A Distributed Game-Theoretic Approach to IaaS Cloud Brokering

Jakub Gąsior(✉) and Franciszek Seredyński

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

Abstract. We consider the problem of profit optimization for cloud brokerage service in the IaaS environment. We replace this optimization problem with 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 the Spatial Prisoner's Dilemma (SPD) game and a phenomenon of collective behavior of players participating in the game composed of two classes of automata-based agents - Cellular Automata (CA) and Learning Automata (LA). We introduce dynamic strategies like local profit sharing, mutation, and competition, which stimulate the evolutionary process of developing collective behavior among players to maximize their profit margin. We present the results of an experimental study showing the emergence of collective behavior in such systems.

Keywords: Collective behavior · Multi-agent systems · Spatial prisoner's dilemma game · Cellular automata · Infrastructure as a service

1 Introduction

Cloud Computing (CC) is a term used with increasing frequency in the past few years, as its popularity continues to grow. They can reduce the cost and complexity of owning and operating computers and networks. In an Infrastructure-as-a-Service (IaaS) cloud, this is achieved by using Virtual Machines (VMs), which can be dynamically assigned to the resources according to the demand and availability, as well as a possibility of consolidating several such VMs into the same virtual server.

As a result, an IaaS system can offer such on-demand computational services at a low cost. Cloud users usually pay for the usage (counted by the number of instance-hours incurred) in a pay-as-you-go model and are therefore freed from the prohibitive upfront investment on infrastructure, which is usually over-provisioned to accommodate peak demands [13].

Users may be charged in several different ways to access such resources. For example, it could be either a long-term reservation or a quick on-demand lease [13]. In long-term reservations, customers pay a fee to reserve a certain amount of

R. Chaves et al. (Eds.): Euro-Par 2021, LNCS 13098, pp. 81–90, 2022.
https://doi.org/10.1007/978-3-031-06156-1_7

computing resources for a period of one or several years. Then, they get charged extra for actually using the resource. While the extra payment may be lower than comparable on-demand service, it may not be desirable for each customer to pay for the more extended reservation period if their workload is relatively light or unpredictable [3].

One of the answers to this problem is cloud service brokering [4], a model in which a trusted third party matches the needs of customers with services of cloud providers. Typically, brokers' service is to find the best deals among a set of clouds that best fit the user requirements. Brokers consider the price and many other factors, such as privacy and security issues, Service Level Agreements (SLAs), performance, and they might offer solutions integrating services from multiple service providers [8].

In this paper, we analyze a game-theoretic approach to consider a problem of multi-broker job allocation and scheduling, aiming to optimize the brokers' profit while maintaining a Quality of Service (QoS) level acceptable to the customer. We propose a fully distributed approach based on converting this optimization problem into a game-theoretic one, where brokers representing users' demands will search for a solution in the form of a Nash equilibrium. For this purpose we will use a variant of SPD game proposed by [5] in the context of Cellular Automata (CA) space.

The remainder of this paper is organized as follows. In the next section, works related to the subject of our study are discussed. Section 3 describes the proposed Cloud Computing (CC) system model and defines the scheduling problem. Section 4 demonstrates the performance metrics, the input parameters, and the experimental results. Finally, Sect. 5 concludes the paper.

2 State of the Art

A multi-objective approach to the cloud brokering problem was recently proposed in [7], where authors provide a dichotomic approach to minimize the cost of service and a second - negatively correlated - objective. The novelty of the approach lies in considering services that can be sold in bundles, in which a set of services is sold together for a lower price than the sum of individual services' prices. In [6], the authors introduced a brokering system for scientific workflows, which optimizes a multi-criteria problem using an aggregated objective function. The brokering part of the system selects the length of the service period to minimize VMs lease cost.

The idea of broker exploitation of pricing model was also studied in [13] and solved using approximate dynamic programming. The theoretical study of users' requests aggregation under a concave cost function assumption and Randomized Online Stack-Centric Scheduling Algorithm (ROSA) was proposed in [14]. In their paper, authors proved the lower bound of the proposed solution's competitive ratio and evaluated its performance with trace-driven simulation using Google cluster data.

Aazam [2] proposed a dynamic broker, which predicts users' behavior based on the so-called relinquish probability, i.e., the likelihood that the user will cease

to use the requested services. The study also involves an advanced refund mechanism based on multiple criteria. It is further extended to the Amazon cloud model and includes historical record integration in [1].

Similarly, in [10], the authors introduced an adaptive learning system that allows the analysis of the sequence of negotiation offers received by the broker for effectively learning the opponent's behavior over several stages of the negotiation process. They formulated this issue as the multi-stage Markov decision problem to suggest the broker with appropriate counter-offer tactics. Authors claim their solution can outperform the existing fixed behavioral learning schemes and maximize the utility value and success rate of negotiating parties without any break-offs.

Closer to our work, in [3] authors analyzed the scenario of user cost minimization in mobile cloud computing (MCC) networks, where multiple cooperative brokers assign cloud resources to mobile users. The work investigated two classes of cloud reservation strategies, i.e., a competitive strategy and a compete-then-cooperate strategy as a performance bound, showing that noticeable cooperative gains can be achieved over the pure competition in markets with only a few brokers. In contrast, the cooperative gain becomes marginal in more crowded markets.

A similar combinatorial auction-based algorithm was proposed in [9]. Authors aimed to solve the optimization problem where cloud users submit their requirements, and in turn, vendors submit their offers containing the price, QoS, and their prepared sets of resources. Results for procurement cost and scalability on a large number of cloud vendors were verified using various standard distribution benchmarks, including random, uniform, decay, and CATS.

3 Multi-objective Scheduling in Cloud Environment

3.1 Cloud Brokering Model

We assume that the Cloud Service Provider (CSP) offers abundant computing capacity at any given time. The Cloud Resource Broker (CRB) purchases computational resource from IaaS provider and has to pay for the resource cost. For the purpose of this paper we follow the specification of Compute Optimized VM series provided by Amazon EC2 and shown in Table 1[1]. Broker then offers to sell a set of VM instances $M_1, M_2, ..., M_m$, specified by several characteristics, including a number of cores $P(M_i)$, memory $M(M_i)$, storage space $S(M_i)$ and cost per hour $C^B(M_i)$.

Cloud Service Users (CSUs) $(U_1, U_2, ..., U_n)$ submit to the broker their workflow applications J_k^j for execution. Each application is the set of n tasks or jobs. Users are expected to pay appropriate fees to the broker dependent on the 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

[1] The price is for Linux Instances (EU Frankfurt) with full upfront payment on 1-year term reservation as of July, 2021.

Table 1. Compute Optimized Dedicated VM Instances in Amazon EC2.

	vCPU	Memory (GiB)	Price (Reserved)	Price (On-demand)
c4.large	2	3.75	$0.074	$0.114
c4.xlarge	4	7.5	$0.146	$0.227
c4.2xlarge	8	15	$0.293	$0.454
c4.4xlarge	16	30	$0.586	$0.909
c4.8xlarge	36	60	$1.173	$1.817

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 and a deadline d_k^j.

The broker's cost function $C^B(M_i)$ is dependent on both the prices of reserved cloud resource instances and on-demand instances (as defined in Table 1). The cloud broker is capable of leveraging the pricing gap between reserved (C^R) and on-demand (C^{OD}) instances to reduce the expenses of all the users. To attract customers, CRBs should charge for a VM lease less than the on-demand pricing (C^{OD}) offered by the cloud provider. In order to ensure a reasonable profit for the broker, we assume that the broker's asking price $C^B(M_i)$ will be 25% lower than the on-demand price requested by the cloud provider.

We consider the multi-broker resource scheduling problem for IaaS clouds, where multiple customers may submit their job requests to a broker at random instants with a random workload that should be fulfilled before a specified deadline. We assume that the inter-arrival times for job requests are arbitrary. If the broker cannot accommodate the request to finish execution before the specified deadline, it must either use a larger VM instance offering more processing capacity or buy additional on-demand instances to fulfill the customer's request. Both solutions account for a negative impact on the broker's profit [8].

From a global system perspective, an appealing design objective is to find the allocation strategy for all brokers' submissions that minimizes all requests' average cost by cooperatively deciding on the reservation and task outsourcing strategies of all the brokers.

However, different brokers may be run by different organizations and may be selfish and only willing to maximize their profits. In other words, there is no incentive for the brokers to cooperate if the resulting profit is not higher compared with that achievable through pure competition [3].

4 A Game-Theoretic Approach to IaaS Multi-broker Scheduling

To mitigate this issue, we introduce an agent-based game-theoretic distributed scheduling scheme. We consider a two-dimensional CA lattice of the size $n \times m$. Each cell of the CA has a Moore neighborhood of radius r and a rule, which depends on its neighborhood state. Each cell of a 2D CA will be considered an agent (player) participating in the SPD game [5]. Each player (a cell of CA) has two possible actions: C (cooperate) and D (defect).The payoff function of the game is given in Table 2.

Table 2. Payoff function of a row player participating in the SPD game.

Player's action	Opponent's action	
	Cooperate (C)	Defect (D)
Cooperate (C)	$R = 1$	$S = 0$
Defect (D)	$T = b$	$P = a$

Each player associated with a given cell plays a game with each of his eight neighbors in a single round, collecting their 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 a non-uniform CA, with one of the following rules: *all-C* (always cooperate), *all-D* (always defect), and *k-D* (cooperate until no more than k $(0 \leq k \leq 7)$ neighbors defect).

A player may change his current strategy into another by comparing his total score collected during q rounds with his neighbors' scores. He selects as his new strategy the best performing neighbor's strategy, i.e., the player whose total collected 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 following q rounds of interaction.

It is worth to notice that choosing the action D by all players corresponds to the Nash equilibrium (NE) point. Looking from the point of view of players' global collective behavior, this average total payoff of all players in NE point is low. Instead, we would expect the players to choose the action C, which provides the highest value of the average total payoff of all players equal to 1. For this instance of the game, it is the maximal value of a possible average total payoff of all players, and it will be achieved when all players decide to select the action C. We are interested in studying conditions when such behavior of players in iterated games is achievable.

4.1 CA–Based Players

We will be using CA–based agents as the first type of participant in the game. CAs are spatially and temporally discrete computational systems initially proposed by Ulam and von Neumann and today are a powerful tool used in computer science and natural science to solve problems and model different phenomena.

When a cell (i, j) is considered a CA-based player, it will be assumed that it is a part of the 2D array, and at a given discrete moment t, each cell is either in state C or D. The state's value is used by CA–based player as an action with an opponent player. For each cell, a local neighborhood is defined. Because we employ a 2D finite space, a cyclic boundary condition is applied.

In discrete moments, CA–based players will select new actions according to local rules (also called strategies or transition functions) assigned to them, which will change the states of the corresponding cells. We will be using several rules, among which one of them will be initially randomly assigned to each CA cell, so we deal with a non-uniform CA.

We will consider two types of CA–based players. To cells of the first type, one of the following rules: *all–C*, *all–D*, and *k–D* will be assigned. The second type of CA–based player uses probabilistic CA. To cells of this type, the following rule will be assigned: *cooperate* with probability p_{coop} or *defect* with probability $1 - p_{coop}$, where p_{coop} is some predefined value.

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 the possibility of changing its rules is called a second-order CA. In opposite to a classical CA, a second-order CA has the potential to solve various optimization problems.

4.2 LA–Based Players

We will also employ a deterministic ϵ–LA as the second group of players in the considered game. The ϵ–LA has $d = 2$ actions and acts in a deterministic environment $c = (c_1, c_2, ..., c_{2*d})$, where c_k stands for a reward defined by the payoff function from Table 2 obtained for its action and action of his opponent (CA or LA–based player) from the Moore neighborhood. It also has a memory of length h and a reinforcement learning algorithm that selects a new action. In our case, C and D are actions of an automaton, and they are associated with states of the array cells occupied by LA–based players.

Whenever ϵ–LA generates action, and its opponent from a neighborhood selects an action, the local environment (payoff function) sends it a payoff in a deterministic way. The objective of a reinforcement learning algorithm represented by ϵ–LA is to maximize its payoff in an environment where it operates.

The automaton remembers its last h actions and corresponding payoffs from the last h moments. As the next action ϵ–LA chooses its best action from the last h games (rounds) with the probability $1 - \epsilon$ $(0 < \epsilon \leq 1)$, and with the probability ϵ/d any of its d actions.

4.3 Sharing, Mutation and Competition Mechanisms in the Game

In this paper, we are more interested in incorporating the global goal of the system into the local interests of individual brokers. In the following, we assume that action (C) is considered an equivalent of the cooperation in a classic PD game and denotes a situation where brokers are willing to share their unused allocation slots within their VMs while receiving a partial payment from other brokers. On the other hand, action (D) means that the broker declines to participate in resource sharing, which is considered an equivalent of the defection (D) in a classic PD game.

To study a possibility of the emergence of global collective behavior of players in the sense of the second class of the collective behavior classification [11] we introduce additional mechanisms of local interaction between players, which can be potentially spread or dismissed during the evolution.

The first mechanism is a competition, where after a q rounds (iterations), each agent compares its total payoff with its neighbors' total payoffs. If a more successful player exists in the neighborhood, this player replaces their own rule with the most successful one. This mechanism converts a classical CA into the *second–order* CA, which can adapt in time. When both players are CA-based players, a rule of a given player is replaced by a rule of the most successful players, and the value of the sharing tag is copied. If both players are LA–based players, then replacing happens only if the best player differs in at least one value of such parameters as h, ϵ, or a sharing tag. If one player is a CA-based player and the other one is an LA-based player, then a player of the most successful class replaces a given player.

The second mechanism used is a mutation of 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. Also, parameters h and ϵ of LA–based agents can be a subject of mutation.

The third mechanism called an *Income Sharing Mechanism* (ISM) provides a possibility of sharing payoffs between players. 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. Before starting the iterated game, each player turns on its tag with a predefined probability $p_{sharing}$. Due to the competition mechanism, rules with tags containing information about willingness to share incomes can be potentially spread or dismissed during the system's evolution.

5 Experimental Analysis and Performance Evaluation

In this section, we evaluate the performance of the considered classes of agents and introduced mechanisms of interactions (mutation, competition, sharing) and their impact on the emergence of cooperative behavior between brokers and their overall performance.

Optimal parameters for the SPD game were adapted from our earlier paper [12] and are as follows. A 2D array of the size of 4×4 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 probability 0.7, and the remaining three rules (*all–C*, *all–D*, probabilistic CA) with probability 0.1. When k–D was applied, k was randomly selected from the range 0–7. If the competition mechanism is turned *on*, updating the array cells (by a winner in a local neighborhood) is conducted after each iteration ($q = 1$). Parameters of the payoff function were set to $a = 0.3$ and $b = 1.4$, respectively. We incorporate these settings into the proposed scheduler to find a job allocation schedule maximizing the broker's income and minimizing the need for procuring additional resources on-demand.

We conduct simulations using the Google cluster trace data, which has been widely employed to perform cloud computing-related simulations. From the above dataset, we generate sample workload batches in the range of 1000–10000 jobs. The experimental scenarios are encoded as follows: $\{\text{Agent}, N, M\}$, where *Agent* denotes the type of employed automata, N - number of brokers and M - number of VM instances, i.e., $\{\text{CA}, 4, 10\}$ denotes scenario employing $N = 4$ brokers in the system with $M = 10$ VM instances and CA–based players.

Jobs were then scheduled by independent brokering agents (in the range $N = \{2, 4, 8, 16\}$) on cloud infrastructure containing $M = 10$, 20, 30, and 40 reserved VM instances using a fast *Minimum Time Maximum Profit* list heuristic [8] and a proposed game-theoretic space-sharing scheme.

We analyze two different performance metrics. First, the *Scheduling Success rate*, which denotes the ratio of completed job requests, i.e., without the need to procure additional on-demand resources. Similarly to [8], we do not count such events as SLA violations. In such cases, a broker will be forced to cover additional lease costs implying lower profits. The second analyzed metric is the aggregate profit improvement resulting from cooperation and multiplexing of job requests between individual brokers. Table 3 reports the average improvement over the results achieved using a simple list scheduling heuristic.

Let us start with a comparison between CA and LA–based agents. In most cases, LA–based players achieve better results than their CA–based counterparts. It might be because CA-based players do not have learning abilities, and the value of the average payoff is a result of the initial settings. In contrast to CA, LA–based players are aware of their environment, they can learn and adapt, and the average payoff depends upon a given memory size and the ϵ–value.

We can also notice that, as the congestion increases, the profit improvement decreases, i.e., the additional lease costs arise as the demand for resources increases. We also observe that ISM's benefits have a more significant impact on larger systems with a higher number of available VMs. As can be seen, the profit improvements increase on average by 8.6% points as the number of VM instances increases from 10 to 40. This means that cooperation is more beneficial in less crowded scenarios, while the benefit is only marginal if the number of job requests is high compared to a number of available VMs instances.

Table 3. Averaged Scheduling Success Rate and Profit Improvement results for multiple scheduling scenarios computed with SPD Scheduler (using Income Sharing and Mutation mechanisms) and Minimum Time Maximum Profit list heuristic.

Problem instance	Scheduling success rate [%]			Profit improvement [%]		
	MinTMaxP	SPD-ISM	SPD-Mut	MinTMaxP	SPD-ISM	SPD-Mut
{CA, 2, 10}	86.35	**89.24**	86.89	5.31	**5.78**	5.41
{CA, 4, 20}	89.78	91.31	**91.85**	7.51	**8.52**	7.89
{CA, 8, 30}	91.24	**92.21**	90.74	9.11	**9.52**	8.89
{CA, 16, 40}	92.45	**93.56**	91.56	12.41	**13.75**	13.24
{LA, 2, 10}	89.54	**90.78**	90.41	5.81	**6.07**	5.84
{LA, 4, 20}	91.44	**91.65**	90.97	9.44	9.26	**9.91**
{LA, 8, 30}	90.84	91.35	**91.57**	10.11	**10.86**	10.31
{LA, 16, 40}	93.33	**93.55**	92.89	13.58	13.76	**14.57**
{CA+LA, 2, 10}	89.98	**90.14**	89.74	5.75	**6.23**	5.89
{CA+LA, 4, 20}	91.24	**92.45**	91.45	8.42	**8.89**	8.74
{CA+LA, 8, 30}	92.98	**93.78**	92.48	8.11	9.89	**9.94**
{CA+LA, 16, 40}	95.54	95.75	**96.87**	14.41	15.37	**16.22**

This could also be attributed to the higher ratio of deadline violations in smaller-scale experiments. In such cases, due to a larger number of job requests, resource sharing between the brokers become less profitable due to the increasing number of additional on-demand leases required to meet the SLA requested by the customers.

6 Conclusion and Future Work

We have presented a theoretical framework to study the behavior of heterogeneous multi–agent systems composed of two classes of automata–based agents: CA and LA agents operating in an environment described in terms of a spatial PD game. This framework was defined to solve global optimization tasks in a distributed way by agents' collective behavior.

We incorporated this framework into the paradigm of a multi-broker job allocation scenario within the IaaS architecture 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 any centralized control and introduced several desirable properties such as adaptation and self-organization.

A set of conducted experiments has shown that these proposed solutions are promising building blocks that enable the emergence of global collective behavior in heterogeneous multi-agent systems. Conditions of the emergence of such systems' global behavior may depend on several additional parameters, and these issues will be a subject of our future work.

References

1. Aazam, M., Huh, E., St-Hilaire, M., Lung, C., Lambadaris, I.: Cloud customer's historical record based resource pricing. IEEE Trans. Parallel Distrib. Syst. **27**(7), 1929–1940 (2016). https://doi.org/10.1109/TPDS.2015.2473850
2. Aazam, M., Huh, E.N.: Cloud broker service-oriented resource management model. Trans. Emerg. Telecommun. Technol. **28**(2), 29–37 (2017). https://doi.org/10.1002/ett.2937
3. Guan, Z., Melodia, T.: The value of cooperation: minimizing user costs in multi-broker mobile cloud computing networks. IEEE Trans. Cloud Comput. **5**(4), 780–791 (2017). https://doi.org/10.1109/TCC.2015.2440257
4. Guzek, M., Gniewek, A., Bouvry, P., Musial, J., Blazewicz, J.: Cloud brokering: current practices and upcoming challenges. IEEE Cloud Comput. **2**(2), 40–47 (2015). https://doi.org/10.1109/MCC.2015.32
5. Katsumata, Y., Ishida, Y.: On a membrane formation in a spatio-temporally generalized prisoner's dilemma. In: Umeo, H., Morishita, S., Nishinari, K., Komatsuzaki, T., Bandini, S. (eds.) ACRI 2008. LNCS, vol. 5191, pp. 60–66. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79992-4_8
6. Kim, S., Kang, D., Kim, W., Chen, M., Youn, C.: A science gateway cloud with cost-adaptive VM management for computational science and applications. IEEE Syst. J. **11**(1), 173–185 (2017). https://doi.org/10.1109/JSYST.2015.2501750
7. Musial, J., et al.: Cloud brokering with bundles: multi-objective optimization of services selection. Found. Comput. Decis. Sci. **44**, 407–426 (2019). https://doi.org/10.2478/fcds-2019-0020
8. Nesmachnow, S., Iturriaga, S., Dorronsoro, B.: Efficient heuristics for profit optimization of virtual cloud brokers. IEEE Comput. Intell. Mag. **10**(1), 33–43 (2015). https://doi.org/10.1109/MCI.2014.2369893
9. Prasad, G.V., Prasad, A.S., Rao, S.: A combinatorial auction mechanism for multiple resource procurement in cloud computing. IEEE Trans. Cloud Comput. **6**(4), 904–914 (2018). https://doi.org/10.1109/TCC.2016.2541150
10. Rajavel, R., Thangarathanam, M.: Adaptive probabilistic behavioural learning system for the effective behavioural decision in cloud trading negotiation market. Futur. Gener. Comput. Syst. **58**, 29–41 (2016). https://doi.org/10.1016/j.future.2015.12.007
11. Rossi, F., Bandyopadhyay, S., Wolf, M., Pavone, M.: Review of multi-agent algorithms for collective behavior: a structural taxonomy. IFAC-PapersOnLine **51**(12), 112–117 (2018). https://doi.org/10.1016/j.ifacol.2018.07.097. iFAC Workshop on Networked & Autonomous Air & Space Systems NAASS 2018
12. Seredyński, F., Gąsior, J.: Emergence of collective behavior in large cellular automata-based multi-agent systems. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) ICAISC 2019. LNCS (LNAI), vol. 11509, pp. 676–688. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20915-5_60
13. Wang, W., Niu, D., Liang, B., Li, B.: Dynamic cloud instance acquisition via IaaS cloud brokerage. IEEE Trans. Parallel Distrib. Syst. **26**(6), 1580–1593 (2015). https://doi.org/10.1109/TPDS.2014.2326409
14. Zhang, R., Wu, K., Li, M., Wang, J.: Online resource scheduling under concave pricing for cloud computing. IEEE Trans. Parallel Distrib. Syst. **27**, 1131–1145 (2015). https://doi.org/10.1109/TPDS.2015.2432799