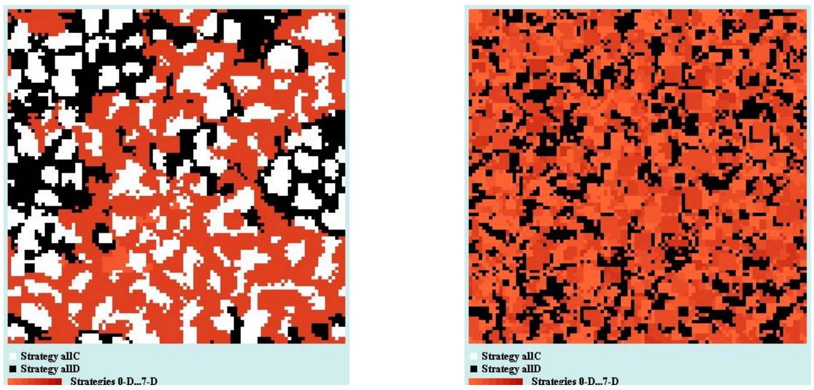


Fig. 5: Strategies distribution for: $b = 1.6$ (left) and $b = 1.7$ right).

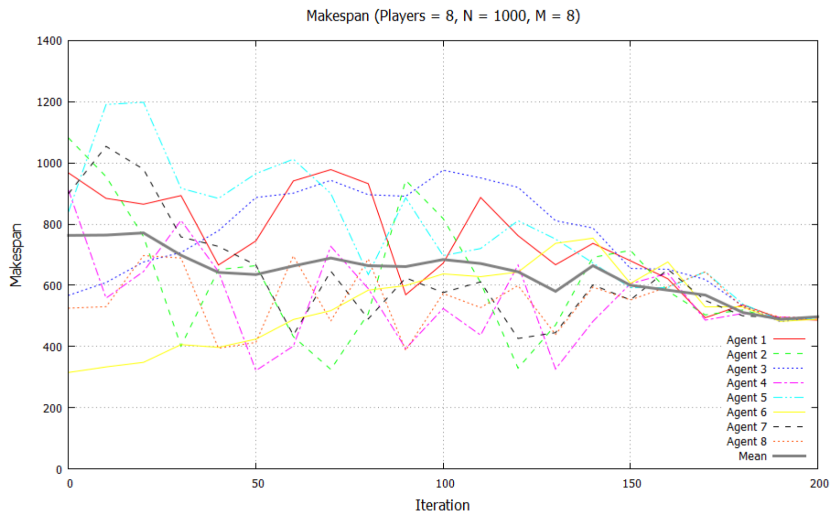
independent users submitted to the system a randomly generated *Bag of Tasks* containing $n_k = 1000$ job instances. Jobs were then scheduled within $m = 8$ CC nodes by independent brokering agents assigned to individual users.

Scheduling game was conducted on a rectangular CA lattice. Initial ISPD game *Actions* and *Spatial Strategies* as well as Pareto *Selection Polices* were equally and randomly distributed between the participating players. Parameters k and b were selected in order to maximize the emergence of global cooperation between participating players. The maximum number of game iterations has been fixed at a value of $T_{Max} = 200$ steps. 100 independent runs per configuration have been carried out to guarantee statistical significance of the results and construct an approximate Pareto frontier by gathering non-dominated solutions in all executions.

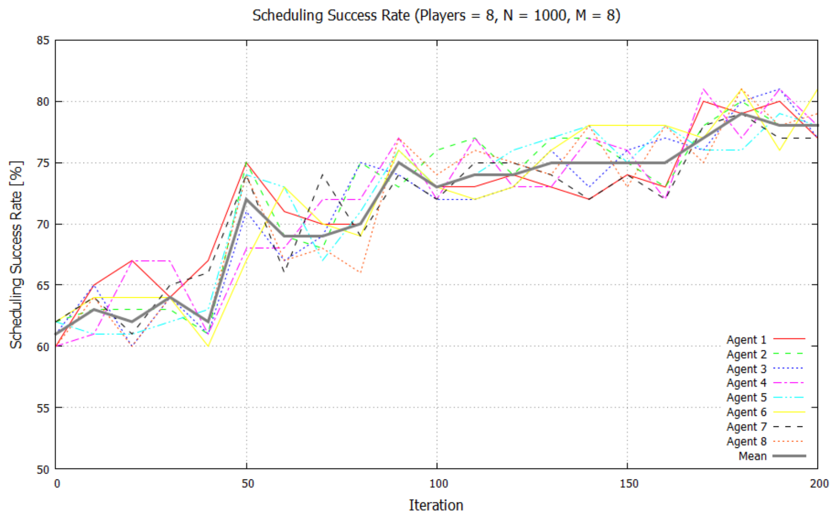
The efficiency of the analyzed job scheduling methods was measured in terms of:

– ***Makespan:*** the time of completion of the latest job submitted to the Cloud, defined as $C_{max} = max_k\{C^k_{max}\}$;
– ***Scheduling Success Rate:*** the percentage of jobs successfully completed in the Cloud;
– ***Total Cost:*** cost function dependent on both the requested level of SLA and a number of used CC nodes.

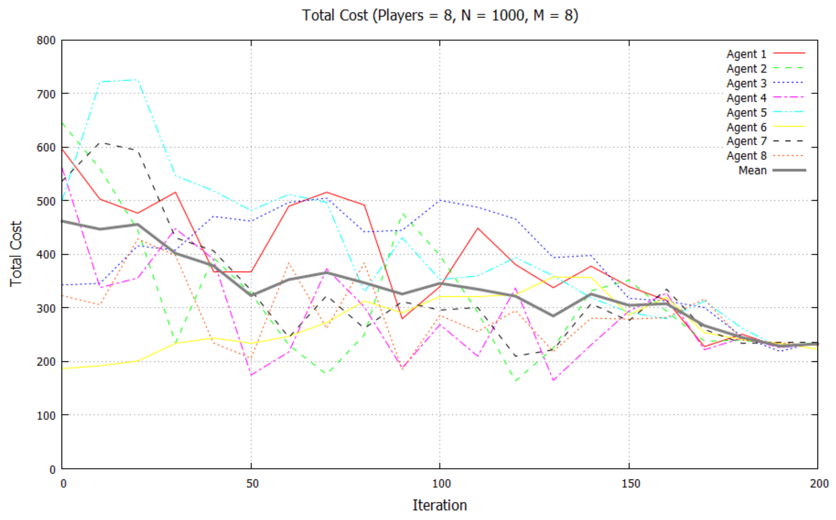
The simulation results are given in Fig. 6 showing the *Makespan* (Fig. 6(a)), *Scheduling Success Rate* (Fig. 6(b)) and *Total Cost* (Fig. 6(c)) during 200 iterations of ISPD game for each individual agent. These experiments aimed to analyze the number of iterations necessary to solve potential conflicts between the selfish strategies of the individual agents and converge to a global equilibrium using our proposed distributed scheduling scheme. It shows how global performance metrics improve over time due to actions taken by the brokers during our distributed resource allocation game.



(a)



(b)



(c)

Fig. 6: Scheduling performance over time for a total of $n = 8000$ jobs scheduled within $m = 8$ CC nodes by 8 independent agents. (a) Makespan. (b) Scheduling Success Rate. (c) Total Cost.

6 Conclusion and Future Work

We have presented a theoretical framework and an experimental software tool to study the behavior of heterogeneous multi-agent systems operating in an environment described in terms of a ISPD game. This framework was defined in order to solve global optimization tasks in a distributed way by the collective behavior of agents. The essential from this point of view was to use a concept of the second order CA and introduce some specific mechanisms of interaction between agents. A set of conducted experiments have shown that these proposed solutions are promising building blocks enabling emergence of global collective behavior in such heterogeneous multi-agent systems.

We have studied the conditions of emergence of global cooperation in large CA-based broker teams. We have shown that the phenomenon of global cooperation depends mainly on values of the parameter b of payoff function reflecting a gain of a player who defects while the other players still cooperate, and to some extent on the value k describing the tolerance of players for defection by their neighboring players.

We incorporated these findings into the paradigm of MOGA-based scheduling in order to use the competition among the entities involved in the scheduling process to converge towards Nash equilibrium. It allowed us to account for often contradicting interests of the clients within the CC system, without the need of any centralized control and introduced a number of desirable properties such as adaptation and self-organization. Conditions of emerging a global behavior of such systems depend on a number of parameters and these issues will be a subject of our future work.

Bibliography

[1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, Mar 2013.

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype and Reality for Delivering Computing as the 5th Utility. *Future Gener. Comput. Syst.*, 25(6):599–616, June 2009.

[3] J. Gąsior and F. Seredyński. Decentralized Job Scheduling In The Cloud Based On A Spatially Generalized Prisoner's Dilemma Game. *Applied Mathematics and Computer Science*, 25(4):737–751, 2015.

[4] J. Gąsior, F. Seredyński, and A. Tchernykh. A Security-Driven Approach to Online Job Scheduling in IaaS Cloud Computing Systems. In R. Wyrzykowski, J. Dongarra, E. Deelman, and K. Karczewski, editors,

Parallel Processing and Applied Mathematics, pages 156–165, Cham, 2018. Springer International Publishing.

[5] W. A. Jansen. Cloud Hooks: Security and Privacy Issues in Cloud Computing. In *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, HICSS '11, pages 1–10, Washington, DC, USA, 2011. IEEE Computer Society.

[6] Y. Katsumata and Y. Ishida. On a Membrane Formation in a Spatio-temporally Generalized Prisoner's Dilemma. In H. Umeo, S. Morishita, K. Nishinari, T. Komatsuzaki, and S. Bandini, editors, *Cellular Automata*, pages 60–66, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[7] J. Kolodziej, S. U. Khan, L. Wang, M. Kisiel-Dorohinicki, S. A. Madani, E. Niewiadomska-Szynkiewicz, A. Y. Zomaya, and C.-Z. Xu. Security, Energy, and Performance-aware Resource Allocation Mechanisms for Computational Grids. *Future Gener. Comput. Syst.*, 31:77–92, Feb. 2014.

[8] S. Nesmachnow, C. Perfumo, and . Goiri. Controlling datacenter power consumption while maintaining temperature and qos levels. In *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, pages 242–247, Oct 2014.

[9] M. A. Nowak and R. M. May. Evolutionary Games and Spatial Chaos. *Nature*, 359:826, 1992.

[10] F. Rossi, S. Bandyopadhyay, M. Wolf, and M. Pavone. Review of multi-agent algorithms for collective behavior: a structural taxonomy. *IFAC-PapersOnLine*, 51(12):112 – 117, 2018. IFAC Workshop on Networked & Autonomous Air & Space Systems NAASS 2018.

[11] F. Seredynski. Competitive coevolutionary multi-agent systems: The application to mapping and scheduling problems. *Journal of Parallel and Distributed Computing*, 47(1):39 – 57, 1997.

[12] S. Song, K. Hwang, and Y.-K. Kwok. Risk-Resilient Heuristics and Genetic Algorithms for Security-Assured Grid Job Scheduling. *IEEE Trans. Comput.*, 55(6):703–719, June 2006.

[13] A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J. E. Pecero, S. Nesmachnow, and A. Y. Drozdov. Online Bi-Objective Scheduling for IaaS Clouds Ensuring Quality of Service. *J. Grid Comput.*, 14(1):5–22, Mar. 2016.

[14] A. Tchernykh, J. E. Pecero, A. Barrondo, and E. Schaeffer. Adaptive energy efficient scheduling in Peer-to-Peer desktop grids . *Future Generation Computer Systems*, 36:209 – 220, 2014.

[15] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.