Int. J. Appl. Math. Comput. Sci., 2016, Vol. 26, No. 2, 391–406
DOI: 10.1515/amcs-2016-0028

EXACT AND HEURISTIC APPROACHES TO SOLVE THE INTERNET SHOPPING OPTIMIZATION PROBLEM WITH DELIVERY COSTS

MARIO C. LOPEZ-LOCES [a], JEDRZEJ MUSIAL [b,*], JOHNATAN E. PECERO [c],
HECTOR J. FRAIRE-HUACUJA [a], JACEK BLAZEWICZ [b,d], PASCAL BOUVRY [c]

[a]National Institute of Technology of Ciudad Madero, National Institute of Technology of Mexico
v. 1o. de Mayo esq. Sor Juana Inés de la Cruz s/n, Col. Los Mangos C.P. 89440, Cd. Madero, Tamaulipas, Mexico
e-mail: mario.cesar@me.com, automatas2002@yahoo.com.mx

[b]Institute of Computing Science
Poznań University of Technology, ul. Piotrowo 2, 60-965 Poznań, Poland
e-mail: {Jedrzej.Musial,Jacek.Blazewicz}@cs.put.poznan.pl

[c]Computer Science and Communications Research Unit
University of Luxembourg, 6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg
e-mail:johnatan.pecero@gmail,compascal.bouvry@uni.lu

[d]Institute of Bioorganic Chemistry
Polish Academy of Sciences, ul. Z. Noskowskiego 12/14, 61-704 Poznań, Poland

Internet shopping has been one of the most common online activities, carried out by millions of users every day. As the number of available offers grows, the difficulty in getting the best one among all the shops increases as well. In this paper we propose an integer linear programming (ILP) model and two heuristic solutions, the MinMin algorithm and the cellular processing algorithm, to tackle the Internet shopping optimization problem with delivery costs. The obtained results improve those achieved by the state-of-the-art heuristics, and for small real case scenarios ILP delivers exact solutions in a reasonable amount of time.

Keywords: Internet shopping optimization, integer linear programming, cellular processing algorithm, heuristic algorithms, optimization in e-commerce.

1. Introduction

The acquisition of products online is one of the most common activities for which the Internet is used today. This is because setting up online shops is easier than ever before, especially for small enterprises and individuals. On the other hand, the increase in competition leads to a reduction in the prices of the products, making them more attractive for potential buyers.

For sellers, the main advantage is that their offers are available to a larger audience, without most of the associated costs, like rent, taxes, maintenance, and advertising. Buyers have the convenience of being able to make their purchases from anywhere, at any time, with better prices and from a wider range of products, as long as they have access to the Internet.

Tenders are not limited to physical items. For instance, one can notice cloud brokering, which is a flourishing industry (Guzek *et al.*, 2015) that offers a huge variety of services. From the point of view of the optimization of Internet shopping, they can be treated exactly as physical items sold via the Internet.

However, as a worldwide market is becoming more common, clients searching for products on their shopping list can feel overwhelmed by the process, due to the immeasurable amount of shops that exist online.

The Internet shopping optimization problem (ISOP) (Musial, 2012) arises when a customer with a shopping list wants to purchase the products in a set of online

*Corresponding author

shops at the minimum cost, taking into account also the delivery costs associated with the shops where one or more products are bought.

This problem was formally modeled as an optimization problem by Blazewicz *et al.* (2010), who presented a proof that it belongs to the NP-complete set and proposed two deterministic polynomial algorithms to solve special cases of the newly described problem: greedy (Wojciechowski and Musial, 2010) and forecasting (Blazewicz *et al.*, 2014b). These heuristics are used to validate the correctness and performance of the proposed solution methods presented in this article.

In previous research, different versions (specializations) of the ISOP have been examined, e.g., a version with price sensitive discounts (Blazewicz *et al.*, 2014a; 2014b) and dual discounting functions, both for price and shipping discounts (Blazewicz *et al.*, 2016). For example, due to the NP-hardness of the optimization problem, Wojciechowski and Musial (2010) designed a heuristic solution to optimize the shopping basket and evaluate it for the customer basket optimization problem to make it applicable for solving complex shopping cart optimization in on-line applications. The first idea of an algorithm solving the ISOP with discounts was presented by Blazewicz and Musial (2011). Moreover, it has been proven that the problem is not approachable in polynomial time (Blazewicz *et al.*, 2010). The archetype of the presented problem was a web-based customer assistance system dedicated to pharmacy shopping that helps customers to find shops in a geographically defined area where the entire shopping list could be fulfilled at the best total price (Wojciechowski and Musial, 2009).

In this paper, we propose an exact solution and two approximate methods to solve the ISOP. The proposed exact solution is an integer linear programming (ILP) model. The first approximate approach is a cellular computing-based algorithm, which consists of simple processing cells that operate on their own population and that share information with other cells to improve the performance of the whole system. The second approximate algorithm is a deterministic heuristic with a local search that selects iteratively the product with the minimum cost from all products and is purchased from the shop in which its cost is also a minimum.

In order to test the correctness and performance of the proposed methods, both exact and approximate instances of the ISOP were created according to the method described by Blazewicz *et al.* (2010), where the name of the instance describes its size, m represents the number of shops, and n the number of products. For example, an instance set 5n20m contains instances of the ISOP with shopping lists of 20 products, available from at most 5 shops.

Therefore, three sets of different sizes, a Small instance set (subsets 3n20m, 4n20m and 5n20m), a Medium instance set (subsets 5n240m, 5n400m and 50n240m), and a Large instance set (subsets 50n400m, 100n240m and 100n400m), were generated. To evaluate the performance in a real case scenario, the algorithms were tested with an instance involving the prices and availability of products from existing online shops.

The results indicate that the ILP model delivers the exact solution in a reasonable time for instances of small sizes. For larger instances, the cellular computing approach obtains better quality solutions than previous algorithms described in the literature. The quality of the solutions is consistent as the size of the instances being solved increases.

To complement Introduction, it is worth mentioning different aspects of Internet shopping. The price optimization tool is definitely a new feature that benefits the customer. Moreover, we can point to such a positive aspect as the much easier stage of information searching (Rose and Samouel, 2009). Furthermore, it should be noted that the Internet customer is under the influence of many factors (Cheung *et al.*, 2005) while shopping online (both internal and external ones).

However, there is evidence that the new kind of shopping experience, where we use the Internet as a shopping platform, may lead to some problematic behavior. One can say that some customers doing the shopping via the Internet may become addicts, very similarly to addictions connected with gaming and general Internet dependency. Many addictive forms of consumption, as well as negative behavior connected with shopping, have been already widely described in business and medical journals. Following the definition of `BusinessDictionary.com`, one can say that shopping is "the process of browsing and/or purchasing items in exchange for money." The platform is a less important aspect when we think globally about shopping. Therefore, it is natural that many of these negative behaviors known from the business and medical literature can be included in the Internet shopping experience. One may consult Rose and Dhandayudham (2014) for more information about different types of online shopping behavior and addiction.

Since we have observed many different aspects of online shopping, it could be possible to tackle the shopping problem from a very general perspective and to introduce it as a multi-objective problem, a multi-objective decision-aided ISOP, a system/tool where a customer makes some decisions with respect to their personal shops, such as trust, delivery time, negative factors, and all other significant elements. While facing the problem, customers could be attracted by and provided with the specific tool in an attempt to help them make reasonable decisions (decisions that will not change the price significantly). We refer to the work of Sawik (2012) to see an interesting approach to providing decision

Table 1. Table of notation.

Symbol	Explanation
M	set of shops
N	set of products
m	number of shops, $\|M\|$
n	number of products, $\|N\|$
i	shop indicator
j	product indicator
N_i	multiset of products available from shop i
d_i	delivery price of all products from shop i
c_{ij}	cost of product j in shop i
x_{ij}	0–1 usage indicator for product j in shop i
y_i	0–1 usage indicator for shop i
T	cumulative value of all products bought in all shops
T_i	cumulative value of all products bought from shop i
$f_i(T_i)$	piecewise function for all products (T_i) bought from shop i
$X = (X_1, \ldots, X_m)$	sequence of selections of products from shops $1, \ldots, m$
$F(X)$	sum of product and delivery costs
$\delta(X)$	0–1 indicator function for $x = 0$ and $x > 0$
X^*	optimal sequence of selections of products
F^*	optimal (minimum) total cost

makers with a simple tool that helps them to make good decisions in the financial market.

A multi-objective general ISOP could be an interesting further step in our research. However, our attention is now focused on price optimization. The price optimization problem, models, and algorithms have to be presented, discussed, and solved to enable further study.

This paper is organized as follows. Section 2 presents a formal definition of the ISOP with the notation used throughout the paper. Section 3 describes related research and existing approaches to solving the ISOP. Section 4 describes the exact methods that have been proposed to solve to optimality instances of the ISOP. Section 5 presents the approximated methods that have been developed to solve larger instances of the ISOP with delivery costs. A description of the computational experiments performed can be found in Section 6, and the obtained results are presented in Section 7. Finally, we conclude in Section 8 with an analysis of the obtained results and a proposal for future research.

2. Problem definition

The notation used in this paper is given in Table 1. The ISOP is defined in the following way: Suppose being given a shopping list $N = \{j = 1, \ldots, n\}$, where n is the number of products, and a list of available shops $M = \{i = 1, \ldots, m\}$, where m is the number of shops. The multiset N_i contains the products available from shop i. Each product $j \in N_i$ costs c_{ij} and has a delivery cost from that shop of d_i. The delivery cost is charged once if one or more products are purchased from shop i.

The ISOP is the minimization of the total cost of the shopping list N, including delivery costs. This is formally described as the finding of a disjoint selection of the products purchased from the different shops $X = (X_1, \ldots, X_m)$, such that from the selection of products that are available at a given shop $X_i \subseteq N_i$, all the products in the shopping list are purchased, $\bigcup_{i=1}^{m} X_i = N$, and the total cost, including the delivery cost from the shops from which one or more products are selected, is minimized: $F(X) = \sum_{i=1}^{m} \left(\delta\left(|X_i|\right) d_i + \sum_{j \in X_i} c_{ij} \right)$, where $|X_i|$ is the cardinality of the multiset X_i, and $\delta(x) = 0$ if $x = 0$ and $\delta(x) = 1$ if $x > 0$.

3. Related research

There are some similarities between the ISOP and the well-known facility location problem (FLP). A simple FLP is the Weber problem (Weber, 1929), which is based on Simpson's idea (Simpson, 1750). The main characteristics of the FLP are the space, a metric, given customer locations, and given or not given positions for facility locations. A traditional FLP is to open a number of facilities in arbitrary positions of the space (the continuous problem) or in a subset of given positions (the discrete problem), and to assign customers to the opened facilities so that the sum of the opening costs and the costs related to the distances between customer locations and their corresponding facility locations is minimized.

Discussions of FLPs can be found in the vast literature (Revelle *et al.*, 2008; Krarup *et al.*, 2002; Eiselt and Sandblom, 2004; Melo *et al.*, 2009; Iyigun and Ben-Israel, 2010). The traditional discrete FLP is NP-hard (Garey and Johnson, 1979) in the strong sense. Note, however, that the general ISOP with price discounts cannot be treated as a traditional discrete FLP because there is no evident motivation for a discount on the cumulative cost in the sense of distances. It can be noted that this problem and the ISOP are not sub-cases of each other, while the traditional discrete FLP is a special case of none of these problems.

Taking into consideration the ISOP and price discounts, one can note some similarities with the total quantity discount problem (TQD) (Goossens *et al.*, 2007). To show the similarities and, most of all, to show distinct differences, we should include a mathematical formulation of the TQD. One can define G as a set of n goods, indexed by k, and S as the set of n suppliers, indexed by i. For each good k in G, d_k is the given amount of good k to be procured. To each supplier i in S, we associate a sequence of intervals $Z_i = \{0, 1, \dots, \max(i)\}$, indexed by j. Furthermore, for each supplier $i \in S$ and interval $j \in Z_i$, l_{ij} and u_{ij} define the minimum and maximum number of goods, respectively, that need to be ordered from supplier i to be in the interval j. Finally, for each supplier $i \in S$, for each interval $j \in Z_i$, and each good $k \in G$, let c_{ijk} be the price of one unit of good k purchased from supplier i in its j-th interval.

The similarities between the ISOP with price discounts and the TQD problem can be noticed if we treat the products to be bought N as goods G (d_k is the amount of the same good k), and the shops M as suppliers S. A piecewise discounting function f_i for shop i will be associated with a sequence of intervals Z_i for supplier i. The price c_{ij} of product j from shop i can be seen as the price p_{ik} for one unit of good k purchased from supplier i. The piecewise function for shop i applied to a product price f_i (c_{ij}) should be treated as p_{ijk}: the price for one unit of good k purchased from supplier i in its j-th interval. However, the ISOP includes shipping costs that are specific to each shop. This feature makes the ISOP a new enhanced version of the TQD problem.

It is worth noting that the decision version of the TQD problem is strongly NP-complete. Moreover, no polynomial-time approximation algorithm with a constant worst-case ratio exists for the TQD problem (unless the complexity class P is equal to the complexity class NP). More information on many variations on the TQD can be found in the literature (Goossens *et al.*, 2007; Mirmohammadi *et al.*, 2009; Munson and Hu, 2010; Krichen *et al.*, 2011).

As can be seen, the related literature has been focused on the complexity of the problem, while in this paper we will concentrate on the optimization point of view by proposing exact and approximate solution methods for the ISOP with delivery costs. Note that the ISOP will be applied to shopping on an Internet website in the future, and therefore it is significantly important to minimize the algorithm's computation time (Marszalkowski *et al.*, 2014; Marszalkowski and Musial, 2011).

4. Integer linear programming model

To solve instances of the ISOP with delivery costs to optimality, we proposed an INTEGER LINEAR PROGRAMMING (ILP) MODEL that will be described next. For the details of the notation used in this model, one can consult Table 1.

The binary variable x_{ij} indicates whether a product j is purchased from shop i, the binary variable y_i indicates if at least one product is purchased from shop i:

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} d_i y_i \tag{1}$$

such that

$$x_{i,j} \in \{0,1\}, \quad \forall i \in M, \forall j \in N, \tag{2}$$

$$y_i \in \{0,1\}, \quad \forall i \in M, \tag{3}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} = n, \tag{4}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad \forall j \in N, \tag{5}$$

$$ny_i - \sum_{j=1}^{n} x_{ij} \geq 0, \quad \forall i \in M. \tag{6}$$

In this ILP, the objective function is shown in (1), which is the total cost of purchasing a shopping list from the selected shops, including delivery costs, subject to the following constraints. The constraint (4) ensures that the number of purchased products is equal to the number of products on the shopping list, while constraint (5) guarantees that only one product of each kind is selected; the constraint (6) ensures that the variable y_i takes the value of 1 when a product is purchased from shop i.

As one can see, this is an integer model, since the decision variables, x_{ij} and y_i, are integer-valued. Additionally, the model is linear inasmuch as the objective function and the functions in the constraints are linear.

The correctness of the proposed ILP was validated with an enumerative algorithm. This method generates all the m^n combinations of shopping lists, which ensures obtaining the optimal solution in all cases, but only for instances of a very small size.

The same group of instances was solved with ILP and both results were compared, to detect discrepancies. This test confirmed that ILP obtained the correct optimal solution on the Small instance set.

5. Heuristic methods

In this section we will present some heuristic approaches to solving the ISOP with delivery costs. The methods that we propose include variants of the MinMin algorithm with a local search stage (MinMin) and a cellular computing approach (cellular processing algorithm) that will be described in the following subsections.

5.1. MinMin and its variants. The MinMin algorithm is well known in the field of scheduling and widely described in the literature (Freund *et al.*, 1998; Diaz *et al.*, 2014; Nesmachnow *et al.*, 2013) as a representative of static task scheduling algorithms, where each task should be assigned to a resource. It is divided into two phases:

– the first phase is to set/calculate all possible expected completion times for every task and resource: a matrix of dimensions $t \times n$;

– during the second phase, all tasks are scheduled, based on the minimum completion times. In every step of this algorithm, the task with the minimum completion time is assigned to the corresponding resource and removed from the list of tasks left to schedule.

One of the main advantages of this approach is that, in general, it is capable of obtaining good quality solutions with a relatively small computational cost.

The MinMin pseudocode is shown in Algorithm 1. The algorithm starts with an empty sequence of selection of products X. From the set N of available products, we search for the one that has the lowest price if added to X, taking into account the selling price of the one and the delivery cost of the shop.

The element that the least increases the total cost of the current solution is added to X. After that, the product previously assigned is removed from the shopping list N and from the multiset N_i.

This process continues until all products have been purchased, in other words, when the shopping list N is empty. The MinMin algorithm is polynomial with a time complexity of $O(mn^2)$.

One of the main issues with this heuristic is that it is prone to stagnation due to its deterministic nature. Moreover, its operation may lead to obtaining locally unoptimal solutions since there is an additional flat delivery cost that is connected with each shop, and this factor should be noticed as one of the main differences between the MinMin algorithm and the greedy one, which was used for comparison with the presented new methods.

A potential strategy to lessen this issue consists of applying one or more local search methods, such as the

Algorithm 1. MinMin algorithm.

Generates a solution by using the MinMin algorithm.

Input: N_i: Multiset of available products per shop.
N: Shopping list.
M: List of shops.
Output: $X = (X_1, \dots, X_m)$: Sequence of selection of products.

$\{F$: sum of product costs and delivery costs$\}$

```
1: $X = (X_1 = \emptyset, \dots, X_m = \emptyset)$
2: while $N \neq \emptyset$ do
3:    $min = \infty$
4:    for all $i \in M$ do
5:       for all $j \in N$ do
6:          assignProduct $j$ to $X_i$
7:          if $F(X) < min$ then
8:             $min = F(X)$
9:             $j' = j$
10:         end if
11:         deleteProduct $j$ from $X_i$
12:      end for
13:   end for
14:   assignProduct $j'$ inShop $i$ to $X_i$
15:   deleteProduct $j'$ from $N$ and product $j'$ from $N_i$
16: end while
17: return $X$
```

one shown in Algorithm 2. This algorithm starts by creating a vector of pair values that contains products j as assigned in the sequence of selection of products X_{old} with its $cost$, which includes c_{ij} and d_i from the multiset N_i.

The vector V is sorted in ascending order. Then, the sequence X_{old} is assigned to the sequence X and its current objective value is stored in the variable $oldTotalCost$.

Then, we check, for each product j in V, starting with the more expensive products, from which shop we could buy it to reduce the total cost of the selection of products X.

When the search finishes for that product, it is moved to a new shop, which might or might not be different from the current shop. This process continues until all the products j in V have been reallocated.

Additionally, the local search algorithm can be applied to partial states of the solution generated by the MinMin algorithm to guide the process to a better region in the search space. The local search algorithm is polynomial with a time complexity of $O(n^2)$.

In the experimentation, the three versions of the MinMin algorithm were compared to determine the quality of the solutions and the consumption of time associated to each version: the original

Algorithm 2. Local search algorithm.

```
Improves a solution to its local optimum in the search space.
Input: $X_{old} = (X_{old_1}, \dots, X_{old_m})$: Sequence of selection of products to improve.
    $N_i$: Multiset of available products per shop.
Output: $X = (X_1, \dots, X_m)$: Improved sequence of selection of products.

{$F$: sum of product costs and delivery costs}
1: add products $j$ and $cost = c_{ij} + d_i$ from $X_{old_i}$ to $V$
2: sort $V$ by $cost$ in ascending order
3: $X = X_{old}$
4: $oldTotalCost = F(X)$
5: for all $j \in V$ do
6:    deleteProduct $j$ from $X_i$
7:    for all $i \in M$ do
8:       assignProduct $j$ to $X_i$
9:       if $F(X) < oldTotalCost$ then
10:         $oldTotalCost = F(X)$
11:         $i' = i$
12:      end if
13:      deleteProduct $j$ from $X_i$
14:   end for
15:   assignProduct $j$ inShop $i'$ to $X_i$
16: end for
17: return $X$
```

MinMin, the MinMin+Local Search algorithm and the MinMin+MidLocal Search algorithm.

The second variation of this method is presented in Algorithm 3. This algorithm starts by generating the solution with the MinMin method (Algorithm 1) to immediately improve it to its local optimum, using the local search method described in Algorithm 2.

The MinMin+Local Search algorithm is polynomial with a time complexity of $O(mn^2)$. The time complexity of the local search $O(n^2)$, is omitted, because it falls within the order of the MinMin algorithm.

Lastly, the third variation, shown in Algorithm 4, applies a local search algorithm to the partial solutions constructed by the MinMin algorithm to improve them and guide them to a better region in the search space. The moment when the local search method is applied is controlled by the variable α, which depends on the size of the instance being solved. The MinMin+MidLocal Search algorithm is polynomial with a time complexity of $O(mn^4)$ in the worst case, where the local search algorithm is applied every time a new shop is added to the partial solution.

Algorithm 3. MinMin+Local Search algorithm.

```
Generates an initial solution using the MinMin algorithm that is improved by a Local Search method.
Input: $N_i$: Multiset of available products per shop.
    $N$: Shopping list.
    $M$: List of shops.
Output: $X = (X_1, \dots, X_m)$: Sequence of selection of products

{$minMin$: MinMin heuristic in Algorithm 1}
{$localSearch$: Local search in Algorithm 2}
1: $X = minMin(N_i, N, M)$
2: $X = localSearch(X, N_i)$
3: return $X$
```

5.2. Cellular processing algorithm.

The cellular computing approach is described by Sipper (1999) as an algorithm design philosophy that is based on three interrelated principles. The principle of simplicity states that a processing cell ideally performs very simple tasks that consume little time. Next, the parallelism principle deploys lots of individual cells in order to solve a single task. The third is the locality principle, which states that, given the high number of cells, communication between all of them is impractical, therefore local communications between neighboring cells is preferred.

This paradigm is especially suitable for use in cloud computing or data centers, as it takes advantage of the large number of processors available.

The ISOP is found naturally in these kinds of environments, where central servers with several computing units are required to attend to the petitions of various users around the world in the minimum period of time. This is the main reason why a cellular processing algorithm that follows the cellular computing philosophy was considered.

Cellular processing algorithms were proposed by Terán-Villanueva *et al.* (2015), initially to solve, with promising results, the linear ordering problem with cumulative costs, a combinatorial problem.

Often compared with the hyper-heuristic approach (Burke *et al.*, 2003), this differs from our cellular processing approach, since each processing cell has complete knowledge of the problem that is being solved. In contrast, the hyper-heuristic approach has a domain barrier between the controller and the low level heuristics. The second main difference is the fact that the processing cells are adapted and matched to the problem that is being solved, while the hyper-heuristic approach depends on generic low level heuristics that can be applied to a greater variety of optimization problems, according to Burke *et al.* (2003).

The cellular processing algorithm relies on independent processing cells that maintain their own population and processes them with their own mechanism.

Algorithm 4. MinMin+MidLocal Search algorithm.

Generates a solution by using the MinMin algorithm that is improved by a local search method during the construction of the solution.

```
Input: $N_i$: Multiset of available products per shop.
    $N$: Shopping list.
    $M$: List of shops.
Output: $X = (X_1, \ldots, X_m)$: Sequence of selection of products.

    {$localSearch$: Local search function Algorithm 2}
    {$F$: sum of product costs and delivery costs}
    {$updateAlpha$: determines next iteration in which $localSearch$ will be applied}

1: $X = (X_1 = \emptyset, \ldots, X_m = \emptyset)$
2: while $N \neq \emptyset$ do
3:   $min = \infty$
4:   for all $i \in M$ do
5:     for all $j \in N$ do
6:       assignProduct $j$ to $X_i$
7:       if $F(X) < min$ then
8:         $min = F(X)$
9:         $j' = j$
10:      end if
11:      deleteProduct $j$ from $X_i$
12:    end for
13:  end for
14:  assignProduct $j'$ inShop $i$ to $X_i$
15:  deleteProduct $j'$ from $N$ and product $j'$ from $N_i$
16:  if $currentIteration == \alpha$ then
17:    $X = localSearch(X, N_i)$
18:    $updateAlpha(\alpha)$
19:  end if
20: end while
21: return $X$
```

The components of each cell are shown in Fig. 1: the Pool contains a candidate solution or a set of them that will be modified by the Processing component.

The Processing component is a method or heuristic that solves and improves the candidate solutions in the Pool. The method that is included in the Processing core can be the same with minor variations or completely different across the cellular processing algorithm.

The Cell Controller component is in control of the stagnation of the Pool, being able to take corrective measures such as generating a new set of candidate solutions in the Pool, moving the existing ones to another region of the search space, or stopping its own execution to avoid the misuse of computational resources such as computational processing time and memory space.

To improve the execution of each cell, the cells communicate with other nearby cells during their execution or after the processing cells have evaluated all their individuals in their respective Pools. This process is illustrated in Fig. 2.

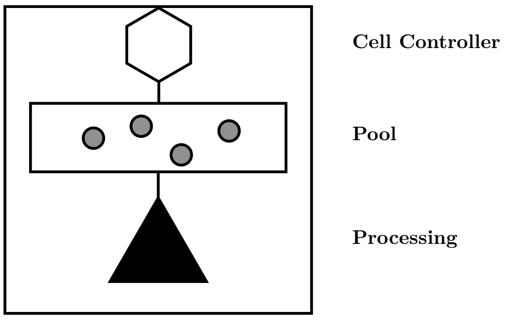

Fig. 1. Representation and components of an individual processing cell.

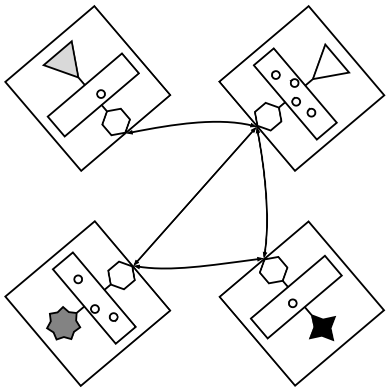

Fig. 2. Communication between processing cells.

The whole process usually finishes when most of the cells converge to a local optimum that might or might not be the optimal solution for the instance being solved.

In this particular implementation of a cellular processing algorithm, applied to the ISOP, the number of processing cells was limited to five and the stopping criterion was five iterations without improvement of the best solution. This configuration was found by decreasing the values of the parameter set, initially set to high values, to the point where the quality of the solutions obtained by the algorithm on the medium-sized set was not affected by the new established values.

The algorithm starts by constructing an initial solution with the MinMin method for each cell, which is then improved by applying a local search algorithm.

The implementation of all the cells in the cellular processing algorithm as a processing unit of iterative local search is presented in Algorithm 5.

The following iterative local search algorithm parameters were established: $maxNoImprov$ that controls the maximum number of iterations without improvement was set to 10 iterations, the variable

$percentage$ that controls the percentage of the solution being randomly rearranged was set to 0.5. The local search algorithm finds the best position for every element in the solution, and, if no other position improves the objective value, the element is left in its original place.

Algorithm 5. Iterative local search.

```
Improves a given solution by reallocating all the elements in the sequence to a better position, until a local optimum value is found.
Input: $N_i$: Multiset of available products per shop.
Output: $X = (X_1, \ldots, X_m)$: Sequence of selection of products.
{$perturbSolution$: Randomly reallocates a percentage of the elements in $X$}
{$localSearch$: Local search function Algorithm 2}
{$F$: sum of product and delivery costs}
1: $bestX = X$
2: while $noImprov < maxNoImprov$ do
3:   $perturbSolution(X, percentage)$
4:   $X = localSearch(X, N_i)$
5:   if $X < bestX$ then
6:     $bestX = X$
7:     $noImprov = 0$
8:   else
9:     $noImprov++$
10:  end if
11: end while
12: $X = localSearch(bestX, N_i)$
13: return $bestX$
```

The communication in this case is performed by sharing all the solutions with all cells after each iteration is completed.

6. Computational experimentation

The experimentation will consider the evaluation of the three newly proposed solution methods: ILP, MinMin and its variants (Algorithms 3 and 4), and the cellular processing algorithm.

To generate instance cases of the ISOP, a realistic model was created. We studied the relationship between the competitive structure, advertising, price, and the price dispersion over Internet shops. As a group of representative products to be taken into account in our computational experiment, we chose books, because of their wide choice in virtual (Internet) shops and the frequency of purchase through this kind of shopping channel.

We adopted for our model some information and computational results reported by Pathak (2012), Ratchford *et al.* (2003), Pan *et al.* (2003) or Clay *et al.* (2001). We focused mainly on a model definition of electronic bookstores (concentrated mostly on books, CDs with music, DVDs with movies), prices, the acceptance factor, the retailer brand (Chu *et al.*, 2005), and, which is important for the definition of the model of the optimization problem, price dispersion. One should also notice that consumers may choose from a large number of Internet bookstores. The data for our sample were collected from 32 shops and covered the largest shops based in the United States, including Amazon, `BarnesandNoble.com`, `Borders.com`, `Buy.com`, and Booksamillion, as well as the top sellers among Internet bookstores in Poland, such as `empik.com` and `merlin.pl`. We decided to review and upgrade the model presented by Blazewicz *et al.* (2010). Our goal was to create a new, more sophisticated (focused not only on books), and even more realistic model than the previous one. The new model was used in parallel by Blazewicz *et al.* (2016).

The working model was prepared on the basis of data from the above-mentioned publications, as well as our own observations of many Internet shops. It is assumed that each shop has all the required books. In each instance, the following values are randomly generated for all i and j in the corresponding ranges. *Reference price* (ref) of a product j: $ref_j \in \{2, 4, \ldots, 100\}$ with a percentage of occurrence: 40% between 0 and 20, 16% between 22 and 30, 12% between 32 and 40, 16% between 42 and 60 and 16% between 62 and 100, respectively. The price of product j from shop i: $p_{ij} \in [a_{ij}, b_{ij}]$, where $a_{ij} \geq 0.75ref_j$, $b_{ij} \leq 1.36ref_j$, and the structure of intervals between $[a_{ij}, b_{ij}]$ is as follows:

$$\begin{array}{ll}
 & minimum \\
[8\%] & \\
 & minimum + \frac{(ref - minimum)}{4} \\
[3\%] & \\
 & minimum + \frac{(ref - minimum)}{2} \\
[9\%] & \\
 & minimum + \frac{(ref - minimum)}{1.25} \\
[21\%] & \\
 & ref \\
[24\%] & \\
 & ref + \frac{(maximum - ref)}{4} \\
[9\%] & \\
 & ref + \frac{(maximum - ref)}{2} \\
[10\%] & \\
 & ref + \frac{(maximum - ref)}{1.25} \\
[16\%] & \\
 & maximum
\end{array}$$

Every shop is connected with a delivery fee, taken arbitrarily between 0 and 20.

With the previously described method of generating instances, we created three sets of different sizes to perform the computational experimentations that will be presented next; a Small set, with three subsets of 3n20m, 4n20m and 5n20m, each one with 30 cases; a Medium set, with three subsets: 5n240m, 5n400m and 50n240m, each one with 20 cases, a Large set with three subsets: 50n400m, 100n240m and 100n400m, each one also with 20 instances of the problem.

To determine the correctness of ILP, we solved the instances from the Small set with an enumerative method and with ILP, and compared the results obtained by both methods to verify whether the two reached the same solution.

For the second test, the Medium and Large sets were solved with ILP without a time limit to obtain the optimum values in all the cases. Then, with the known optimal values, we compared the performance of the proposed solution methods, MinMin+Local Search and the cellular processing algorithm with the heuristics proposed by Blazewicz *et al.* (2010), greedy and forecasting.

The last test was performed on an instance consisting of real data obtained by a web crawler to investigate the performance of this method in a real case scenario. The recollected data set consisted of 400 distinct products from 57 shops offering in all 6387 tenders, since not every shop could offer all of the products.

ILP was implemented in the Java programming language, version 1.7.0_45, using the API provided by the CPLEX optimization suite, 12.1, developed by IBM.

The greedy, forecasting, cellular processing and MinMin algorithms were developed using the PHP programming language, seeing that it was originally proposed in the literature as a core coding language for testbed web-site applications.

The specification of the hardware on which the computational experimentations were conducted includes an Intel i5 processor at 2.3 GHz. with 8 GB of DD3 RAM at 1333 GHz and an SSD drive of 256 GB.

The instance sets used in this experimentation will be available via the Internet Shopping Optimization Project web page.[1]

7. Results

In this section, we will present the results obtained in the computational experiments previously described.

In the case of ILP, we selected the family sets of products 5n, 50n, and 100n from the Small, Medium and Large sets. This family was used to measure the time required by ILP to solve instances of different sizes.

To observe the behavior of ILP we present the results in the notched box plot shown in Fig. 3. The box plot divides all the observations into quartiles, where the whiskers represent the minimum and maximum observations, excluding outliers.

[1]`http://www.cs.put.poznan.pl/ishop/`.

The box represents the central quartiles, i.e., 50% of the occurrence of the observations, where the size of the box indicates how spread the data are in that range. The median is depicted by the line drawn inside the box, taking into account the observations between the whiskers.

The notches in the boxes were added to indicate that the medians of the subsets are statistically different from each other. When the differences are not significant, those overlap with the notches of the other subsets.

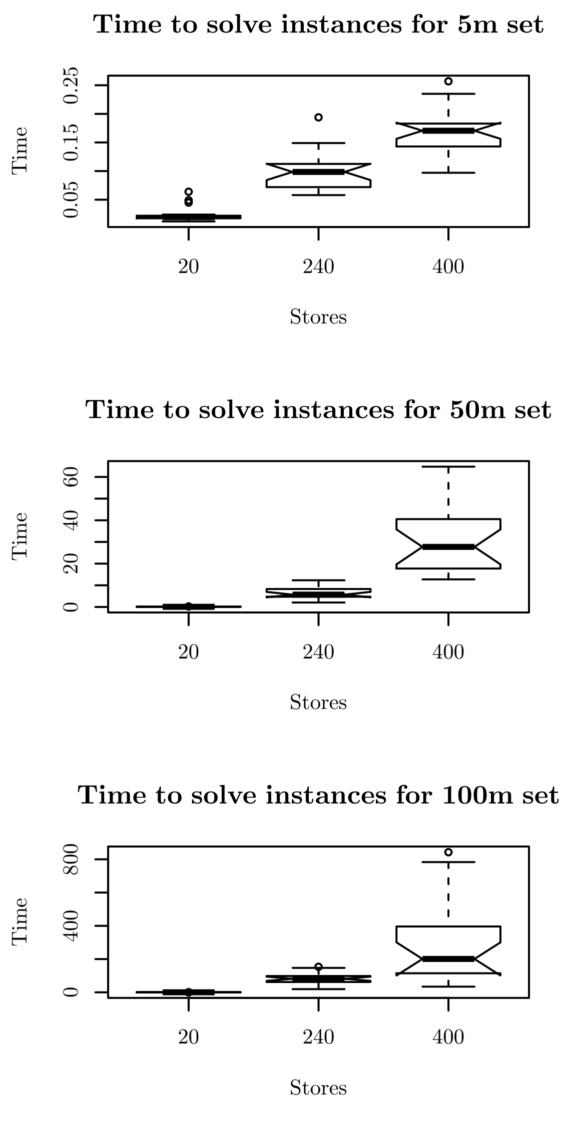

Fig. 3. Set of instances with 5 products.

For the set 5n, the median time used by ILP to solve each instance is close to 0.25 seconds. For the 50n set, the time increases by as much as one minute in the cases of the instances of the subset 50n400m, with the median time close to 30 seconds and for the subset 50n240m less than 10 seconds. In the case of the 100n set, the maximum

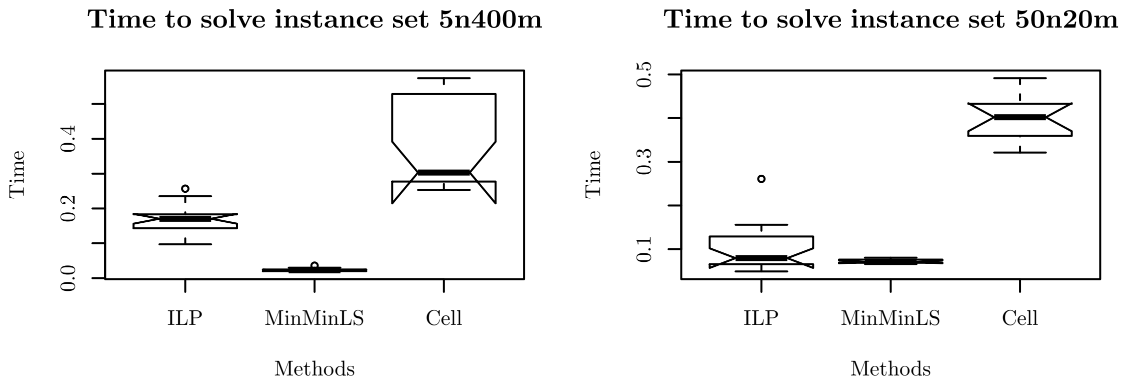

Fig. 4. Time comparison of the ILP, MinMin+Local Search and cellular methods.

time used to solve an instance was 800 seconds, from the 100n400m subset, while for most of the instances, the time needed is close to 200 seconds.

The test with data recollected from online shops under real conditions, such as prices, delivery costs, and the differing inventories of each shop, shows solid performance of the ILP model since an optimal solution for the instance of 400 products and 52 shops was obtained in 1.05 s.

The usage of ILP in solving instances of the ISOP in real case scenarios is presented in Fig. 4. Here one can observe the instance sets that are solved by ILP, MinMin+Local Search and cellular in less than one second. Moreover, it is noticeable that ILP is able to match, or outperform, the time of the two other main methods proposed.

For the bigger sets, the use of heuristics is preferred, given that the time increase factor of ILP is noticeably higher than the one in the heuristics. Therefore, the Cellular and MinMin+Local Search methods are more suitable to solve bigger instances with less computational resources. This last remark is critical, given the context in which the ISOP is found.

Once the correctness of the proposed ILP model was established, and a benchmark with optimal values from the Medium and Large sets obtained, we proceeded to evaluate the performance of the newly proposed heuristics along with the ones described by Blazewicz *et al.* (2010).

To achieve this, the four proposed methods (MinMin, MinMin+MidLocal Search algorithm, MinMin+Local Search algorithm, cellular processing algorithm) were put to solve the Medium and Large sets, until reaching their own stop conditions. The results obtained are shown in Fig. 5 in the case of the Medium set and in Fig. 6 for the Large set. In the figures, the x axis represents an instance of the set, and on the y axis, the objective value on the left, and the computational time on the right side of the plot. As the ISOP is a minimization problem, lower values in the objective value, just as in the case of time, are considered better.

In the Medium set, shown in Fig. 5, all heuristics return similar solutions in terms of the objective value for the subsets 5n240m and 5n400m, but the cellular processing algorithm is slower than the heuristics based on the MinMin algorithm.

In the instance sets of 50 and 100 products, we start to see that the cellular heuristic gets better quality solutions than the heuristics based on MinMin, but also at the expense of the computational time used.

In the case of the heuristics based on MinMin, the versions that included the local search algorithm in their process also obtained better solutions: they were more consistent in terms of the quality of the solutions obtained when compared with the basic MinMin. However, there were no significant differences that would justify including the local search algorithm in the construction process, or as a final step, to the already constructed solution. The most significant difference appeared in the time needed by each heuristic to solve the instances. MinMin+Local Search needed as much as one-half of the time of MinMin+MidLocal Search.

With those results, we selected the proposed heuristics, MinMin+Local Search and the cellular processing algorithm, to compare them in terms of the quality of the solution with the one described by Blazewicz *et al.* (2010).

The results of this experiment are shown in Fig. 7. In this graph, the x axis represents an instance from the Medium or Large sets, while the y axis indicates the percentage error regarding the optimal value, represented by the dashed line. Values closer to 0 are better.

We can observe, in the case of the subsets 5n240m and 5n400m from the Medium set, that all heuristics are able to achieve optimal values in most cases, but the

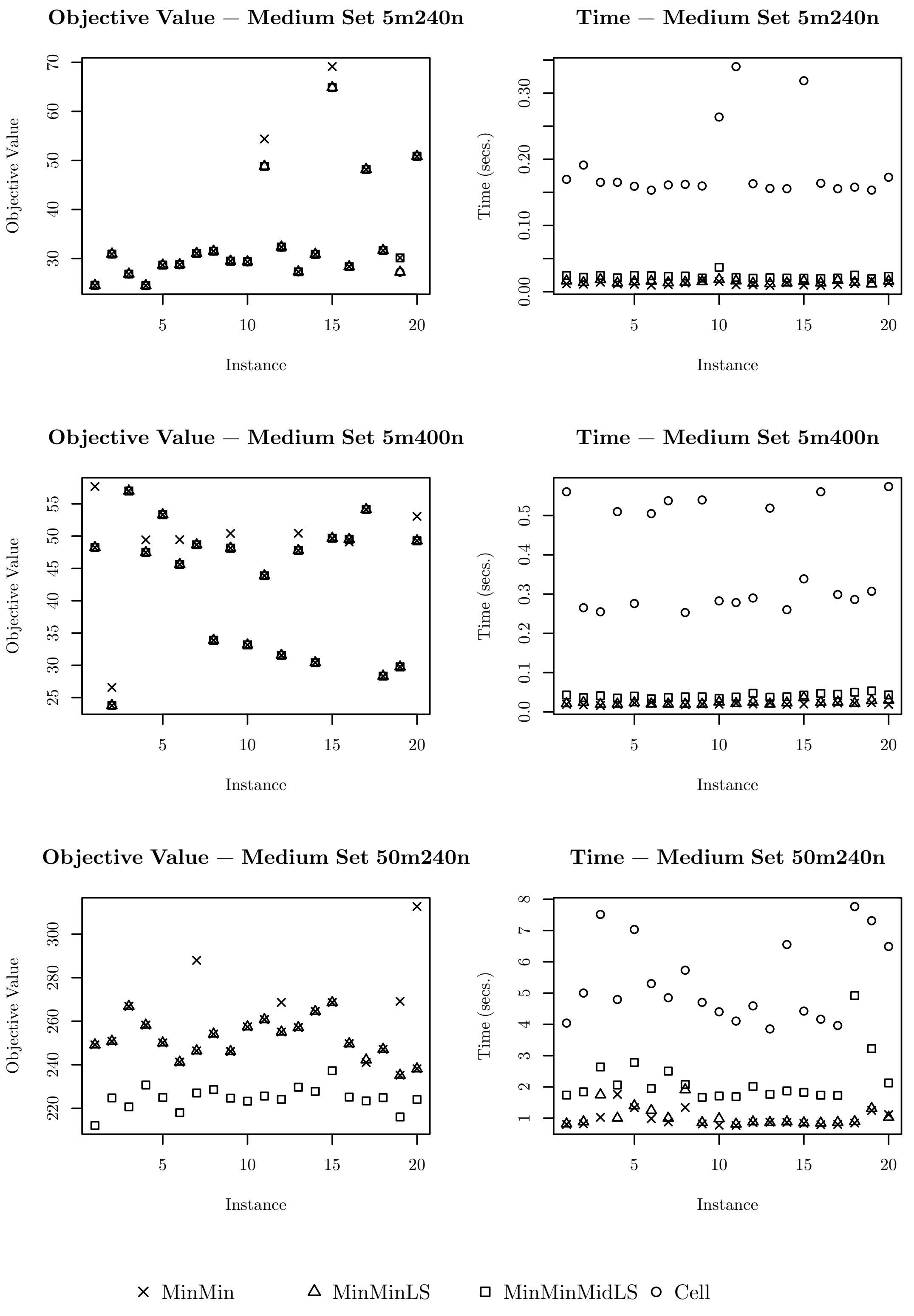

Fig. 5. Results obtained by the heuristics on Medium set instances.

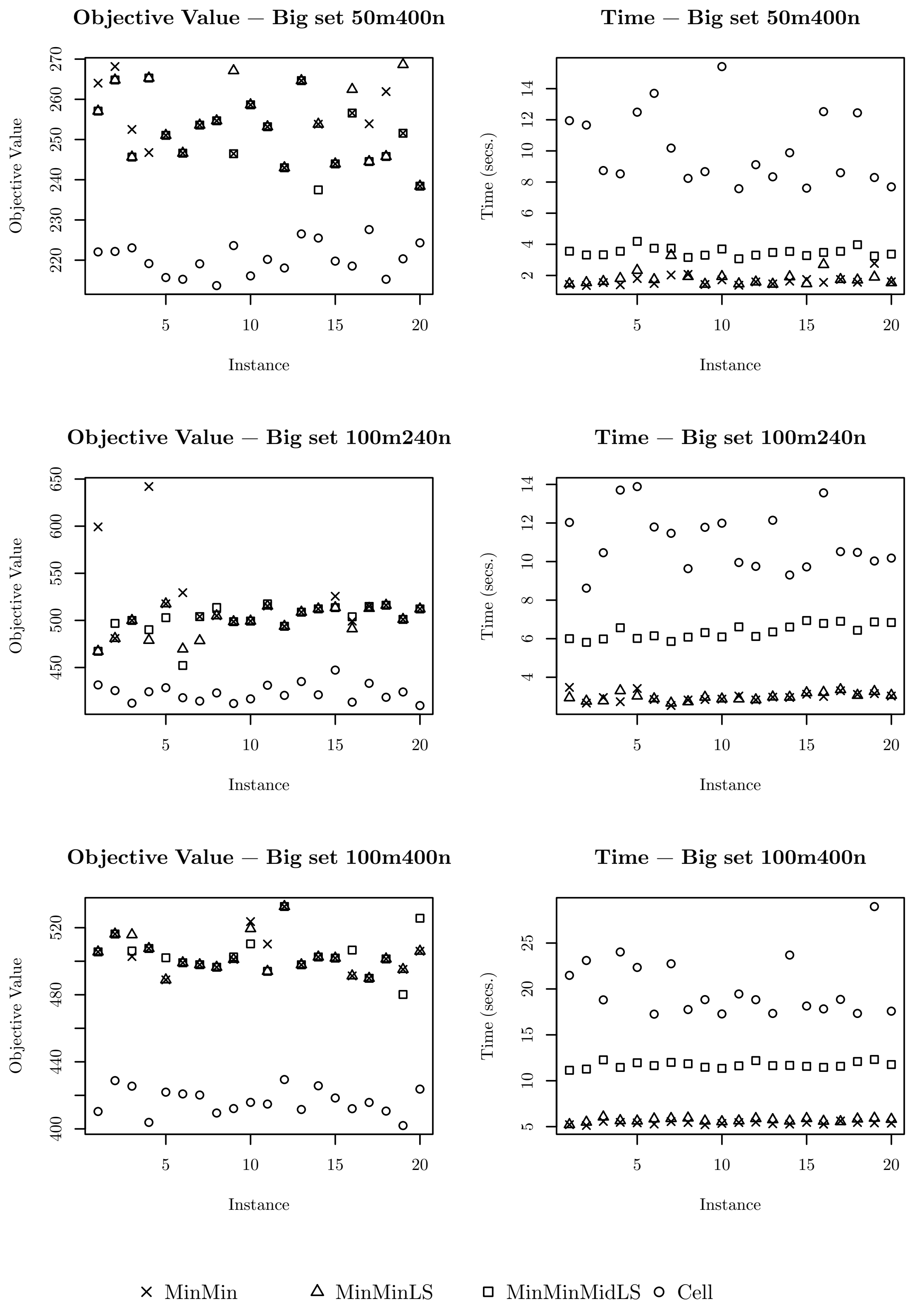

Fig. 6. Results obtained by the heuristics on Large set instances.

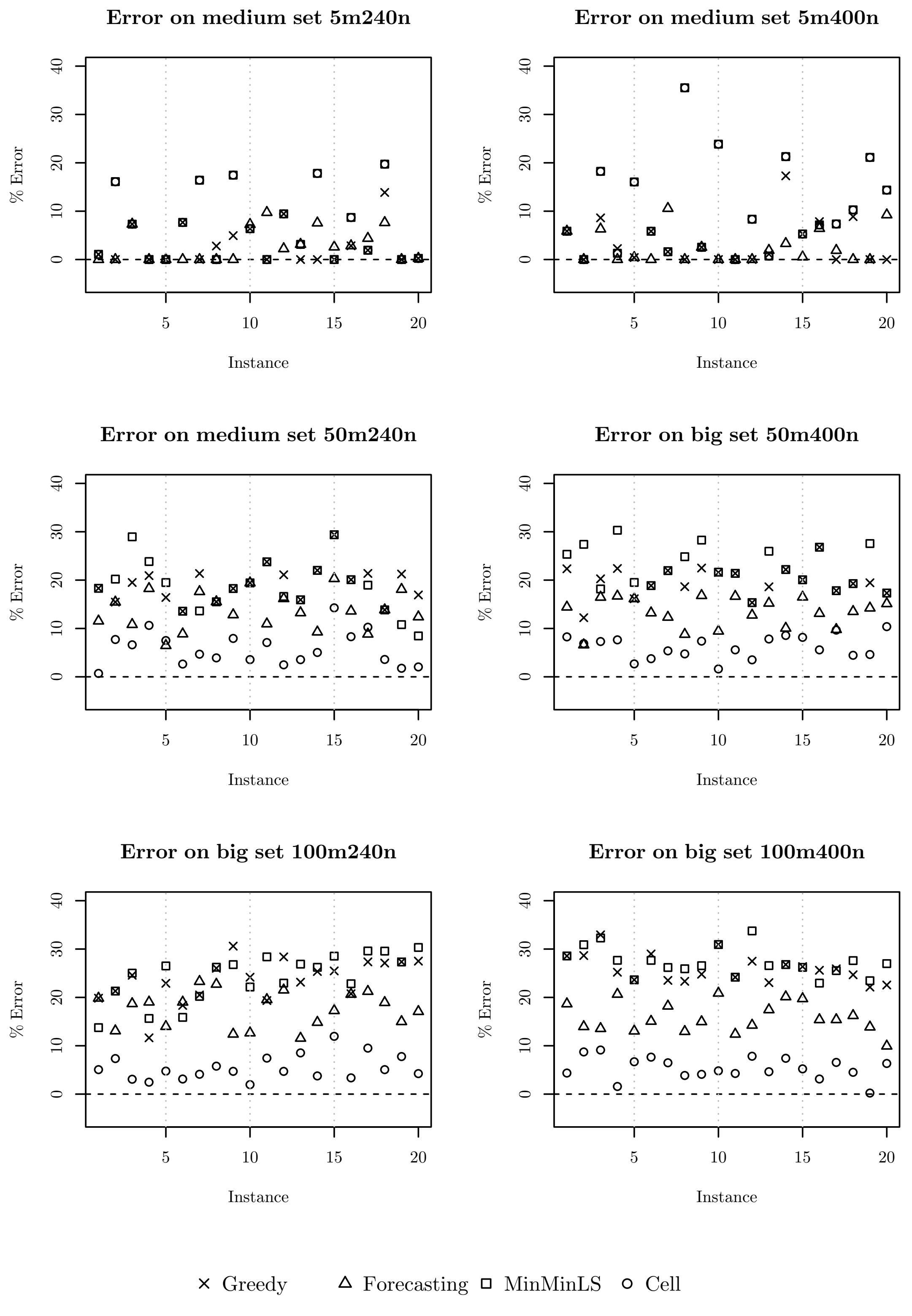

Fig. 7. Percentage of error of the heuristics from the optimum value.

forecasting and cellular heuristics are the ones with the least amount of errors compared with the heuristics from the literature.

For the instances from the Large set, we observe that the gap between the heuristics increases with the size of the instance. Moreover, the heuristic that obtains the best solutions in general is the cellular processing algorithm, followed by the forecasting heuristic. It is also remarkable that the deviation from the optimal value of the cellular processing algorithm falls consistently in the range from zero to ten percent, independently of the size of the instance set being solved.

8. Conclusions

This article presented an ILP model applied to the ISOP that optimally solves generated instances of small and medium size. Additionally, the MinMin algorithm with two variants and a cellular processing algorithm which solves bigger instances in an approximate way were proposed.

Exact solution methods that are adequate to deliver optimal values for small instances of the ISOP can be useful to test and evaluate the performance of the proposed and future heuristics as well as approximation algorithms.

The proposed ILP is able to solve optimally the instances of the ISOP generated with the algorithm described by Blazewicz *et al.* (2010) with the sizes up to 400 shops and 5 products, which can be reasonable for most users who employ price comparison sites.

In the case of the instance that was created by recollecting data from real shops, which considered the search of a shopping list of 57 items among 400 shops, with some of the products unavailable from some shops, the ILP model was able to obtain the optimal solution in 1.05 seconds. The performance of the algorithm improves in real case scenarios when not all products can be purchased from all shops.

In addition to the previously described ILP model, we proposed four new heuristics, one based on the concept of cellular computing, described by Sipper (1999), and the other three based on the MinMin algorithm that is found in the area of allocation of resources and grid computing.

Taking into account the results of the experimentation, the cellular processing algorithm is the one with the best performance, including those previously proposed in the literature. The measures also showed that the time needed by the cell algorithm increases in a linear fashion, in contrast to other methods, whose computational complexity is bigger. This means that, for small instances, the algorithms from the state of the art have better performance than the cell algorithm, but as the instances increase in size, this tendency is reversed.

Meanwhile, the algorithm MinMin+Local Search, based on the MinMin algorithm, shows performance similar to that of the state-of-the-art heuristics, but with a better in terms of time, since it also presents the linear complexity as the instance grows in size.

In the case of larger instances, where the ILP model is unable to offer a solution in a reasonable amount of time, the cellular processing algorithm presents solutions with less than 10% of errors, in a fraction of the time used by the exact method.

For instances with a larger number of products, in the cases such as buying items from online supermarkets and grocery shops, the ILP model can be useful to determine the gap between the optimal solution for medium size instances and the objective values calculated by the heuristics proposed by Blazewicz *et al.* (2010).

In the future, we plan to extend the Internet shopping model to include additional constraints such as minimum delivery time (using the experience in modelling delivery systems in flexible manufacturing systems will be helpful (Blazewicz *et al.*, 1994)) and incomplete shopping list realization. The experimental results demonstrated the potential of the new cellular processing optimization algorithm. However, one of the main concerns for its applicability to the ISOP is the time needed to find a solution. In order to alleviate this problem and also deal with scalability, we will consider investigating a parallel version of the algorithm within the framework of the general purpose computing on graphics processing units (GPGPU).

It is worth noting that shopping optimization can be considered not only from the customers point of view, but also from the service providers point of view. To complete customer orders, Internet shop owners must solve many scheduling problems, similar to the ones met in production systems (cf., e.g., Sterna, 2007). Within future research we would like to study the problem of scheduling tasks corresponding to assembling, packing and shipping customer orders to minimize the late parts of those orders, which might be modeled as a job shop system (Blazewicz *et al.*, 2007).

Acknowledgment

This study was partially supported by the FNR (Luxembourg) and NCBiR (Poland), through the IShOP project, INTER/POLLUX/13/6466384. Moreover, the authors would like to thank the journal reviewers as well as Mateusz Guzek for their valuable comments and suggestions for improving the paper.

References

Blazewicz, J., Bouvry, P., Kovalyov, M.Y. and Musial, J. (2014a). Erratum to: Internet shopping with price-sensitive discounts, *4OR—A Quarterly Journal of Operations Research* **12**(4): 403–406.

Blazewicz, J., Bouvry, P., Kovalyov, M.Y. and Musial, J. (2014b). Internet shopping with price sensitive discounts, *4OR—A Quarterly Journal of Operations Research* **12**(1): 35–48.

Blazewicz, J., Burkard, R., Finke, G. and Woeginger, G. (1994). Vehicle scheduling in two-cycle flexible manufacturing systems, *Mathematical and Computer Modelling* **20**(2): 19–31.

Blazewicz, J., Cheriere, N., Dutot, P.-F., Musial, J. and Trystram, D. (2016). Novel dual discounting functions for the Internet shopping optimization problem: New algorithms, *Journal of Scheduling* **19**(3): 245–255.

Błażewicz, J., Kovalyov, M., Musiał, J., Urbański, A. and Wojciechowski, A. (2010). Internet shopping optimization problem, *International Journal of Applied Mathematics and Computer Science* **20**(2): 385–390, DOI: 10.2478/v10006-010-0028-0.

Blazewicz, J. and Musial, J. (2011). E-commerce evaluation—multi-item internet shopping. Optimization and heuristic algorithms, *in* B. Hu *et al.* (Eds.), *Operations Research Proceedings 2010*, Springer-Verlag, Berlin, pp. 149–154.

Blazewicz, J., Pesch, E., Sterna, M. and Werner, F. (2007). A note on the two machine job shop with the weighted late work criterion, *Journal of Scheduling* **10**(2): 87–95.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P. and Schulenburg, S. (2003). Hyper-heuristics: An emerging direction in modern search technology, *in* F. Glover and G.A. Kochenberger (Eds.), *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, Vol. 57, Springer-Verlag, Berlin, pp. 457–474.

Cheung, C.M., Chan, G.W. and Limayem, M. (2005). A critical review of online consumer behavior: Empirical research, *Journal of Electronic Commerce in Organizations* **3**(4): 1–19.

Chu, W., Choi, B. and Song, M. (2005). The role of on-line retailer brand and infomediary reputation in increasing consumer purchase intention, *International Journal of Electronic Commerce* **9**(3): 115–127.

Clay, K., Krishnan, R. and Wolff, E. (2001). Prices and price dispersion on the web: Evidence from the online book industry, *Journal of Industrial Economics* **49**(4): 521–539.

Diaz, C.O., Pecero, J.E. and Bouvry, P. (2014). Scalable, low complexity, and fast greedy scheduling heuristics for highly heterogeneous distributed computing systems, *The Journal of Supercomputing* **67**(3): 837–853.

Eiselt, H. and Sandblom, C.-L. (2004). *Decision Analysis, Location Models, and Scheduling Problems*, Springer-Verlag, Berlin.

Freund, R., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., Kidd, T., Kussow, M., Lima, J., Mirabile, F., Moore, L., Rust, B. and Siegel, H. (1998). Scheduling resources in multi-user, heterogeneous, computing environments with smartnet, *Proceedings of the 7th Heterogeneous Computing Workshop, Orlando, FL, USA*, pp. 184–199.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman & Co., San Francisco, CA.

Goossens, D.R., Maas, A., Spieksma, F.C. and Van de Klundert, J. (2007). Exact algorithms for procurement problems under a total quantity discount structure, *European Journal of Operational Research* **178**(2): 603–626.

Guzek, M., Gniewek, A., Bouvry, P., Musial, J. and Blazewicz, J. (2015). Cloud brokering: Current practices and upcoming challenges, *IEEE Cloud Computing* **2**(2): 40–47.

Iyigun, C. and Ben-Israel, A. (2010). A generalized Weiszfeld method for the multi-facility location problem, *Operations Research Letters* **38**(3): 207–214.

Krarup, J., Pisinger, D. and Plastria, F. (2002). Discrete location problems with push–pull objectives, *Discrete Applied Mathematics* **123**(1): 363–378.

Krichen, S., Laabidi, A. and Abdelaziz, F.B. (2011). Single supplier multiple cooperative retailers inventory model with quantity discount and permissible delay in payments, *Computers & Industrial Engineering* **60**(1): 164–172.

Marszalkowski, J., Marszalkowski, J.M. and Drozdowski, M. (2014). Empirical study of load time factor in search engine ranking, *Journal of Web Engineering* **13**(1&2): 114–128.

Marszalkowski, J. and Musial, J. (2011). Database scheme optimization for online applications, *Foundations of Computing and Decision Sciences* **36**(2): 121–129.

Melo, M.T., Nickel, S. and Saldanha da Gama, F. (2009). Facility location and supply chain management—A review, *European Journal of Operational Research* **196**(2): 401–412.

Mirmohammadi, S.H., Shadrokh, S. and Kianfar, F. (2009). An efficient optimal algorithm for the quantity discount problem in material requirement planning, *Computers & Operations Research* **36**(6): 1780–1788.

Munson, C. and Hu, J. (2010). Incorporating quantity discounts and their inventory impacts into the centralized purchasing decision, *European Journal of Operational Research* **201**(2): 581–592.

Musial, J. (2012). *Applications of Combinatorial Optimization for Online Shopping*, NAKOM, Poznań.

Nesmachnow, S., Dorronsoro, B., Pecero, J.E. and Bouvry, P. (2013). Energy-aware scheduling on multicore heterogeneous grid computing systems, *Journal of Grid Computing* **11**(4): 653–680.

Pan, X., Ratchford, B. and Shankar, V. (2003). The evolution of price dispersion in internet retail markets, *in* M.R. Baye (Ed.), *Organizing the New Industrial Economy*, Advances in Applied Microeconomics, Vol. 12, Emerald Group Publishing, Bingley, pp. 85–105.

Pathak, B. (2012). Comparison shopping agents and online price dispersion: A search cost based explanation, *Journal of Theoretical and Applied Electronic Commerce Research* **7**(1): 64–76.

Ratchford, B., Pan, X. and Shankar, V. (2003). On the efficiency of internet markets for consumer goods, *Journal of Public Policy & Marketing* **22**(1): 4–16.

Revelle, C., Eiselt, H. and Daskin, M. (2008). A bibliography for some fundamental problem categories in discrete location science, *European Journal of Operational Research* **184**(3): 817–848.

Rose, S. and Dhandayudham, A. (2014). Towards an understanding of Internet-based problem shopping behaviour: The concept of online shopping addiction and its proposed predictors, *Journal of Behavioral Addictions* **3**(2): 83–89.

Rose, S. and Samouel, P. (2009). Internal psychological versus external market-driven determinants of the amount of consumer information search amongst online shoppers, *Journal of Marketing Management* **25**(1–2): 171–190.

Sawik, B. (2012). Downside risk approach for multi-objective portfolio optimization, *in* D. Klatte *et al.* (Eds.), *Operations Research Proceedings 2011*, Springer-Verlag, Berlin, pp. 191–196.

Simpson, T. (1750). *The Doctrine and Application of Fluxions*, Nourse, London.

Sipper, M. (1999). The emergence of cellular computing, *Computer* **32**(7): 18–26.

Sterna, M. (2007). Late work minimization in a small flexible manufacturing system, *Computers & Industrial Engineering* **52**(2): 210–228.

Terán-Villanueva, J.D., Huacuja, H.J.F., Valadez, J.M.C., Rangel, R.P., Soberanes, H.J.P. and Flores, J.A.M. (2015). A heterogeneous cellular processing algorithm for minimizing the power consumption in wireless communications systems, *Computational Optimization and Applications* **62**(3): 787–814.

Weber, A. (1929). *The Theory of the Location of Industries*, Chicago University Press, Chicago.

Wojciechowski, A. and Musial, J. (2009). A customer assistance system: Optimizing basket cost, *Foundations of Computing and Decision Sciences* **34**(1): 59–69.

Wojciechowski, A. and Musial, J. (2010). Towards optimal multi-item shopping basket management: Heuristic approach, *in* R. Meersman *et al.* (Eds.), *On the Move to Meaningful Internet Systems: OTM 2010 Workshops*, Lecture Notes in Computer Science, Vol. 6428, Springer-Verlag, Berlin, pp. 349–357.

Mario C. Lopez-Loces (M.Sc. in computer science from Instituto Tecnolgico de Ciudad Madero, 2012) is a Ph.D. student of computer science and a researcher at Instituto Tecnolgico de Ciudad Madero. His research interests include algorithm design, combinatorial optimization, linear programming and fuzzy logic.

Jedrzej Musial is an assistant professor at the Poznań University of Technology. His research interests include e-commerce, Internet shopping, cloud brokering, algorithms, applications of combinatorial optimization and operations research. He has a Ph.D. in computer science from the Poznań University of Technology and from the University of Luxembourg. He has published over a dozen papers and a monograph. Follow his website for more details at `http://www.cs.put.poznan.pl/jmusial/`.

Johnatan E. Pecero obtained his Ph.D. in computer science at the Grenoble Institute of Technology (INPG, France). He is currently the head of the Standardization Department at GIE ANEC, working on ICT technical standardization, mainly cloud computing and data centers. Before joining GIE ANEC, Dr. Pecero held a postdoctoral research position at the University of Luxembourg, with the main research focus on cloud computing, scheduling and optimization.

Hector J. Fraire-Huacuja received the Ph.D. degree in computer science from Centro Nacional de Investigacion y Desarrollo Tecnologico Cuernavaca (Mexico) in 2005. His scientific interests include heuristic optimization and machine learning.

Jacek Blazewicz is a professor at the Poznań University of Technology. His research interests include algorithm design, computational complexity, scheduling, combinatorial optimization, bioinformatics, e-commerce. He has a Ph.D. in computer science from the Poznań University of Technology. His publication record includes over 340 papers in many outstanding journals. He is also the author and a co-author of over ten monographs. His science citation index is 3650 and the h-index 27. He is an IEEE fellow.

Pascal Bouvry is a professor in the computer science and communication research unit of the Faculty of Science, Technology and Communication at the University of Luxembourg, and a faculty member at the Luxembourg Interdisciplinary Center of Security, Reliability, and Trust. His research interest include cloud and parallel computing, optimization, security and reliability. He has a Ph.D. in computer science from the University of Grenoble (INPG), France. He is on the *IEEE Cloud Computing* editorial board. His full biography is available at `http://pascal.bouvry.org`.

Received: 21 March 2015
Revised: 28 July 2015
Re-revised: 5 August 2015
Accepted: 20 December 2015