BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 64, No. 3, 2016
DOI: 10.1515/bpasts-2016-0056

Algorithms solving the Internet shopping optimization problem with price discounts

J. MUSIAL[1]*, J.E. PECERO[2], M.C. LOPEZ-LOCES[3], H.J. FRAIRE-HUACUJA[3], P. BOUVRY[2], and J. BLAZEWICZ[1,4]

[1]Institute of Computing Science, Poznan University of Technology, 2 Piotrowo St., 60-965 Poznan, Poland
[2]Comp. Sci. and Commun. Res. Unit, University of Luxembourg, 6 rue Coudenhove Kalergi, L-1359 Luxembourg, Luxembourg
[3]Instituto Tecnológico de Ciudad Madero, Tecnológico Nacional de México, 1º de Mayo S/N, 89440, Ciudad Madero, Mexico
[4]Institute of Bioorganic Chemistry, Polish Academy of Sciences, 12/14 Noskowskiego St., 61-704 Poznan, Poland

Abstract. The Internet shopping optimization problem arises when a customer aims to purchase a list of goods from a set of web-stores with a minimum total cost. This problem is NP-hard in the strong sense. We are interested in solving the Internet shopping optimization problem with additional delivery costs associated to the web-stores where the goods are bought. It is of interest to extend the model including price discounts of goods.
The aim of this paper is to present a set of optimization algorithms to solve the problem. Our purpose is to find a compromise solution between computational time and results close to the optimum value. The performance of the set of algorithms is evaluated through simulations using real world data collected from 32 web-stores. The quality of the results provided by the set of algorithms is compared to the optimal solutions for small-size instances of the problem. The optimization algorithms are also evaluated regarding scalability when the size of the instances increases. The set of results revealed that the algorithms are able to compute good quality solutions close to the optimum in a reasonable time with very good scalability demonstrating their practicability.

Key words: e-commerce, Internet shopping, applications of operations research, approximations, algorithms, heuristics, combinatorial optimization.

1. Introduction

Internet shopping, fitting into a business-to-consumer subcategory, becomes more and more popular thanks to the facilities provided by the new information and communication technologies such as cloud computing, mobile computing (mobile devices – smartphones, tablets), the availability of data centers, and the improvement in the payment gateways on trusted computing platforms. Products available in web-stores are often cheaper than those offered by regular local retailers, and a wide choice of offers is available just a click away from the customer. Based on outstanding logistics, the delivery can usually be operated within 48 hours or less. Customers often need to take into account that shipping cost are charged, so that it is a good idea to group purchased products into sets and buy them from small number of retailers to minimize these delivery costs. Automating such decisions requires three elements: information about the product availability, price lists and finally a specialized analytical tool that could find the minimal subset of shops (in this work we use shops and web-stores interchangeably) where all products from the customer's shopping list could be bought at the lowest price. To automate these decisions one can prepare an algorithm / computer tool to calculate the best possible solution from the customers perspective. The issue could be perceived as an Internet optimization problem connected with online shopping. Partial solution to the most trivial version of the problem (just one product, no discounts, no additional costs, and so on) comes from software agents [1], so called price comparators. However their current functionality is limited to building a price rank for a single product among registered offers that fit to the customer's query (search phrase).

It is worth noting that beside all benefits of Internet shopping, it is still a type of shopping that may lead to problematic behaviors. One can notice that some customers may be addicted to shopping on the Internet. This kind of addiction is somehow similar to gaming and general Internet dependency. Many types of addictions and negative forms of consumptions connected with shopping in general have been widely researched and described in both medial and business journals. Business-Dictionary.com states that shopping is "the process of browsing and/or purchasing items in exchange for money". What will be important here that the platform is not so important. Therefore, Internet shopping experience may cause many of these negative behaviors. Follow the publication [2] for more information about different types of online shopping behaviors and addictions.

State-of-the-art research on the problem we will tackle in this paper consists of works that introduced Internet shopping optimization problem (ISOP in short) and more complicated versions of the mentioned problem, taking into account also price discounts. [3] introduced ISOP with the following idea. The addressed problem is to manage a multiple item shopping list over several shopping locations. The objective is to have all

*e-mail: Jedrzej.Musial@cs.put.poznan.pl

shopping done at minimum total expense. It is worth pointing out that dividing the original shopping list into several sub-lists with items being delivered by different providers increases delivery costs. These are counted and paid individually for each package (sub-list) assigned to a specific Internet shop in the optimization process. Authors focused on the problem definition and its complexity analysis (with some sub-cases of the main problem). The first algorithms and computational results were introduced in [4].

Much more complicated version of ISOP that takes into account price discounts was introduced in [5, 6]. The Focus was on the problem definition and complexity analysis. Some basic algorithms were introduced. The current paper is focused on the introduced problem with new achievements and contributions to the shopping optimization topic.

ISOP could be perceived as a base general name for Internet shopping problems since it leaves a lot of space for future additional requirements and attributes. Introducing complicated multi-objective decision-aided ISOP or a multi-objective problem could be one the upcoming interesting research steps. Many additional decisions could be made by customers with respect to their personal shops preferences, trust, delivery time, negative factors, and all other significant elements. Since the number of decisions could increase, it would be reasonable to support the client with a specific system or tool. This program would attempt to help them to make reasonable decisions that satisfy the customer and do not imply significant price changes. An interesting idea of providing a simple tool that helps decision makers make good decisions in the financial market is described in the literature [7].

In this paper we address the Internet shopping optimization problem considering delivery costs. It is of interest from the customer side to investigate an extended version of the problem which also considers price discounts [5, 6] offered by different sellers. The discount policy is based on real world observations and similar to that used in [8], in the sense that is expresses the discount as a function of the total quantity of money spent in the web-store; the more money a customer spends, the bigger discount he may obtain. In this work, we assume that there are no differences between the quality of the goods the web-stores sell beside the prices they charge for the different products.

We introduce a new set of heuristic approaches to solve the problem. The set of heuristics is composed of a new lightweight metaheuristic based on a cellular optimization process, a extended greedy algorithm and two state-of-the-art greedy algorithms. We have designed the heuristics as a compromise solution balancing computational time and results close to the optimum solution. We empirically evaluate the heuristics on large set of real world data. The set of data was collected from 32 stores and covered the largest United States-based stores, including Amazon, BarnesandNoble.com, Borders.com, Buy.com, Booksamillion and top sellers among Internet bookstores in Poland such as empik.com and merlin.pl. We compare the performance of the proposed heuristics to the optimal values. The optimal solutions for small problem instances are computed using a branch and bound algorithm. To evaluate the scalability of the heuristics, we increased the problem size.

To enlighten the pure original content of the paper one should notice such important elements as: new original cellular algorithm created especially to solve a problem, redesigned forecasting algorithm, problem-specific modification of a state-of-the-art algorithm, new world data model as a data generator for an experiment, original designed experiment with a vast number of computational test, huge portion of computational results carefully analyzed and described by the authors.

The paper is organized as follows. Section 2 describes the problem model. In Section 3, some of the relevant work is presented. In Section 4, we consider the extended version of the investigated problem. Section 5 provides a detailed description of a new set of proposed heuristics. It is followed by experimental results presentation in Section 6. We conclude the paper in Section 7.

2. Problem definition

Notation used throughout this paper is given in Table 1.

Table 1
Notation

Symbol	Explanation
M	set of products
N	set of shops
m	number of products
n	number of shops
i	product indicator
j	shop indicator
M_j	multiset of products available in shop j
d_j	delivery price of all products from shop j
y_j	usage indicator for shop j
p_{ij}	cost of product i in shop j
x_{ij}	usage indicator for product i in shop j
T	cumulative value of all products bought in all shops
T_j	cumulative value of all products bought in shop j
$f_j(Tj)$	piecewise function for all products bought in shop j
$X = (X_1,\ldots, X_n)$	sequence of selections of products in shops $1,\ldots, n$
$F(X)$	sum of product and delivery costs
$d(x)$	0-1 indicator function for x = 0 and x > 0
X^*	optimal sequence of selections of products
F^*	optimal (minimum) total cost

Internet shopping optimization problem (ISOP) should be described as follows. A single buyer is looking for a multiset of products $M = \{1, \ldots, m\}$ to buy in n shops. A multiset of available products M_j, a cost p_{ij} of product i in store j, and a delivery cost d_j of any subset of the products from shop j. It is assumed that $p_{ij} = \infty$ if $i \notin M_j$. The problem is to find a sequence of disjoint selections (or carts) of products $X = (X_1, \ldots, X_n)$, which we call a *cart sequence*, such that $X_j \subseteq M_j, j = 1, \ldots, n$,

$\cup_{j=1}^{n} X_j = M$, and the total product and delivery cost, denoted as $F(X) := \sum_{j=1}^{n} \left(\delta(|X_j|)d_j + \sum_{i \in X_j} p_{ij}\right)$, is minimized. Here $|X_j|$ denotes the cardinality of the multiset X_j, and $\delta(x) = 0$ if $x = 0$ and $\delta(x) = 1$ if $x > 0$. Its solution is denoted as X^*, and its solution value as F^*.

It has been proved that ISOP belongs to the NP-hard problems family [3].

3. Related work

Motivated by the problem of buying multiple products from different web-stores, [3] modeled Internet shopping as an optimization problem, in which a customer wants to buy a list of products from a set of online stores at the minimum final price. The authors showed that the problem is NP-hard in the strong sense and designed a set of polynomial time algorithms for special cases of the problem. During previous research different versions (specializations) of the Internet shopping problem were examined [5]. For example, due to NP-hardness of the optimization problem, [9] 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. Moreover, it is proven that the problem is not approximable in polynomial time [3]. The archetype of the presented problem was a web-based customer assistance system dedicated to pharmacy shopping that helps customers find shops in a geographically defined range where the entire shopping list could be realized at the best total price [10]. For the general discussion on the e-commerce problems development and the way it evolves over last years one should refers to the situation in the recent past [11].

It is worth noting that there are some similarities between ISOP and the well-known facility location problem (FLP) [12]. The main characteristics of the FLP are space, the 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 (continuous problem) or in a subset of given positions (discrete problem) and to assign customers to the opened facilities so that the sum of opening costs and costs related to the distances between customer locations and their corresponding facility locations is minimized.

Discussions of FLPs can be found in [13–16]. The traditional discrete FLP is NP-hard [17] in the strong sense. Note, however, that the general problem 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. The important point to note here is that this problem and ISOP are not each other's sub-cases, while the traditional discrete FLP is a special case of any of these problems.

Looking at the Internet shopping optimization problem with the focus on price discounts, one can notice some similarities with total quantity discount problem (TQD) [8]. To show the similarities and most of all to show distinct differences we should enclose mathematical formulation of TQD. One can define G as the set of m goods, indexed by k, and S as the set of n suppliers, indexed by i. For each good k in G, d_k as the amount of good k to be procured is defined. To each supplier i in S we associate a sequence of intervals $Z_i = \{0, 1, \ldots, maxi\}$, 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 needs to be ordered from supplier i to be in 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 for one item of good k purchased from supplier i in its j-th interval.

Similarities between ISOP with price discounts and TQD problem can be noticed if we threat products to buy M as goods G (d_k is amount of the same good k) and shops N as suppliers S. Piecewise discounting function f_j for shop j will be associated with a sequence of intervals Z_i for supplier i. Price p_{ij} of product i from shop j can be shown as price $c_i k$ for one item of good k purchased from supplier i. Piecewise function for shop j applied to a product price $f_j(p_{ij})$ should be treated as c_{ijk} – price for one item of good k purchased from supplier i in its j-th interval. However ISOP includes shipping costs that are specific to each shops. This feature makes ISOP new enhanced version of the TQD problem.

It is worth pointing out that the decision version of the TQD problem is strongly NP-complete. Moreover, no polynomial-time approximation algorithm with constant worst-case ratio exists for the TQD problem (unless $\mathscr{P} = \mathscr{NP}$). More information on many variations on TQD (i.e. permissible delay in payments in [18]), solutions, exact algorithms [19] can be found in the literature [8, 20, 21].

4. Extended model

In this section we present a known model of the optimization problem, Internet shopping optimization problem considering delivery costs and including price discounts (ISOPwD) [5]. Its mathematical program can be written as follows:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} f_j(p_{ij}x_{ij}) + \sum_{j=1}^{n} d_j y_j,$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1,\ i = 1, \ldots, m,$$

$$0 \le x_{ij} \le y_j,\ i = 1, \ldots, m,\ j = 1, \ldots, n,$$

$$x_{ij} \in \{0, 1\},\ y_j \in \{0, 1\},\ i = 1, \ldots, m,\ j = 1, \ldots, n,$$

where a customer wants to buy products from a given set $M = \{1, \ldots, m\}$ in a given set of Internet shops $N = \{1, \ldots, n\}$ at the minimum total final price. There are the following given parameters and decision variables:

d_j – delivery price of all products from shop j,
y_j – usage indicator for shop j,
p_{ij} – standard price of product i in shop j,
x_{ij} – usage indicator for product i in shop j,
$f_j(T_j)$ – piecewise function (discounting) for final price of all products T bought in shop j.

A piecewise function model is non-linear according to its original nature and it is why we are using non-linear model here.

Proposition 1. In [3] ISOP is demonstrated to be part of NP-Hard problems. The ISOP can be reduced to the basic ISOPwD problem.

Statement 1. The ISOPwD is NP-Hard in the strong sense.

Basic ISOP problem (known as strongly NP-hard) can be transformed to the ISOPwD considering the sub-case for which all piecewise functions (discounts for each shop $n \in N$) equals $f_j(T_j) = T_j$. Therefore the extended ISOPwD problem is NP-hard in the strong sense.

5. Proposed algorithms

The ISOPwD is strongly NP-hard. Moreover, to our best knowledge it cannot be reduced to one of the known problems. Therefore, it is apparent right to propose heuristic solution, simple efficient greedy algorithms [5, 22] that use local knowledge and do not allow any backtracking for efficiency purpose. It is worth to notice that the greedy algorithm does not always yield optimal solutions. However, it could provide an optimal or close to optimal solution using much less resources and time than other optimal working algorithms (i.e., full scan).

5.1. Greedy algorithm. In the first heuristic for the ISOPwD, denoted as Greedy [9], products are considered in a certain order. The algorithm is run for various product orders and the best solution found is presented to the customer. Let us consider that the products are sorted in an ascending order $1, \ldots, m$. Values of the total delivery for each store are initially set as d_j, $j = \{1, \ldots, n$. y_j is the shipping indicator. The standard price for each product i in shop j is set as p_{ij}. In iteration i of Greedy, product i is selected in its eligible shop j with minimum value $T + f_j(p_{ij}) + d_j$, and the corresponding T-value is re-set: $T = T + f_j(p_{ij}) + d_j$. Afterwards, d_j is set to 0.
Piecewise function $f_j(p_{ij})$ returns value of product i after applying discount for shop j.

We observed that Greedy demonstrates very good performance on the experimental data. However, it can provide a solution whose value is much worse than the optimum. One can consider products and delivery prices in Table 2. The first experimental computation results of Greedy can be found in [5] and [9].

For any product sequence algorithm Greedy selects all products in shop 1, which costs $nW - \varepsilon$, while an optimal solution is to select all products in shop 2, which costs W.

Table 2
Price structure for poor performance of *Greedy*

	prod 1	prod 2	…	prod n	delivery
shop 1	$W - \varepsilon$	$W - \varepsilon$	…	$W - \varepsilon$	0
shop 2	0	0	…	0	W

5.2. Algorithm with forecasting. Observed weak points of Greedy led to the creation of a new, upgraded version. The local step choice analysis is more complicated than in the basic Greedy. The forecasting method is looking for a step ahead [4]. Therefore, technically this algorithm is not a strict greedy algorithm. Sometimes it proposes a current solution which is not optimal for the current step (local solution) but for a better overall solution in hope of providing an optimal global solution.

In order to hedge against the instances similar to that in Table 2, we developed another heuristic algorithm, denoted as Forecasting. The main idea is to check the situation one step ahead (forecasting bad situations). From the first step to the penultimate, the algorithm calculates "rating" to pick an eligible shop j for product i. Instead for looking for a local optimum it looks one step ahead (which prevents a bad case from occurring) and calculates the choosing factor as $T + \frac{f_j(p_{ij}+p_{ij+1})+d_j}{2}$ for every shop j and picking the one with the lowest calculated value. In each following step next product i is taken into account, $i = i + 1$. T is set as $T = T + f_j(p_{ij}) + d_j$. Afterwards, d_j is set to 0.
The last step of the algorithm works in a different way (forecast could not work beyond the set of products i). The last product is selected in its eligible shop j with minimum value $T + f_j(p_{ij})$.

Piecewise function $f_j(p_{ij} + p_{ij+1})$ returns total costs of products i and $i + 1$ after applying discount for shop j.

5.3. Cellular processing algorithm. The cellular processing based algorithm is a new pseudo-parallel optimization approach [23]. It includes multiple processing cells that explore different regions of the search space. Each processing cell can be implemented using population or search based heuristics or an hybridization of them. The main idea and the principle of the algorithm is to split a sequential algorithm into several pseudo-parallel processing (i.e., cell) modules, so that each cell can explore different regions of the search space. The main feature of the new approach is that the iterative verification of the stagnation conditions prevents wasting time on unnecessary tasks.

We design an new algorithm based on a pseudo-parallel optimization approach introduced by [23]. The new algorithm we design is a cellular processing approach, which includes multiple processing cells that helps to explore different regions of the search optimization space.

The components of the algorithm are a *pool* of candidate solutions, generated either by a constructive or a random algorithm and a *cell* set that is simple, independent, self-contained and applied to work with the subset of candidate solutions that were given to solve. This process continues until the cells stall all solutions in their local optimum. After that, the solutions return to the pool, and the cells share information with each other in order to escape from the local optimum and continue the search for the global optimum.

In this work we generated the candidate solutions at random. We designed an iterated local search algorithm (ILS) as the core of the cells. This choice was made due to the simplicity and high configurability of this structure, which allows it to be highly scalable and to run in a variety of hardware configurations.

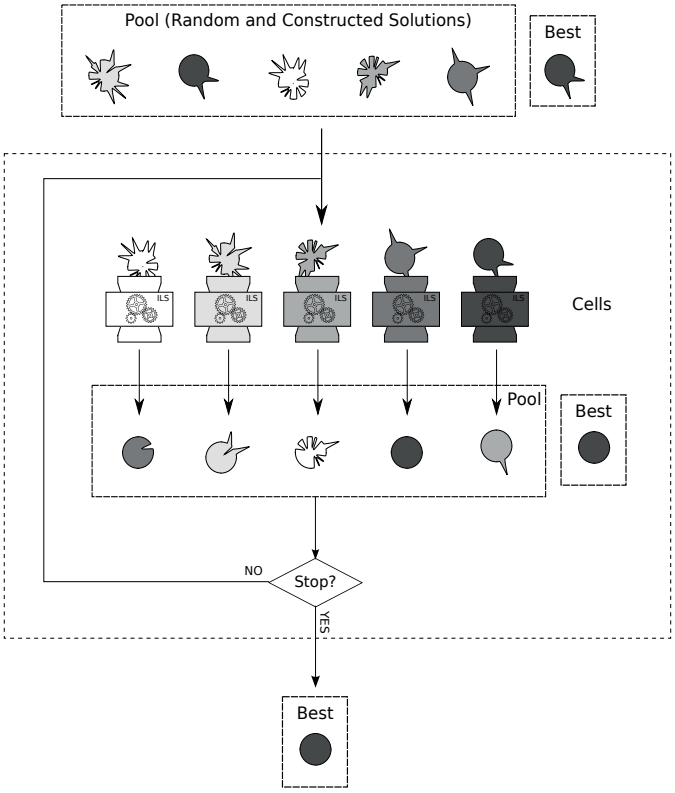

Fig. 1. Algorithm Cellular-ILS

Moreover, the ILS algorithm is a trajectory-based metaheuristic that can be seen as a straight-forward, yet powerful technique for extending simple local search algorithms.

The algorithm starts off by generating an initial solution. Then, a local search process is applied to the candidate solution. After that, following an iteration based approach, it seeks to improve the solutions from one iteration to the next. At each iteration, a perturbation of the obtained local optimum is carried out. The perturbation mechanism introduces a modification to a given candidate solution to allow the search process to escape from a local optimum. A local search is applied to the perturbed solution. The new solution is then evaluated and accepted as the new current solution under some conditions. The algorithm finishes when the termination condition is met.

The proposed local search algorithm comprises the following methods:

- The *criticalElements* obtains a list of elements that may be promising in finding a better search region. This case comprises products that are more expensive and are bought only in one store, so the heuristic favors elements that are cheaper and are bought in the same store.
- The *evaluateChange* method explores the entire neighborhood of search to find the permutation that improves the current solution, but without perturbing it, and returns the store in which it is more convenient to purchase the product under consideration.
- The *makeChange* makes the proposed change given by *evaluateChange*.
- Once the change is performed, the *applyDiscount* method applies the discounts on the price of the products (piecewise function).

The number of cells used in this configuration was established to five, each one starting from a different region of the search space and performing the process described beforehand. The communication, once all were stagnated, was done by comparing the quality of the solutions obtained by each processing cell.

The stagnation was determined by the number of consecutive iterations without improvement, that in the case of each cell was established to ten iterations and for the whole process to five iterations without improvement.

Coded algorithm was carefully designed and build especially to solve ISOPwD problem taking into account its specific and original nature. The algorithm is original and created from the basis especially for this purpose.

It is natural to try to relate *Cellular* to Hyper heuristic. However, it is worth pointing out that Cellular differs from the Hyper heuristic approach in that each processing cell has complete knowledge on the problem that is being solved, as opposed to the Hyper heuristic, where exists a domain barrier among the controller and the low level heuristics. Another main difference resides in that the processing cells are adapted and tailored to the problem that is being solved, while the Hyper heuristic depends on generic low level heuristics that can be applied to a greater variety of optimization problems, according to Burke et al. [24].

5.4. Minimum-minimum algorithm. The min-min algorithm is commonly and widely used in the context of scheduling independent tasks in distributed computing to minimize the total completion time of the tasks [25, 26]. One of the main advantages of this approach is that in general is capable to obtain good quality solutions with a relatively small computational cost, but as it schedules those tasks or processes with minimum cost first, it may result in an imbalanced solution [27]. [28] described the process of the heuristic min-min approach. In this paper we extended min-min to ISOPwD for the selection of a list of products in a set of web-stores. The extended version is called *MinMin* in this paper.

The process begins with the search of the product in the list of unassigned products N, which minimizes the total cost TC in the shopping cart among the different stores M, given the cost of the product, C_{ij} plus delivery cost, D_i. In the case of a tie, the product with the lower delivery cost is selected, and if both stores have the same delivery cost, the product is chosen randomly.

Once assigned, the total cost of the shopping cart is updated with the selected product, and it is removed from the unassigned product list N. The process continues until the unassigned product list is empty.

5.5. Branch and bound algorithm. To calculate the optimal solution, we designed a branch and bound algorithm [29].

The branch and bound (BB) algorithm starts off by calculating an upper bound (UB) employing the solution given by

the Cell Processing Algorithm and proceeds to branch the first level of the search tree in the stack. Subsequently, the algorithm pops the top element of the stack and evaluates the objective value it would have if it were part of the current solution. If the partial solution exceeds the limit given by the upper bound (UB), the current branch is fathomed. Otherwise, if it were not a leaf of the tree, the algorithm would pile up the following elements within the stack. If it were a leaf, it would mean that the current solution is better than the best global solution found so far. Consequently it would update the upper bound with the new value found.

This process continues as long as there are elements in the stack, which means that the whole search tree has been explored. Founded upper bound is now considered as the optimal solution of the instance being evaluated. We consider it as a B&B model, even if it was implemented by ourselves, as it follows the procedure described in the literature. Generating the search tree and stopping further exploration where it's not possible to find a better solutions on a branch.

6. Computational experiments results

Computational experiments were performed and divided into two groups (due to the computational complexity time) – a set of experiments including BB exact algorithm and a set of experiments without optimal solution as a comparison of all heuristics to evaluate scalability issues by increasing instances' size.

All algorithms were implemented in the PHP programming language in its 5.2 version. The aim of using PHP was to easy embed the algorithms in a website – to work in a conditions that are similar to its future final version.

The experiments were carried out in an Apple MacBook Pro, version 8.1 with an Intel processor running at 2.3 GHz, two physical cores and two virtual for each one and 4 GB DDR3 main memory at 1333 MHz. The experimentations were made on a virtual machine running GNU/Linux Mint 11 on a Mac OS X 10.7 host with two virtual cores assigned and 2 GB of main memory.

6.1. World working model and instances generator. A challenging step in experimental research was to create a model, which would be as close to real Internet shopping conditions as possible. An experiment with real world data would be the next step in research, which involves much more dialog with the business community. We studied the relationship between the competitive structure, advertising, prices and price dispersion over Internet stores. As a group of representative products to be taken into account in our computational experiment we chose books, because of their wide choice in web-stores and frequency of purchase through this kind of shopping channel. We used some information from [30]. It focuses mainly on electronic bookstores model definition, prices, acceptance factor, retailer brand [31] and, what is important for the optimization problem model definition, price dispersion. Moreover, we analyzed many Internet stores (i.e. Amazon, BarnesandNoble.com, Borders.com, Buy.com, Booksamillion and top sellers among Internet bookstores in Poland such as empik.com and merlin.pl) to create our own model with instances generator as close to reality as possible.

The working model was prepared as follows. In the computational experiments we assume that $n \in \{20, 40\}$, $m \in \{2, 3, \ldots, 10, 15, \ldots, 100\}$. For each pair (n, m), 100 instances were generated. In each instance, the following values were randomly generated for all i and j in the corresponding ranges. Delivery price: $d_j \in \{5, 10, 15, 20, 25, 30\}$, *publisher's recommended price* of book i: $r_i \in \{5, 10, 15, 20, 25\}$, and price of book i in bookstore j: $p_{ij} \in [a_{ij}, b_{ij}]$, where $a_{ij} \geq 0.69 r_j$, $b_{ij} \leq 1.47 r_j$, and the structure of intervals $[a_{ij}, b_{ij}]$ is prepared as follows:

$$
\begin{aligned}
&[32\%] && minimum \\
&[9\%] && minimum + \tfrac{(median - minimum)}{4} \\
&[9\%] && minimum + \tfrac{(median - minimum)}{2} \\
&[8\%] && minimum + \tfrac{(median - minimum)}{1.33} \\
&[13\%] && median \\
&[6\%] && median + \tfrac{(maximum - median)}{4} \\
&[11\%] && median + \tfrac{(maximum - median)}{2} \\
&[12\%] && median + \tfrac{(maximum - median)}{1.33} \\
& && maximum
\end{aligned}
$$

where $minimum = 0.69r$ and $maximum = 1.47r$.

Based on real world observation the following discounting function was proposed.

$$
f(x) = \begin{cases} x & \text{if } x \leq 25, \\ 0.95x & \text{if } 25 < x \leq 50, \\ 0.90x & \text{if } 50 < x \leq 100, \\ 0.85x & \text{if } 100 < x \leq 200, \\ 0.80x & \text{if } 200 < x. \end{cases}
$$

This kind of advertisement is well-known and very often used by sellers. The more money is spent, the more discount one can achieve.

6.2. A set of experiments including branch and bound algorithm. The first group of experiments is the one in which the optimal solutions obtained by the exact BB algorithm were compared to two state-of-the-art heuristic algorithms: Greedy, Forecasting, and two newly developed algorithms: Cellular

and MinMin (for the ISOPwD). In these examples $n \in \{20\}$, $m \in \{2,3,4,5,6,7,8,9,10\}$, and discounts follow the proposed piecewise discounting function. For each pair (n,m), 100 instances were generated using the information in subsection 6.1. The number of instances with optimal solutions computed by BB is equal to 900.

The algorithms were compared using three metrics: an approximation factor, the run time spent by each heuristic to compute a solution, and the dispersion analysis based on the standard deviation. The approximation factor of a heuristic is defined as $\rho = \frac{F(X)}{F(X)^*}$, where $F(X)$ represents the solution found by a heuristic and $F(X)^*$ denotes the optimal solution.

Figure 2 depicts the average solution of the aggregate values of the approximation factor.

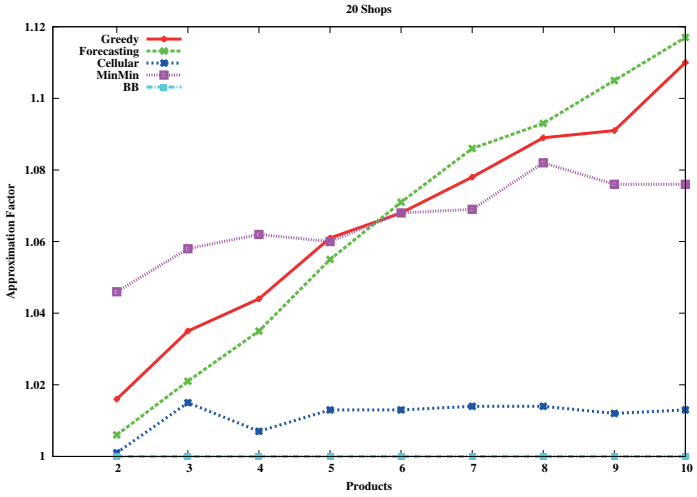

Fig. 2. Algorithm results comparison – experiment with 20 shops (including the optimal solution). The approximation factor is $\rho = \frac{F(X)}{F(X)^*}$, where $F(X)$ represents the solution found by a heuristic and $F(X)^*$ denotes the optimal solution.

Among all heuristics, Cellular provides the best quality of solutions (closest to the optimum), regardless the size of the instances. In the worst case, the solution proposed by this algorithm was merely 1.47% more expensive than the cheapest one. For many instances Cellular algorithm computes the optimal solution. Cellular computed the optimal solution for 62% of the instances. It is worth noting that the algorithm is very stable as regards to the quality of solutions (for most cases the solution is between 1.24% – 1.47% worse than optimum).

Both Greedy and Forecasting algorithms provide similar quality of solutions (for a lower number of products m, the latter was better and for a higher number of products $m > 5$ the former outperformed Forecasting). Moreover, it is easily noticeable that the quality of solution degrades with the increasing number of products m. Greedy was able to find the optimal solution for 6% of the instances and Forecasting computed the optimal solutions for 7% of the instances.

The last heuristic algorithm – MinMin provides the worst solutions for a lower number of products regarding the rest of algorithms. However, the algorithm is quite stable in quality, therefore for a bigger number of products it provides better solutions than Greedy and Forecasting. The algorithm was able to find the optimal solution of 20% of the instances.

If one is looking solely for the quality of solutions the undisputed leader among algorithms is Cellular. Using this algorithm the customer is able to save 5.88% more of the total cost than using MinMin, 6.3% and 6.64% more than Greedy and Forecasting, respectively.

The performance of Cellular is as expected given the search iterative process and the iterative verification of the stagnation conditions to escape from local optimal solutions different than the constructive procedure of the rest of the developed algorithms.

Figure 3 shows the results considering dispersion (including the optimal solution). Each point represents the standard deviation value (deviation around optimum value) from 100 computational tests for 20 shops, where every value for each test is presented as correlation of the result to the optimum ($\frac{Greedy}{BB}$, $\frac{Forecasting}{BB}$, $\frac{Cellular}{BB}$, $\frac{MinMin}{BB}$). Higher standard deviation means more unstable work (big difference in distance from the optimal solution) over 100 tests from a given instance (sometimes optimum or close to, sometimes quite far from it). As a reference value the BB algorithm results were used, as it computes the exact solution. In each cell the standard deviation value is presented for the same instance of n shops and m products for 100 computational tests. It can provide us with very useful information on the stability of the algorithms' results or the lack thereof.

Table 3 provides more detailed information which is represented by the coefficient of variation (CV) normalized measure (percentage value). Each cell represents the coefficient of variation (CV) normalized measure (deviation around optimum value) from 100 computational tests for 20 shops where every value for each test is presented as a correlation of the result to the optimum. A higher CV value means more unstable work (big difference in distance from the optimal solution) over 100

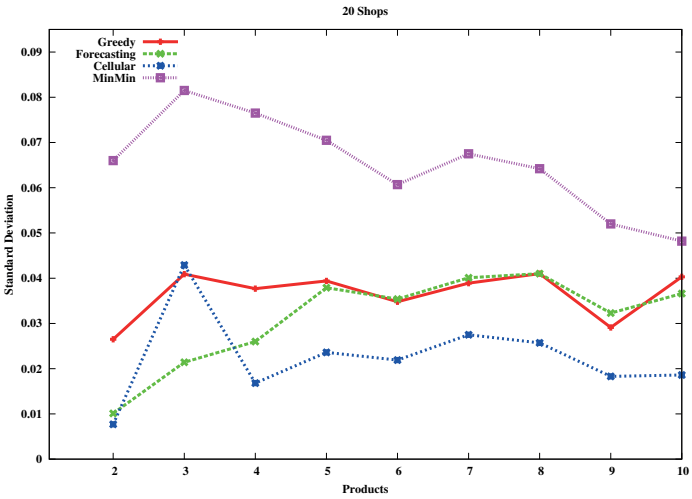

Fig. 3. Algorithm standard deviation chart – experiment with 20 shops (including the optimal solution)

Table 3
A comparison of algorithm results' dispersion for the coefficient of variation (CV) normalized measure

products	Greedy	Forecasting	Cellular	MinMin	BB
2	2.62%	1.02%	0.78%	6.33%	0.00%
3	3.96%	2.11%	4.23%	7.67%	0.00%
4	3.63%	2.53%	1.68%	7.21%	0.00%
5	3.72%	3.61%	2.33%	6.55%	0.00%
6	3.27%	3.31%	2.17%	5.69%	0.00%
7	3.62%	3.70%	2.71%	6.32%	0.00%
8	3.78%	3.81%	2.54%	5.93%	0.00%
9	2.67%	2.93%	1.82%	4.84%	0.00%
10	3.64%	3.28%	1.84%	4.48%	0.00%

test from a given instance (sometimes optimum or close to, sometimes quite far from it). We used the results of the BB algorithm as a reference value. In each cell the CV value is presented for the same instance of n shops and m products for 100 computational tests.

An important factor to consider in the context of online shopping is the run time needed by any algorithm to compute a solution. Figure 4 displays the results regarding the comparison of algorithms run time in microseconds [ms] (including the optimal solution). Each cell represents the average value from 100 computational tests for 20 shops.

For a low number of products ($m \leq 5$) MinMin is the fastest algorithm. It can be observed that when the number of products is bigger than five ($m > 5$), Greedy outperforms the rest of algorithms. Differences between all algorithms are very significant.

Run time execution for BB grows exponentially and it was impossible to prepare experiments for a bigger number of products. On the other hand, all heuristics are very fast so the idea is to further test its quality and run time for scalability issues by increasing the number of products m. The results are presented in the next subsection after the dispersion analysis.

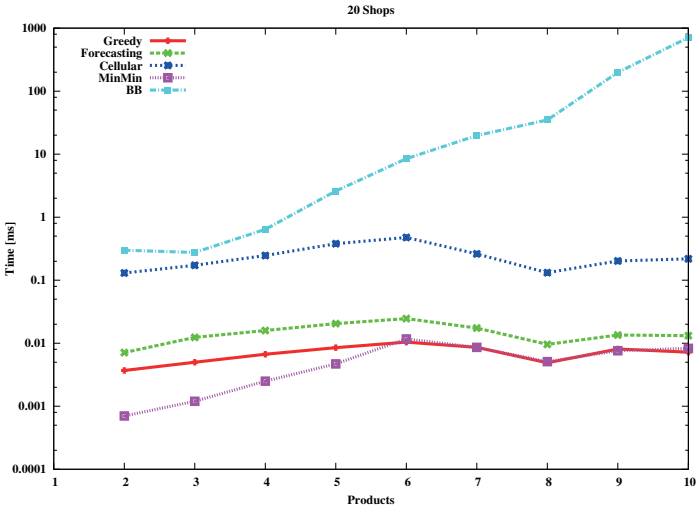

Fig. 4. Algorithm run time comparison – experiment with 20 shops (including the optimal solution)

6.3. Scalability: a set of experiments for heuristic algorithms. In the second group of experiments a comparison between the set of algorithms was done regarding scalability. In these examples we increase the size of the instances as follow: $n \in \{20, 40\}$, $m \in \{5, 10, 15, \ldots, 100\}$, and piecewise discounting functions follow definition from subsection 6.1.
For each pair (n, m), 100 instances were generated. Instances were generated using the same procedure described in Section 6.2.

As we are not able to compute the optimal solution value in a reasonable computational time given the size of the instances we evaluate the relative error γ of each strategy under each metric. The relative error is formally defined as $\gamma = \frac{F(X)}{F(X)_{best}}$, where $F(X)$ represents the solution found by a heuristic and $F(X)_{best}$ denotes the best solution among all heuristics.

6.3.1. A set of experiments with 20 shops. Figure 5 presents average solution values to the relative error for the 100 instances over $n = 20$ shops, which are obtained by the evaluated heuristics.

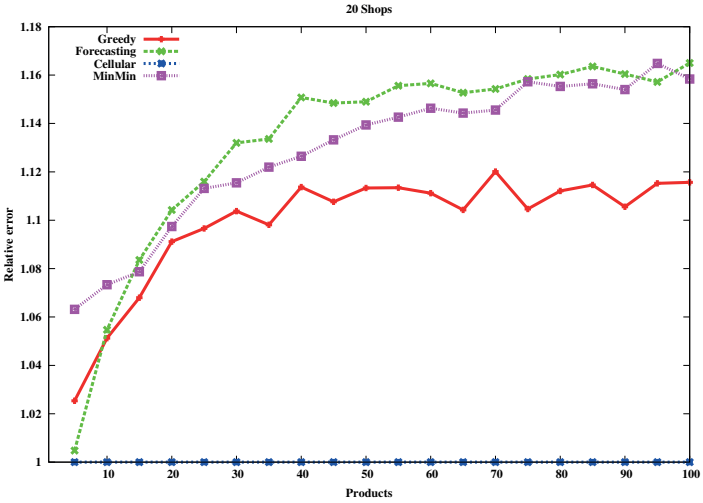

Fig. 5. Algorithm results comparison – experiment with 20 shops (without BB solution). Relative error is $\gamma = \frac{F(X)}{F(X)_{best}}$, where $F(X)$ represents the solution found by a heuristic and $F(X)_{best}$ denotes the best solution among all heuristics

We can observe that Cellular outperforms the rest of the algorithms. For all instances n, m where $m \in \{5, 10, 15, \ldots, 100\}$, it provides the best quality solution. The behavior of Cellular is the same than in the first set of experiments.

Greedy presents better scalability than MinMin and Forecasting by providing the second best quality of solutions. For more than $m > 15$ products it stabilized within 10–11% of the Cellular algorithm solution. For a low number of

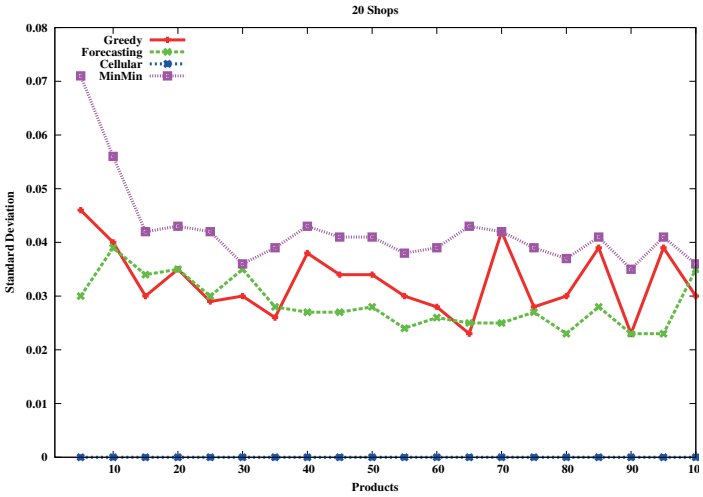

Fig. 6. Algorithm standard deviation chart – experiment with 20 shops

products $m = 5$ Forecasting propose almost the same quality of solution as the best Cellular algorithm (0.48% more expensive). From more than $m > 15$ products it becomes the worst algorithm among the tested ones. Moreover, it is easily noticeable that the quality of solution degrades with the increasing number of products m (compared to the best Cellular algorithm).

Table 4
Heuristic algorithm results' dispersion for the coefficient of variation (CV) normalized measure

products	shops	Greedy	Forecasting	Cellular	MinMin
5	20	4.45%	2.92%	2.62%	6.58%
10	20	3.75%	3.73%	0.94%	5.24%
15	20	2.79%	3.17%	0.00%	3.86%
20	20	3.18%	3.19%	0.00%	3.93%
25	20	2.65%	2.65%	0.00%	3.74%
30	20	2.71%	3.09%	0.00%	3.26%
35	20	2.36%	2.47%	0.00%	3.50%
40	20	3.42%	2.36%	0.00%	3.82%
45	20	3.04%	2.38%	0.00%	3.62%
50	20	3.09%	2.40%	0.00%	3.61%
55	20	2.69%	2.06%	0.00%	3.29%
60	20	2.51%	2.27%	0.00%	3.40%
65	20	2.10%	2.17%	0.00%	3.72%
70	20	3.77%	2.12%	0.00%	3.62%
75	20	2.57%	2.35%	0.00%	3.37%
80	20	2.73%	1.98%	0.00%	3.23%
85	20	3.48%	2.36%	0.00%	3.56%
90	20	2.06%	1.99%	0.00%	3.05%
95	20	3.47%	1.98%	0.00%	3.49%
100	20	2.71%	3.09%	0.00%	3.26%

The last heuristic algorithm – MinMin provides the worst solutions for a lower number of products ($m < 15$). However, for a bigger number of products it provides better solutions than the Forecasting algorithm (but worse than Greedy). MinMin provides solutions between 6% and 16% (on average) bigger than best Cellular solutions.

Figure 6 contains a comparison of heuristic algorithm results dispersion. Each point represents the standard deviation value (deviation around the best value calculated from all algorithms) from 100 computational tests for 20 shops where every value for each test is presented as a correlation between the result and the best algorithm result for this computation $(\frac{Greedy}{best}, \frac{Forecasting}{best}, \frac{Cellular}{best}, \frac{MinMin}{best})$. In each point the standard deviation value is presented for the same instance of n shops and m products for 100 computational tests. It can provide very useful information concerning the quality (based on stability) of the algorithm results. Table 4 contains the information which is represented by the coefficient of variation (CV) normalized measure (percentage value).

Figure 7 exposes information regarding the comparison of algorithm run time [ms]. Each cell represents the average value from 100 computational tests for 20 shops. We can observe that for a low number of products $m = 5$ MinMin is the fastest algorithm. For other instances of the ISOPwD problem (number of products $m > 5$), algorithm Greedy is the fastest. Differences between all algorithms are very significant. The following example illustrates these differences. A few observations are worth noting.

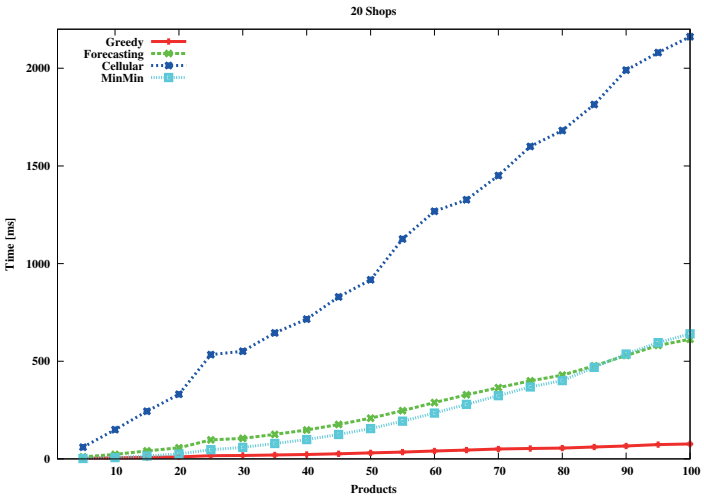

Fig. 7. Algorithm processing time comparison – experiment with 20 shops (without BB solution)

6.3.2. A set of experiments with 40 shops. There were no changes regarding the behavior of Cellular, it provides the best quality of solutions. That is, for all instances n, m it provided the best solution.

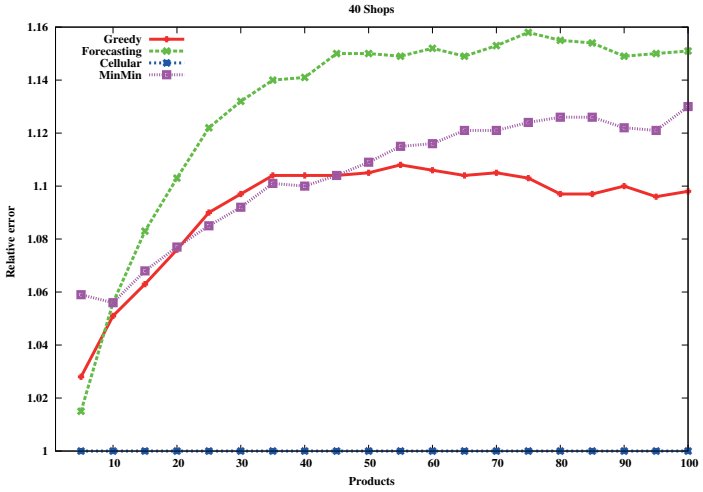

Fig. 8. Algorithm results comparison (Relative error measure) – experiment with 40 shops (without BB solution)

Figure 8 displays relative error average results. We can notice that characteristics of the results are not far away from experiment with $m = 20$ shops.

Algorithm Greedy provides the second best quality of solutions. For more than $m > 25$ products it stabilized within 9.6–10.8% of the Cellular algorithm solutions. For a low number of products $m = 5$ Forecasting computes almost the same quality of solution as the best Cellular algorithm (1.55% more expensive). From more than $m > 10$ products it becomes the worst algorithm among the tested ones. Moreover, it is easily noticeable that the quality of solution degrade with the increasing number of products m (compared to the best {Cellular algorithm}) from 1.55% to 15.83% percent on average greater than Cellular. MinMin provides the worst solutions for a lower number of products ($m = 5$). However, for a bigger number of products it provides better solutions than the Forecasting algorithm (but worse than Greedy). The algorithm also provides better scalability than the previous experiment for smaller number of web-stores.

Figure 9 contains a comparison of heuristic algorithm results dispersion.

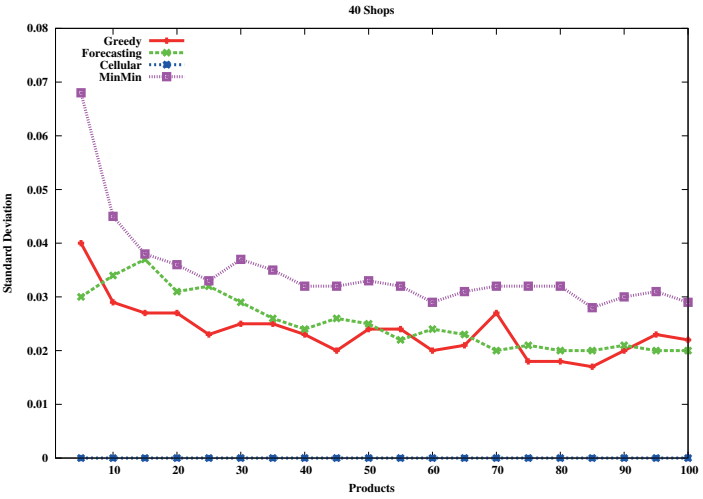

Fig. 9. Algorithm standard deviation chart – experiment with 40 shops

Regarding 40 shops experiment general observations are similar to the ones from the 20 web-stores experiment. However, some differences are significant and worth to notice. All standard deviation and CV values are lower for the experiment with 40 shops. Algorithm Greedy provides a lower dispersion level than Forecasting (slightly but still). Observations for the Cellular and MinMin algorithms are similar to the experiment with 20 web-stores.

Regarding the computation time (see Fig. 10), for all instances with $n = 40$ web-stores, algorithm Greedy is the fastest. Differences between all algorithms are very significant. Calculation times are even more important here, since goal application will work as an online web-site. There is plenty of research showing that users do not want to wait to see a webpage content, because they quickly lose interest. An up-to-date survey is given in [32].

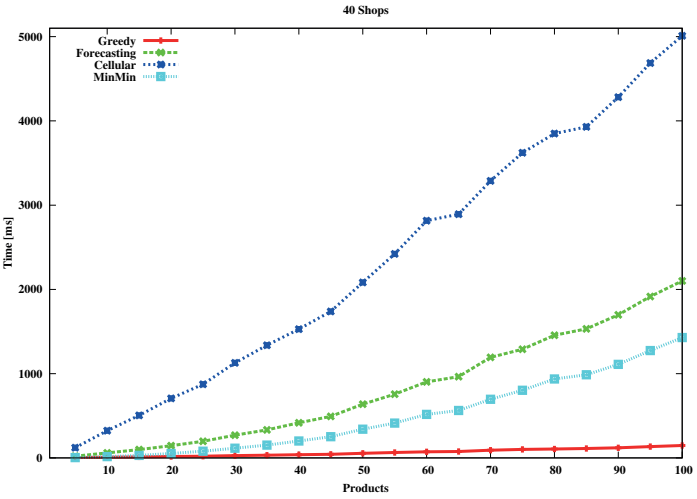

Fig. 10. Algorithm processing time comparison – experiment with 40 shops (without BB solution)

Concerning all results collected in the computational experiment it can be stated that the Cellular algorithm provides the best solutions among all heuristic algorithms. The algorithm also computes more stable solutions than the rest of compared algorithms. On the other hand, it is the slowest one. What is worth noting is that it could be much more efficient when parallelized on five machines (each cell can process on a different machine). Another important consideration is that in a real e-commerce web site the instances of the algorithms will be executed in the server machines (i.e. in a web-server of a data center), which usually are more powerful than customers' machines.

Algorithm Greedy can result in a lot of interest due to very fast processing times in which it can provide a good quality of solutions. Both algorithms Greedy and Forecasting can be parallelized for two machines (both algorithms make two separate executions and pick the best of these two at the final step).

The performance of MinMin can be improved by using a local search algorithm [33]. To improve the run time execution MinMin can be easily parallelized. There are different

parallel implementations of the min-min scheduling algorithm reported in literature [33, 34], both CPU and GPU parallel implementations.

Given the fast run time execution and the good quality of solutions that Greedy, Forecasting and MinMin can compute in average their solutions can be used as initial seeds for Cellular. It could help to improve the run time execution of the algorithm without degrading its performance. Special database system optimization could be performed to decrease all-around computational times [35].

7. Conclusions

In this paper, we addressed the Internet shopping optimization problem including price discounts. For the practical application, we created a working model as close to real Internet shopping conditions as possible. The main reason was their wide choice in Internet web-stores and frequency purchase. A new set of algorithms is presented, three heuristics and a new cellular processing optimization algorithm. Computer experiments demonstrated their good performance. The results generated by the algorithms were compared to the optimal solutions computed by a proposed branch and bound algorithm.

As for current version of the problem (doesn't count yet with a linear model) it is not possible at the moment to perform a relaxation technique on the restrictions or to implement it on a MIP solver such as CPLEX or Gurobi. Nonetheless, presentation of a valid MILP model is certainly worth considering and interesting future step – this will be one of our next challenges.

In the future, we plan to extend the Internet shopping model to include additional constraints such as: minimum delivery times, incomplete shopping lists realization, and maximum budget. Moreover, we plan to propose new quality algorithms for dual discounting function ISOP [36]. The experimental results demonstrated the potential by the new Cellular processing optimization algorithm. However, one of the main concerns for its applicability to ISOP is the time needed to find a solution. To alleviate the problem and also deal with scalability we consider investigating a parallel version of the algorithm on a GPU infrastructure. Furthermore, there are some interesting similarities between ISOP and exciting Cloud Brokering [37]. Linking it with ISOP may result in the possibility of using algorithms prepared for ISOP.

Acknowledgments. This study was partially supported by the FNR (Luxembourg) and NCBiR (Poland), through IShOP project, INTER/POLLUX/13/6466384.

References

[1] K. M. Tolle and H. Chen, "Intelligent software agents for electronic commerce", *Handbook on Electronic Commerce,* Springer Berlin Heidelberg, 365–382 (2000).

[2] S. Rose and A. Dhandayudham, "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 (2014).

[3] J. Blazewicz, M. Kovalyov, J. Musial, A. Urbanski and A. Wojciechowski, "Internet shopping optimization problem", *International Journal of Applied Mathematics and Computer Science* 20 (2), 385–390 (2010).

[4] J. Blazewicz and J. Musial, "E-commerce evaluation – multi-item Internet shopping. Optimization and heuristic algorithms", *Operations Research Proceedings 2010*, 149–154 (2011).

[5] J. Blazewicz, P. Bouvry, M. Y. Kovalyov and J. Musial, "Internet shopping with price sensitive discounts", *4OR-Q J Oper Res* 12 (1), 35–48 (2014).

[6] J. Blazewicz, P. Bouvry, M. Y. Kovalyov and J. Musial, "Erratum to: Internet shopping with price-sensitive discounts", *4OR-Q J Oper Res* 12 (4), 403–406 (2014).

[7] B. Sawik, "Downside risk approach for multi-objective portfolio optimization", *Operations Research Proceedings 2011*, 191–196 (2012).

[8] D. R. Goossens and A. J. T. Maas, "Exact algorithms for procurement problems under a total quantity discount structure", *European Journal of Operational Research* 178 (2), 603–626 (2007).

[9] A.Wojciechowski and J. Musial, "Towards optimal multi-item shopping basket management: Heuristic approach", *Lecture Notes in Computer Science (LNCS)* 6428, 349–357 (2010).

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

[11] M. Shaw, R. Blanning, T. Strader and A. Whinston, *Handbook on Electronic Commerce, International Handbooks on Information Systems*, Springer, Berlin-Heidelberg (2000).

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

[13] J. Krarup, D. Pisinger and F. Plastria, "Discrete location problems with push-pull objectives", *Discrete Applied Mathematics* 123 (13), 363–378 (2002).

[14] H. Eiselt and C. Sandblom, *Decision Analysis, Location Models, and Scheduling Problems*, Springer-Verlag, Berlin-Heidelberg-New York (2004).

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

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

[17] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York, Freeman (1979).

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

[19] D. Manerba and R. Mansini, "An exact algorithm for the capacitated total quantity discount problem", *European Journal of Operational Research* 222 (2), 287–300 (2012).

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

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

[22] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, McGraw-Hill Higher Education, 2nd edition (2001).

[23] J. D. Terán-Villanueva, H. J. F. Huacuja, J. M. C. Valadez, R. A. Pazos Rangel, H. J. P. Soberanes and J. A. M. Flores, "Cellular processing algorithms", *Studies in Fuzziness and Soft Computing* 294, 53–74 (2013).

[24] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology", *Handbook of Metaheuristics*, Springer, 457–474 (2003).

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

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

[27] M. Wu, W. Shu and H. Zhang, "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems", *Heterogeneous Computing Workshop IEEE*, 375–385 (2000).

[28] T. Braun, H. Siegel, N. Beck, L. Bölöni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", *Journal of Parallel and Distributed Computing* 61 (6), 810–837 (2001).

[29] A. Land and A. Doig, "An automatic method for solving discrete programming problems", *Econometrica* 28 (3), 497–520 (1960).

[30] E. W. Karen Clay, Ramayya Krishnan, "Prices and price dispersion on the web: evidence from the online book industry", *The Journal of Industrial Economics* 49 (4), 521–539 (2001).

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

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

[33] F. Pinel, B. Dorronsoro and P. Bouvry, "Solving very large instances of the scheduling of independent tasks problem on the GPU", *Journal of Parallel and Distributed Computing 73* (1), 101–110 (2013).

[34] P. Ezzatti, M. Pedemonte and A. Martin, "An efficient implementation of the Min-Min heuristic", *Computers & Operations Research* 40 (11), 2670–2676 (2013).

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

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

[37] M. Guzek, A. Gniewek, P. Bouvry, J. Musial and J. Blazewicz, "Cloud brokering: current practices and upcoming challenges", *IEEE C*