Novel dual discounting functions for the Internet shopping optimization problem: new algorithms

Jacek Blazewicz · Nathanael Cherié ·
Pierre-François Dutot · Jędrzej Musiał ·
Denis Trystram

Received: 7 November 2013 / Accepted: 30 June 2014 / Published online: 17 August 2014
© The Author(s) 2014. This article is published with open access at Springerlink.com

Abstract One of the very important topics in discrete optimization, motivated by practical applications, is Internet shopping, which is becoming increasingly popular each year. More classical versions of the Internet shopping optimization problem (ISOP) are closely related to the facility location problem and some scheduling problems and have been intensively studied in the literature. In this paper, extensions of the problem are defined and studied. The issue is to buy all the necessary products for a minimum total possible price. This includes all prices of products as well as shipping costs. Studies in this paper include the ISOP with price sensitive discounts and a newly defined optimization problem: the ISOP including two different discounting functions, namely a shipping cost function as well as a price discounting function. First, these are formulated as mathematical programming problems. Then, some algorithms are constructed and extensively tested in a computational experiment.

Keywords Internet shopping · Algorithms · Optimization · Computational complexity · Applications

1 Introduction

Electronic commerce is definitely one of the youngest branches of computer science. Moreover, one could claim that this discipline is also one of the fastest developing fields of computer science. Revenue in the e-commerce market is growing rapidly. One could say that the multidisciplinary nature (connecting many of the well-known branches of computer science such as operations research, combinatorial optimization, and algorithms with other sciences, e.g., logistics, marketing, and many others) lies behind the success of e-commerce science. E-commerce is an industry which focuses on selling and buying products and services through web pages (see Hagel III 1999; Timmers 1998).

Online (Internet) shopping, fitting into the business-to-consumer (B2C) subcategory, is one of the key business activities offered over the Internet. It has become increasingly popular over the past decade. Products available in online stores are often cheaper than the ones offered by regular local retailers, and a wide choice of offers are available just a click away from the customer. A crucial aspect of online shopping is the time spent comparing offers and a convenient way of shopping regardless of shop location. Price comparison sites (Google Shopping, Shopping.com, PriceGrabber, or Kelkoo are among the most popular shopping engines in the EU) are search tools designed to provide price information from many retailers through a single portal. However, current price ranking solutions target only single product buying. A significant percentage of price comparison websites perform suboptimally in one of their major functions: presenting prices and considering multiple products. For a specific decision aid system tackling Internet shopping with self pick-up, see Wojciechowski and Musiał (2009).

J. Blazewicz · J. Musiał (✉)
Institute of Computing Science, Poznań University of Technology,
ul. Piotrowo 2, 60-965 Poznań, Poland
e-mail: jedrzej.musial@cs.put.poznan.pl

J. Blazewicz
Institute of Bioorganic Chemistry, Polish Academy of Sciences,
Z. Noskowskiego str. 12/14, 61-704 Poznań, Poland

N. Cherié · P.-F. Dutot · D. Trystram
Grenoble Institute of Technology, 46 Avenue Félix Viallet,
38031 Grenoble Cedex 1, France

The Internet shopping optimization problem (ISOP) (see [Blazewicz et al. 2010](#)) considers the situation where a single buyer is looking for a multiset of products $N = \{1, \dots, n\}$ to buy in m shops, $M = \{1, \dots, m\}$. A multiset of available products N_i , a cost p_{ij} of each product $j \in N_i$, and a delivery cost d_i of any subset of the products from the shop to the buyer are associated with each shop $i \in M$, $i = 1, \dots, m$. It is assumed that $p_{ij} = \infty$ if $j \notin N_i$. The problem lies in finding a sequence of disjoint selections (or carts) of products $S = (S_1, \dots, S_m)$, which we call a *cart sequence*, such that $S_i \subseteq N_i$, $\forall i \in M$, $\cup_{i=1}^m S_i = N$, and the total product and delivery cost, denoted as $F(S) := \sum_{i=1}^m (\delta(|S_i|)d_i + \sum_{j \in N_i} p_{ij})$, is minimized. Here, $|S_i|$ denotes the cardinality of the multiset S_i , and $\delta(x) = 0$ if $x = 0$ and $\delta(x) = 1$ if $x > 0$. We denote this problem as ISOP, its optimal solution as S^* , and its optimal solution value as F^* . Follow [Wojciechowski and Musial \(2010\)](#) for the first heuristic algorithm definition. Moreover, in [Musial \(2012\)](#) one can find a detailed description of a computational experiment for the problem ISOP and comments on its results. The most interesting (and most complicated) specialization of the ISOP is the so-called Internet shopping with price sensitive discounts problem presented by [Blazewicz et al. \(2014\)](#). For each Internet shop, standard prices for products are known, as well as an increasing discounting function of total standard and delivery price. Buying all the required products at the minimum total discounted price constitutes the problem. Its mathematical program can be written as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^m f_i \left(\sum_{j \in N_i} p_{ij} x_{ij} \right) + \sum_{i=1}^m d_i y_i, \\ \text{s.t.} \quad & \sum_{i \in M} x_{ij} = 1, \quad j = 1, \dots, n, \\ & 0 \leq x_{ij} \leq y_i, \quad i = 1, \dots, m, \quad j = 1, \dots, n, \\ & x_{ij} \in \{0, 1\}, \quad y_i \in \{0, 1\}, \quad i = 1, \dots, m, \quad j = 1, \dots, n. \end{aligned} \quad (1)$$

Note that $f_i(T)$ is defined as a discounting function for the final sum of standard price T in shop i at all points $T > 0$, $f_i(0) = 0$.

It is worth noticing that there are some similarities to the well-known facility location problem (FLP) (see [Revelle et al. 2008](#)). The main characteristics of the FLP are the metric, i.e., space, given customer locations and given or not given positions for facility locations. A traditional FLP is to open a number of facilities at arbitrary positions of space (continuous problem) or at 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. Dis-

cussions of FLPs can be found in [Krarup et al. \(2002\)](#), [Eiselt and Sandblom \(2004\)](#), [Revelle et al. \(2008\)](#), [Melo et al. \(2009\)](#), and [Iyigun and Ben-Israel \(2010\)](#). The traditional discrete FLP is NP-hard in the strong sense. Note, however, that the general problem (basic ISOP with price sensitive 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 distance. It can be noticed that this problem and the basic ISOP problem are not subcases of one another, while the traditional discrete FLP is a special case of either of these problems.

At this point it is also worth mentioning the relation between the ISOP and the scheduling area (see [Blazewicz et al. 2007](#) or [Leung et al. 2004](#) for a general treatment of this topic).

Let us consider a scheduling problem with a set $J = \{J_1, J_2, \dots, J_n\}$ of jobs which need to be scheduled on a set $M = \{M_1, M_2, \dots, M_m\}$ of nonidentical, parallel machines. Additional parameters that describe the problem are:

- p_{ij} : processing time of job J_i on machine M_j
- t_j : set-up (warm-up) time for machine M_j

The goal is to find a non-preemptive schedule that minimizes total time (including set-ups); the machines are used to process the jobs. The equivalence between the ISOP and the above scheduling problem can be easily seen if we assume that machines represent shops, jobs are equivalent to products that have to be purchased, processing times are treated as prices, and the machine set-up times are equivalent to the delivery costs.

As a natural enlargement of the ISOP, a new optimization problem is introduced in this paper: the Internet shopping optimization problem with dual discounting functions. Price sensitive shipping costs are often used in Internet shops to attract customers and to encourage them to buy more products.

After the mathematical programming formulation of this problem, algorithms to solve it are proposed. Extensive tests prove their good computational features.

The organization of the paper is as follows: In Sect. 2 the new problem (dual discounting functions for ISOP) is described and formulated as a mathematical programming problem. Section 3 provides a detailed description of (both exact and heuristic) algorithms used to solve the problem. Section 4 contains theoretical analysis of novel algorithm G3. The following Sect. 5 provides information on a model that was used during computational tests. Section 6 defines the experimental environment, which is followed by presentation of the experimental results in Sect. 7. We conclude the paper in Sect. 8.

Table 1 Example price structure for delivery

Delivery	Payment	$T_i < 25$	$25 \leq T_i < 50$
Post	Credit card	10	5
Post	At pick-up	15	10
Courier	Credit card	15	15
Courier	At pick-up	20	20
Delivery	Payment	$50 \leq T_i < 75$	$T_i \geq 75$
Post	Credit card	0	0
Post	At pick-up	5	0
Courier	Credit card	5	0
Courier	At pick-up	10	0

2 Dual discounting functions for the Internet shopping optimization problem

A typical example is the following advertisement in an Internet shop selling books and CDs: If the value of your purchase is at least 25, then we will ship your products for half the shipping cost; 35—half the price of delivery by a courier; 40—free delivery by post; 50—free courier shipping. Values and thresholds can vary depending on the seller. Price sensitive discounts are often used in Internet shops to attract customers. A typical example is the following advertisement in an Internet shop selling computer parts: If the value of your purchase is more than 50 then your discount is 3 %, more than 100 then 7 %, more than 150 then 10 %, more than 250 then 15 %. Many Internet shops offer free delivery if the price of a purchase exceeds a certain threshold. One can describe the ISOP with two price sensitive functions as follows: We would like to buy a number of products (n). Products are offered by many stores (m) from different locations—some are distant; some are close to our position. The price of each product n in every store m could be written as p_{ij} , where $i = 1, \dots, m$, $j = 1, \dots, n$. To every store m corresponds a discounting function (on the base of the total cost of the shopping from that one store) $f_i(T_i)$. Our goal is to find out how to pay the least for all the products, including delivery costs. The mathematical programming formulation can be written as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^m f_i \left(\sum_{j \in N_i} p_{ij} x_{ij} \right) + \sum_{i=1}^m d_i \left(\sum_{j \in N_i} p_{ij} x_{ij} \right), \quad (2) \\ \text{s.t.} \quad & \sum_{i \in M} x_{ij} = 1, \quad j = 1, \dots, n, \\ & x_{ij} \in \{0, 1\}, \quad i = 1, \dots, m, \quad j = 1, \dots, n. \end{aligned}$$

Both discounting (f) and shipping cost (d) functions can be aggregated to present one common function (fd) based on both criteria. The formulation can be written as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^m (fd)_i \left(\sum_{j \in N_i} p_{ij} x_{ij} \right), \quad (3) \\ \text{s.t.} \quad & \sum_{i \in M} x_{ij} = 1, \quad j = 1, \dots, n, \\ & x_{ij} \in \{0, 1\}, \quad i = 1, \dots, m, \quad j = 1, \dots, n. \end{aligned}$$

The discounting function could look as follows (an example):

$$\begin{aligned} f_i(T_i) = & \dots \\ & T_i \text{ if } 0 < T_i \leq 50. \\ & 50 + 0.97(T_i - 50) \text{ if } 50 < T_i \leq 100. \\ & 50 + 0.97 * 50 + 0.93(T_i - 100) \text{ if } 100 < T_i \leq 150. \\ & 50 + 0.97 * 50 + 0.93 * 50 + 0.9(T_i - 150) \text{ if } 150 < T_i \leq 250. \\ & 50 + 0.97 * 50 + 0.93 * 50 + 0.9 * 100 + 0.85(T_i - 250) \text{ if } T_i > 250. \end{aligned}$$

Note that T_i is the total standard price of books selected in bookstore i . The shipping cost function is more complicated due to its shape, which depends additionally on a customer's later choices (delivery type, e.g., by post office, courier; payment method). We can define it as $d_i(T_i, dv_1, dv_2)$, $i = 1, \dots, m$. Therefore, for the present discussion, we can state that a customer always makes the same decisions dv_1, dv_2 . Shipping cost function examples are presented in Table 1.

To solve this problem one can use some of the already existing heuristics. However, before finishing computational experiments, it cannot be said that these will be as efficient as for the ISOP with price sensitive discounts. Moreover, we can presume that new specific algorithms for this interesting case should be developed.

3 Algorithm design

Finding an exact solution to this problem takes an exponential time, but to be able to compare the results of heuristics with an optimal solution, a branch-and-bound algorithm is implemented with some improvements to skip solutions that are not optimal.

Furthermore, it is interesting to compare the new heuristic with existing algorithms that are already in use in price

comparison sites available on the Internet; let us call them PCS and its improved version PCS+, respectively.

3.1 Algorithms PCS and PCS+

Current price comparison site platforms could be used to solve the ISOP. For testing purposes, we coded both algorithms (PCS and PCS+). As authors of the algorithm, we could indicate Andersen Consulting (now Accenture) with its product BargainFinder including the SmartStore feature, described by Krulwich (1996)—the first widely known tool to compare a music CD's price from different online shops. PCS is a simpler version of the algorithm. For each product j it selects its eligible shop $i \in M_j$ with a minimum value p_{ij} . At the end, it calculates discounts and adds shipping cost. PCS+ is somehow an upgraded version of the former, while all delivery costs are known (basic or flat rates). For each product j it selects its eligible shop $i \in M_j$ with a minimum value $p_{ij} + d_i$, where d_i is a basic price for a delivery from shop i . At the end, it calculates discounts and adds shipping costs. Patents by Bastnagel et al. (2003), Christensen (2010) use the above known functionality.

3.2 Algorithm G1: GREEDY (see Wojciechowski and Musial 2010)

In the first heuristic algorithm, denoted as G1, 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. In the following example, it is assumed that products are ordered $1, \dots, n$. Values of total delivery and standard price for all shops are initially set as $T_i = d_i$, $i = 1, \dots, m$. In iteration j of algorithm G1, product j is selected in its eligible shop $i \in M_j$ with minimum value $f d_i (T_i + p_{ij})$, and the corresponding T_i -value is re-set: $T_i := T_i + p_{ij}$.

The discounting function returns the value of product j after applying the discount for shop i (cf. Algorithm 1).

We observed that algorithm G1 demonstrates very good performance on the experimental data. However, it can provide a solution whose value is n times worse than the optimum. Consider the product and delivery prices in Table 2. For any product sequence, algorithm G1 selects all the products in shop 1 which cost $nW - n$, while an optimal solution is to select all the products in shop 2 with cost W . First impressions on algorithm G1 can be found in Wojciechowski and Musial (2010). What is worth noticing is that G1 is a very

Table 2 Price structure for poor performance of algorithm GREEDY

	Prod 1	Prod 2	...	Prod n	Delivery
Shop 1	$W - 1$	$W - 1$	...	$W - 1$	0
Shop 2	0	0	...	0	W

Algorithm 1 G1

Require: Number of shops m
Require: Number of products n
Require: Products available in shop i , N_i
Require: Order function on the products σ
Require: Functions $f d_i$
Require: Prices p_{ij}
Require: Delivery fee d_i
Ensure: Array of baskets S

```

1:  $N := \{1..n\}$ 
2: for  $i := 1$  to  $m$  do
3:    $S[i] := \emptyset$ 
4: end for
5: for  $j := 1$  to  $n$  do
6:   Select  $k \in M$  such as  $\sum_{i \in M} f d_i (T_i (S[i]) + \delta_{ik} p_{k\sigma(j)})$  minimal
   and  $\sigma(j) \in N_k$ 
7:    $S[k] := S[k] \cup \{\sigma(j)\}$ 
8: end for
9: return  $S$ 

```

fast algorithm. Its complexity is $O(nm)$, which is good to quickly provide an approximation.

3.3 Algorithm G2: Forecasting (see Blazewicz and Musial 2011)

Since an approximation can be n times worse than the optimal solution (following Blazewicz et al. 2010), G2 (Algorithm 2) has been designed to counterbalance this problem.

G2, unlike G1, starts by choosing a shop in which items should be bought; the one chosen has the lowest average price of items including delivery fees. For each shop i the valuation $V_i = (d_i + \sum_{j \in N_i} p_{ij}) / |N_i|$ is calculated, then from the shop k which has the lowest valuation V_k , the set S of the $\alpha\%$ cheapest available items is taken. Then those items are removed from the shops $\forall i \in M$, $N_i := N_i \setminus S$, and d_k is updated to the value $d_k = 0$, since the delivery fee for shop k has already been paid.

The complexity of G2 is $O(n^2m)$, which is slightly more than for G1. Nonetheless, it is still linear in the number of shops, the parameter that should be biggest for an average consumer.

These two heuristics provide efficient results. However, to get the best out of them, both must be used with many parameters and then the best solution obtained taken. Driven by these observations and the fact that they need some parameters, a new algorithm has been created.

3.4 A new algorithm: G3

The idea behind the new heuristic that is proposed, denoted as G3 (Algorithm 3), is to make groups of products to maximize the ratio V_i of how much is saved to how much it costs at its maximum in all the shops. In order to do so, for every shop, the item with the biggest ratio is put aside into a temporary basket C_i . If by adding another product this ratio can be

Algorithm 2 G2

Require: Number of shops m
Require: Number of products n
Require: Products available in shop i , N_i
Require: Functions $f d_i$
Require: Prices p_{ij}
Require: Delivery fee d_i
Require: Parameter α
Ensure: Array of baskets S

```

1: while  $N \neq \emptyset$  do
2:   for  $i \in M$  do
3:     if  $S[i] = \emptyset$  then
4:        $V[i] := (d_i + \sum_{j \in N_i \cap N} p_{ij}) / |N_i \cap N|$ 
5:     else
6:        $V[i] := (\sum_{j \in N_i \cap N} p_{ij}) / |N_i \cap N|$ 
7:     end if
8:   end for
9:   Select  $k$  such as  $V[k] = \min_{i \in M \setminus \{N_i \cap N \neq \emptyset\}} V[i]$ 
10:  for  $c := 1$  to  $\max(1, \alpha * |N_k \cap N|)$  do
11:    Select  $l$  such as  $p_{il} = \min_{j \in N_i \cap N} p_{ij}$ 
12:     $S[k] := S[k] \cup \{l\}$ 
13:     $N := N \setminus \{l\}$ 
14:  end for
15: end while
16: return  $S$ 

```

increased, the product is also taken into C_i , and this process continues until there are no more items available in the shop or the ratio cannot be increased any more. Then the shop k with the highest ratio V_k is chosen, and all the items in C_k are placed in B_i to be bought from this shop. This step is repeated until all the products on the shopping list have been bought.

The expression for V_i takes into account the basket B_i containing everything already chosen for this shop as well as the temporary basket C_i :

$$V_i(C_i, B_i) = 1 - \frac{f_i(T_i(B_i \cup C_i)) - f_i(T_i(B_i))}{p_{\max}(C_i)}$$

G3 has complexity $O(n^3 m)$, which is more than G1 (Algorithm 1) and G2 (Algorithm 2), but G3 does not need any extra tuning and does not depend on the order of items.

G3 has some interesting properties such as an upper bound and optimality in some particular cases. These features are provided in Sect. 4.

4 Theoretical analysis of Algorithm G3

In this section, some interesting properties of Algorithm G3 are analyzed.

4.1 An upper bound for the approximation

Theorem 1 *For an instance of the INTERNET SHOPPING PROBLEM with n products and m shops, with an optimal solution of cost c^* , and $\forall i \in M$, $\forall x \in [0, \sum_{i \in M} \max_{j \in N_i} p_{ij}]$, $f_i(x) \geq \beta x$, the cost of the solution suggested by G3 (Algorithm 3) c_{G3} is such that*

Algorithm 3 G3

Require: Number of shops m
Require: Number of products n
Require: Products available in shop i , N_i
Require: Functions f_i
Require: Prices p_{ij} and $p_{\max}(j)$
Require: Delivery fee d_i
Ensure: Array of baskets S

```

1:  $N := \{1, \dots, n\}$ 
2: while  $N \neq \emptyset$  do
3:   for  $i = 1$  to  $m$  do
4:      $C_i := \emptyset$ 
5:      $val_i := 0$ 
6:      $\forall j \in N \cap N_i, r[j] := V_i(\{j\}, S[i])$ 
7:     Select  $k$  such as  $r[k] = \max_{j \in N_i \cap N} r_j$ 
8:     repeat
9:        $C_i := C_i \cup \{k\}$ 
10:       $val_i := r[k]$ 
11:       $\forall j \in N \cap N_i \setminus C_i, r[j] := V_i(C_i \cup \{j\}, S[i])$ 
12:      Select  $k$  such as  $r[k] = \max_{j \in N_i \cap N \setminus C_i} r_j$ 
13:    until  $N \cap N_i \setminus C_i = \emptyset$  or  $r[k] < val_i$ 
14:   end for
15:   Select  $k$  such as  $val_k = \max_{i \in \{1..m\} \setminus \{C_i \neq \emptyset\}} val_i$ 
16:    $S[k] := S[k] \cup C_k$ 
17:    $N := N \setminus C_k$ 
18: end while
19: return  $S$ 

```

$$c_{G3} \leq (n-1) \max_{i \in M} d_i + \frac{c^*}{\beta}$$

Before detailing the proof, we informally describe the structure of the worst case. Since each shopping basket is constructed iteratively, there can be instances where buying two objects does not improve the ratio V_i . In these instances, the algorithm may pick at each step a single object in a new shop where this object has a lower price than everywhere else (including delivery cost).

The optimal strategy might be to buy everything from a single shop where the value for a basket of any single item is slightly worse than where the algorithm picked it, but where the discounting function is such that there is a big discount when everything is bought in this shop. This leads to the bound, as the left term is the expression of the fact that this algorithm could use many different shops, while the right term shows that the optimal solution could use a very good discount which could be completely missed by the algorithm.

Proof Let S^* be an optimal solution.

Let us consider the k th iteration of G3; for each previous iteration l , the products $C^{(l)}$ were chosen from shop $i^{(l)}$, so all that remains is to find a shop for the products $N^{(k)} = N \setminus \bigcup_{1 \leq l < k} C^{(l)}$.

Let $B_i^{(<k)} = \bigcup_{q < k \mid i^{(q)} = i} C^{(q)}$ be the basket of items selected in shop i before the k th iteration.

$C_i^{(k)}$ is the set of items selected from shop i at the k th iteration; $C^{(k)}$ is the one chosen at the end of this iteration in shop $i_{\max(k)}$. To simplify the notations while the work is

done on the k th iteration, the (k) is omitted:

$$\begin{aligned} C_i^{(k)} &= C_i \\ B_i^{(<k)} &= B_i \\ i_{\max(k)} &= i_{\max} \\ \forall i \in M, V_{i_{\max}}(C_{i_{\max}}, B_{i_{\max}}) &\geq V_i(C_i, B_i). \end{aligned}$$

Given $i \in M$, let x_1 be the first element taken to build C_i :

$$\begin{aligned} - V_{i_{\max}}(C_{i_{\max}}, B_{i_{\max}}) &\geq V_i(\{x_1\}, B_i) \\ - \forall x \in N_i \cap N^{(k)}, V_i(\{x_1\}, B_i) &\geq V_i(\{x\}, B_i). \end{aligned}$$

Let $i^*(x)$ be the shop in which x is bought for the optimal solution, i.e., $x \in S_{i^*(x)}$.

So,

$$\begin{aligned} \forall x \in C_{i_{\max}}, V_{i_{\max}}(C_{i_{\max}}, B_{i_{\max}}) \\ \geq V_{i^*(x)}(\{x\}, B_{i^*(x)}) \end{aligned} \quad (4)$$

and

$$\begin{aligned} V_{i_{\max}}(C_{i_{\max}}, B_{i_{\max}}) \\ \geq \frac{\sum_{x \in C_{i_{\max}}} p_{\max}(x) V_{i^*(x)}(\{x\}, B_{i^*(x)})}{\sum_{x \in C_{i_{\max}}} p_{\max}(x)}, \end{aligned} \quad (5)$$

i.e.,

$$\begin{aligned} f_{i_{\max}}(C_{i_{\max}} \cup B_{i_{\max}}) - f_{i_{\max}}(B_{i_{\max}}) \\ \leq \sum_{x \in C_{i_{\max}}} f_{i^*(x)}(x). \end{aligned} \quad (6)$$

Let q be the total number of iterations of $G3$ on this instance,

$$\begin{aligned} \sum_{k=1}^q f_{i_{\max(k)}}(C_{i_{\max(k)}}^{(k)} \cup B_{i_{\max(k)}}^{(<k)}) - f_{i_{\max(k)}}(B_{i_{\max(k)}}^{(<k)}) \\ \leq \sum_{k=1}^q \sum_{x \in C_{i_{\max(k)}}^{(k)}} f_{i^*(x)}(x) \end{aligned} \quad (7)$$

$$\begin{aligned} c_{G3} &\leq \sum_{k=1}^q \sum_{x \in C^{(k)}} f_{i^*(x)}(x) \\ c_{G3} &\leq \sum_{k=1}^q \sum_{x \in C^{(k)}} d_{i^*(x)} + p_{i^*(x)} x \\ c_{G3} &\leq \sum_{x \in N} d_{i^*(x)} + p_{i^*(x)} x \\ c_{G3} &\leq (n-1) \max_{i \in M} d_i + \frac{1}{\beta} c^*. \end{aligned}$$

4.2 Optimality in a simplified case

In the case where all shops have all the products and offer them at the same price, the problem is solvable in polynomial

time by a specific algorithm in $O(n+m)$ [Blazewicz et al. \(2014\)](#), and under those assumptions $G3$ also provides an optimal solution.

Theorem 2 If $\forall i \in M, N_i = N$ and $\forall j \in N$ and $(j, j') \in M^2, p_{ij} = p_{ij'}$ then $G3$ provides an optimal solution.

Proof Given $i \in M, E \subset N$ and $j \in N \setminus E$, set $s = \sum_{k \in E} p_{ik}$; thus, using the fact that g_i is concave and nondecreasing and that h_i is decreasing, we have

$$\begin{aligned} V_i(E) - V_i(E \cup \{j\}) \\ \leq \frac{g_i(T_i(E) + p_{ij})}{s + p_{ij}} + \frac{g_i(T_i(E))}{s} + \frac{h_i(s + p_{ij})}{s + p_{ij}} - \frac{h_i(s)}{s} \end{aligned} \quad (8)$$

$$\begin{aligned} V_i(E) - V_i(E \cup \{j\}) \\ \leq \frac{g_i(T_i(E)) + \frac{p_{ij}}{s} * g_i(s) - \frac{s + p_{ij}}{s} * g_i(T_i(E))}{s + p_{ij}}, \end{aligned} \quad (9)$$

i.e.,

$$V_i(E) - V_i(E \cup \{j\}) \leq 0.$$

Therefore, for all subsets E of N , for any element $x \in N \setminus E, V_i(E \cup \{x\}) \geq V_i(E)$. Following the process of $G3$, at the first iteration, all the products are selected in each shop to maximize $V_i: \forall i, C_i = N$.

Let $i_{\max}$ be the value of i that maximizes $V_i(C_i)$.

$$\begin{aligned} \forall i \in M, V_{i_{\max}}(C_{i_{\max}}) &\geq V_i(C_{i_{\max}}) \\ \forall i \in M, V_{i_{\max}}(N) &\geq V_i(N) \\ \forall i \in M, f_{i_{\max}}(N) &\leq f_i(N) \end{aligned}$$

Yet, with such a constrained problem, it has been proven that all products are bought from only one shop ([Blazewicz et al. 2014](#)). Accordingly $G3$ returns an optimal solution. $\square$

5 Benchmark suite with instances generator

A challenging step in the experimental research was to create a model that would be as close to real Internet shopping conditions as possible. We studied the relationship between competitive structure, advertising, price, 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 virtual (Internet) stores and frequency of purchase through this kind of shopping channel. We adopted some information and computational results from [Pathak \(2012\)](#), [Ratchford et al. \(2003\)](#), [Pan et al. \(2003\)](#), and [Clay et al. \(2001\)](#) for our model. It focuses mainly on electronic bookstore (we focused mainly on books, CDs with music, DVDs with movies), price definition models, acceptance factor, retailer brand ([Chu et al. 2005](#)), and (what is important for the optimization problem model definition) price dispersion. One should also notice

that consumers may choose from a large number of Internet bookstores. Yahoo.com lists more than 100 online bookstores. Data for our sample were collected from 32 stores and covered the largest US-based stores, including Amazon, BarnesandNoble.com, Borders.com, Buy.com, and Booksamillion, and 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 working model was prepared on the basis of data from the above-mentioned publications as well as our own observations of many Internet stores. We assume that all the tests should be done for a number of stores $m \in \{20, 30, 40\}$, with a given number of products in the basket. It is assumed that each store 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 : $ref_j \in \{2, 4, \dots, 100\}$ 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. The price of a product j in store 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:

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

Every shop is connected with a delivery fee, taken arbitrarily between 0 and 20, and a discount function is generated with the following observations:

- On some websites we can notice that, if more than 50 € worth of products is bought, half of the delivery cost is deducted from the total, and then if the total cost is greater than 100 € delivery is free. To generate similar situations, the delivery fee can be deducted at three times at most: if the shipment cost is given at k times, then there are

k thresholds, and a k th of the shipment fee is deducted every time.

- Moreover, websites with discounts such as 5 % deducted if more than 50 € is taken and then 10 % if the basket is worth more than 100 € are common, so the model takes into account a discount up to 10 %, applied step by step.

These two strategies can be combined together, and all thresholds used to generate data are multiples of 25 to be as close as possible to the observations of real websites.

6 Experiments

As most price comparison websites only compare prices from 25 to 30 shops, m is taken in $\{20, 30, 40\}$; likewise, an average customer will only buy a few books at the same time, so n is taken in $\{2, 3, 4, 5, 6, 7\}$.

These choices also take into account the fact that the search for an optimal solution is exponential in time. To study how the algorithms react for bigger instances, for n from 5 to 100 in steps of 5, the heuristics $G1$, $G2$, PCS , and $PCS+$ are later compared with the results of $G3$ instead of being compared with an optimal solution. For every experiment, 400 instances of the problem are used.

7 Results

7.1 Sensitivity of $G1$ to the order of items

$G1$ was tested with two different strategies: taking items in order of nonincreasing reference price and in the opposite order. Using the descending price order provided on average better solutions than the ascending order (Fig. 1). Furthermore, $G1$ also provides approximated solutions within 2.5 % of the corresponding optimal solution. This shows that a pure greedy algorithm can provide a good approximation in a short time: $G1$ is $O(nm)$, but sorting the products also takes at least $O(n \ln n)$.

7.2 Sensitivity of $G2$ to its parameter

To establish the sensitivity of $G2$ to its parameter α , it was tested with four different values: 25, 50, 75, and 100 %.

The minimum value from the four tests was also taken into account.

The results suggest that $G2$ needs some fine-tuning to be able to provide good approximations (Fig. 2): even the best solution out of the four parameters exhibited a difference ratio greater than 5 % for a low number of items ($n = 4$). The choice of the parameter α has a great impact on the performance of $G2$, so it has to be chosen carefully.

Fig. 1 Differences between G1 following different strategies and an optimal solution for $m = 30$ (with 95 % confidence intervals)

Fig. 2 Differences between G2 using different parameters and an optimal solution for $m = 30$ (with 95 % confidence intervals)

Fig. 3 Differences between G1, G2, G3, and PCS+ and an optimal solution for $m = 20$ (with 95 % confidence intervals)

7.3 Results for G3

For every instance (n, m) of the problem, PCS+ and G3 were tested in regard to G1 with the descending prices strategy and then compared with an optimal solution.

As G2 needs fine-tuning, the best solution provided by G2 using the parameters 25, 50, 75, and 100 % was taken.

PCS was omitted, because it provided solutions that were at least 15 % worse than the optimal solution.

G3 provides solutions that are at most 0.59 % worse than the optimal solution; it is the algorithm that suggests the best solution out of all the tested heuristics (Figs. 3, 4, 5). G1 also provides good results, but all its solutions but two were on average at least 0.59 %

Fig. 4 Differences between G1, G2, G3, and PCS+ and an optimal solution for $m = 30$ (with 95 % confidence intervals)

Fig. 5 Differences between G1, G2, G3, and PCS+ and an optimal solution for $m = 40$ (with 95 % confidence intervals)

worse than the optimal solution; it is also better than PCS+.

The ratio for G3 tends to increase as the number of items increases, but much more slowly than for PCS+ or G1. Moreover, G2 is highly dependent on the number of shops: for $m = 20$, the difference ratio between G2 and an optimal solution does not exceed 7 % for n in $\{2, 3, 4, 5, 6, 7\}$, but for $m = 40$ and $m = 30$, it is exceeded for $n = 6$. G3 provides good solutions close to the optimal one and better than those provided by previous algorithms.

7.4 Scalability of G3

To study the behavior of the algorithms on bigger instances, G1, G2, PCS, and PCS+ were run and compared with G3 instead of being compared with an optimal solution. Without searching for an optimal solution, n could be much larger than in the previous tests so n was now taken in $\{5, 10, \dots, 95, 100\}$, while m does not change.

G3 provided better results than other heuristics even with large numbers of products (Figs. 6, 7, 8): G1 produced solu-

tions at least 0.66 % bigger, while those from PCS+ were at least 2.5 % bigger.

These gaps quickly increased, and for n bigger than 20, the difference between G1 and G3 was at least 1.3 %. The solutions provided by PCS became better as the impact of delivery fees on the total cost decreased when the number of items increased.

Besides, PCS provides even better results than PCS+ when n increases.

G3 provides better solutions than the other algorithms even with larger numbers of items.

7.5 Summary of the comparison

G3 provides better solutions than G1, G2, and the algorithms commonly used on price comparison websites, but it also deals better with situations where not all items are available on every website, making it more versatile. Nonetheless, G3 has a higher cost in complexity ($O(mn^3)$) than G1 ($O(nm)$).

ISOP is a new formulated problem, fresh in the electronic commerce world. According to our best knowledge

Fig. 6 Differences between G1, G2, PCS, and PCS+ and the solution from G3 for $m = 20$

Fig. 7 Differences between G1, G2, PCS, and PCS+ and the solution from G3 for $m = 30$

Fig. 8 Differences between G1, G2, PCS, and PCS+ and the solution from G3 for $m = 40$

and deep review of state-of-the-art papers, we could not find any already defined problem that could be used as a transformation target from ISOP. Therefore, it is not possible to compare our results with other authors. Consequently, we focus on algorithm development. As an additional comparison, we used PCS and PCS+ algorithms, which are known engines for price comparison sites. However, as could be expected, these solutions were not dedicated to the ISOP problem, and

the results obtained from these algorithms were much worse than those obtained by the newly developed ones.

8 Conclusions

E-commerce is an industry which focuses on selling and buying products and services through web pages (see Hagel III 1999; Timmers 1998). One of the very important topics of

research in this area is, motivated by practical applications, Internet shopping, which is becoming increasingly popular each year.

Herein we introduce the ISOP including two different discounting functions, namely a shipping cost function as well as a price discounting function. A new algorithm denoted as G3 is introduced to provide solutions to this newly defined problem. An upper bound is proved, as well as the optimality in a particular case where the problem is solvable with a polynomial algorithm. Moreover, other known algorithms were coded to determine the quality of the new algorithm G3 using computational experiments.

An advanced benchmark was created to perform exhaustive computational experiments. It is worth noticing that a new, more realistic world model was proposed. Results of these experiments demonstrated that G3 provides better solutions than existing heuristics, even when items may be unavailable.

The computational experiments used in this work were based on a model that was made to be as realistic as possible, but the next challenging step for this subject would be to test the algorithms on real data collected from online stores.

The succeeding step will be preparation of an exhaustive, vast computational experiment (if possible already based on real data from online stores) engaging all known algorithms. However, to prepare a real big comparison, we plan to check the performance of mathematical programming optimizers. We plan to solve the ISOP in its mathematical programming formulation using optimizers such as CPLEX, Gurobi, LINDO, SCIP, and maybe others.

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

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

- Bastnagel, M., Berk, B., Burkard, A., Carpenter, M., Christen, H., & Codd et al. (2003). Systems and methods of comparing product information. US Patent App. 09/728,583
- Blazewicz, J., Bouvry, P., Kovalyov, M. Y., & Musial, J. (2014). Internet shopping with price sensitive discounts. *4OR - A Quarterly Journal of Operations Research*, 12, 35–48.
- Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Weglarz, J. (2007). *Handbook on scheduling: From theory to applications*. Berlin, New York: Springer.
- Blazewicz, J., Kovalyov, M., Musial, J., Urbanski, A., & Wojciechowski, A. (2010). Internet shopping optimization problem. *International Journal of Applied Mathematics and Computer Science*, 20(2), 385–390.
- Blazewicz, J., & Musial, J. (2011). E-commerce evaluation: Multi-item internet shopping. optimization and heuristic algorithms. In B. Hu, K. Morasch, S. Pickl, M. Siegle (Eds.), *Operations Research Proceedings 2010, Operations Research Proceedings*, pp. 149–154. Springer, Berlin Heidelberg
- Christensen, E. (2010). Method and system for aggregating, standardizing and presenting purchase information from shoppers and sellers to facilitate comparison shopping and purchases. US Patent 7,693,750
- Chu, W., Choi, B., & Song, M. (2005). The role of on-line retailer brand and infomediary reputation in increasing consumer purchase intention. *International Journal of Electronic Commerce*, 9, 115–127.
- Clay, K., Krishnan, R., & Wolff, E. (2001). Prices and price dispersion on the web: Evidence from the online book industry. NBER Working Papers 8271, National Bureau of Economic Research Inc.
- Eiselt, H., & Sandblom, C.-L. (2004). *Decision analysis, location models, and scheduling problems*. Berlin: Springer.
- Hagel, J. I. I. (1999). Net gain: Expanding markets through virtual communities. *Journal of Interactive Marketing*, 13(2), 55–65.
- Iyigun, C., & 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., & Plastriab, F. (2002). Discrete location problems with push-pull objectives. *Discrete Applied Mathematics*, 123, 363–378.
- Krulwich, B. (1996). The bargainfinder agent: Comparison price shopping on the internet. In J. Williams (Ed.), *Agents, bots, and other internet beasts* (pp. 257–263). Indianapolis: Sams. Net Publishing.
- Leung, J., Kelly, L., & Anderson, J. H. (2004). *Handbook of scheduling: Algorithms, models, and performance analysis*. Boca Raton, FL: CRC Press Inc.
- Melo, M., & Nickel, S. (2009). Facility location and supply chain management: A review. *European Journal of Operational Research*, 196(2), 401–412.
- Musial, J. (2012). *Applications of combinatorial optimization for online shopping*. Poznan: NAKOM.
- Pan, X., Ratchford, B., & Shankar, V. (2003). The evolution of price dispersion in internet retail markets. *Advances in Applied Microeconomics: Organizing the New Industrial Economy*, 12, 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., & 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., & Daskin, M. (2008). A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research*, 184(3), 817–848.
- Timmers, P. (1998). Business models for electronic markets. *Electronic Markets*, 8(2), 3–8.
- Wojciechowski, A., & Musial, J. (2009). A customer assistance system: Optimizing basket cost. *Foundation Of Computing And Decision Sciences*, 34(1), 59.
- Wojciechowski, A., & Musial, J. (2010). Towards optimal multi-item shopping basket management: Heuristic approach. In R. Meersman, T. Dillon, & P. Herrero (Eds.), *Lecture notes in computer science* (Vol. 6428, pp. 349–357). On the move to meaningful internet systems: OTM 2010 workshops Berlin/Heidelberg: Springer.