A CUSTOMER ASSISTANCE SYSTEM: OPTIMIZING BASKET COST

Adam WOJCIECHOWSKI, Jedrzej MUSIAL*

Abstract. A strong advantage of buying products in on-line shops is a wide choice of alternatives. From customers' point of view it is good to know where to go in order to buy all required products – the entire shopping list – at the best possible price. Making such decisions requires three elements: information where the required products are available, access to price lists in considered shops, and finally, specialized analytical tool that could find the minimal subset of shops where all the products from the customers' shopping list could be bought at the lowest price. In the paper we propose the *mini-mini* algorithm and describe 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 price total. The same algorithms might be applied to other applications optimizing Internet shopping where the cost of basket delivery is negligible when compared to product prices.

Keywords: virtual shopping assistant, basket cost optimization, mini-mini algorithm.

1. Introduction

Shopping on-line became a popular practice. A survey concerning Polish Internet users behaviour published by Gemius S.A. in June 2008 [1] shows that population of on-line customers grows rapidly and systematically from year to year. While in 2005, 41% of Internet users in Poland had some experience in buying products in virtual shops, the ratio grew to 55% in 2006 and reached 66% in 2007. The development of on-line shopping is also stimulated by the increasing number of Internet users. All those factors brought growth of the market of on-line shopping by more than 60% year to year in Poland in 2007. A survey concerning behaviour of customers in America [2] confirms the tendency observed in Poland, however the numbers are even higher.

Auctions [3][4] and on-line stores are key business activities offered over the Internet. A systematically growing number of customers is the energy that stimulates new merchandisers to provide functionality of on-line shopping [5]. A wide choice of alternative

* Poznan University of Technology, ul. Piotrowo 2, 60-965 Poznan, Poland

providers increases competition on the market and forces low prices [6]. All of the aforementioned processes are good from the customers' point of view. However, a growing number of shops, which seem to be at the same distance from the customer's perspective (a click away) rises a new problem: shop and product selection. A common solution of the problem comes from the so-called price comparators – recommender system [7] applications equipped with a database of offers fed by on-line shops with the ability to search and sort required products by price. 36% of Polish Internet users who practiced on-line shopping, used price comparators. Unfortunately price comparison works only on a single product and if the customer's basket is composed of several products complete shopping list optimization needs to be done manually.

In the paper we provide a short review of functionality of selected price comparators (Section 2). In order to compare price comparators we introduce a quality measure that may provide a general assessment whether a given price comparator is better or worse than the competition (Section 3). In Section 4 we pay special attention to find weak elements of the price comparison sites. In Section 5 we propose new approach to the on-line shopping problem (a kind of decision support system) and presents the *mini-mini* algorithm and a prototype application ready to be used in optimizing customer's shopping list containing many products. Applicability of proposed solution and discussion on further development is based on experimental results.

2. Price comparison

Shopping is a decision process. However, the process may have two different starting points:

- *intentional shopping*, when a customer knows the products he wants to buy, and optimization is reduced to searching for a supplier where the products can be obtained at the best price (delivery costs may be taken into account when the basket cost is considered); this shopping paradigm may be supported well by price comparators;
- *looking for a bargain*: this model of customer's behaviour describes situations when the customer cannot ask a price comparator about the best prices for particular product, because he does not initially know the product; this shopping paradigm may be supported by recommender systems (RS). The effectiveness of RS is quite high. It is claimed that even 20% of customers may be encouraged to purchase a product by successful recommendation [8]. Some steps towards conceptual merging price comparator, recommender system and social price watch system were proposed in [9].

Wide competition among on-line shops leads to substantial price reductions. The range of prices of the same product may reach as high as 50% or even more. Thus, the will to search for a better price of a known product is quite natural and economically justified. On the other hand, it is often impossible to visit dozens of on-line shops, review the offers, delivery conditions, etc. and choose the best option. Even if we assume that the customer has complete knowledge (e.g. addresses) of all the shops providing the required product and he has the ability to search the offers manually, the effort may cost more than savings achieved by buying the cheapest alternative. In a particular case customer may have the luck to visit the cheapest shop in his first move. Following visits to other shops bring no economical return, because the best offer was found in the first step. The customer's gain in this case was in increased confidence. The more offers are reviewed the better confidence is attributed to the best option found so far. The observation given above forms the basis for quality measures that can be used to compare price comparators.

3. Quality Metrics to Assess Offers Found by Price Comparison Applications

A typical price comparator application offers functionality of displaying offers of products whose name or description matches a text pattern specified in the query string. The results are ordered by price. Although the functionality allows users only to optimize single product shopping, the services are very popular. Another weakness of price comparators is the focus on on-line shops. In general, information about price offers in conventional shops is not available on-line and price comparison applications do not take those offers into account while presenting the cheapest options referring to required product.

Quality measure of solutions (offers) proposed by a price comparator may be different when assessed from the perspective of a customer or seller. The customer's indicator of quality of the best offer found by a price comparator may be the difference between average market price and the best offer pointed to by the price comparator. The indicator may be calculated in percentage terms when related to the average market price (1).

$$Q_K = \frac{c_{mkt} - c_p}{c_{mkt}} [\%] \tag{1}$$

where c_{mkt} average market price for observed product
c_p best (lowest) price for observed product found by price comparator

If equation (1) would be applied to all products indexed by tested price comparator the average quality measure may be considered as a quality indicator for price comparator (2).

$$\overline{Q}_K = \frac{\sum_{i=1}^{n} \frac{c^i_{mkt} - c^i_p}{c^i_{mkt}}}{n} [\%] \tag{2}$$

where c^i_{mkt} average market price of i-th product
c^i_p best (lowest) price of i-th product found by price comparator
n number or products indexed by price comparator

A separate problem, intentionally skipped in this paper, is how to calculate average market price for products indexed by a tested price comparator. Some approximation may come from the price comparator itself. Offers clicked (selected) by price comparator users are often finalized by purchase. Counting the average price of all clicked offers for particular product, a price comparator may collect good enough approximation of the market price.

Utility of price comparator assessed from seller's perspective is quite different. It is in the merchandiser's interest to persuade customers to his own offers. The better the offer contrasts to the competitors the more likely the customer will buy the product. Moreover, the seller knows that after he encourages customers to visit the shop he may use his own recommender techniques and lead the customer to offers in his shop. Taking the above observations into account we may formulate the following quality metrics that base on the information provided by the merchandiser:

- buy/visit ratio; how many visits directed from particular price comparator are finalized with a buy transaction;
- average position in price comparator ranking list after executing queries with names of products offered by a given seller;
- percent of products offered by a given seller that appears on the top position in price ranking constructed by price comparator (in this measure we assume that customers are most likely to choose the cheapest offer);
- average transaction net profit generated by a customer visit directed from a price comparator.

The indicators given above are only examples of quality and utility measures that can be used to assess price comparator system from a merchandiser's perspective.

4. Weak points of price comparison applications

An evident weakness of currently used price comparison applications is a lack of support for managing multiple-item basket optimization. So, the customer cannot place several products in a basket and ask a system for advice on how to divide the basket into sub-baskets which may be bought in different shops in order to have all the products bought and delivered at the minimal price. The reason why such a service is not available is the computational complexity of the optimization process and often a lack of sure information whether the product described in shop A as "yellow pencil" is the identical product to the one named "yellow pencil" offered by shop B. Solving the optimization problem requires not only the information about product names (descriptions) and price; in order to effectively find equivalent products offered by various suppliers the application needs an attribute which could be used to match identical products. In many cases it could be enough to use a barcode as a universal product identifier, good enough to assess if similar goods offered by two suppliers are, in fact, the same product. A short survey among on-line stores gave us an answer that on-line shops are not prepared for that kind of service because very few of them use barcodes to identify the products in their offer.

Aside from the technical drawback we may also consider organizational issues that keep many merchandisers away from price comparators. It is a general rule that price comparator operators charge some money from the sellers, either for submitting their shop offer to the system, or for redirecting customers from the price comparator to particular on-line shop. The payment is worth to be paid if the offer is placed on top positions in price ranking produced by price comparator, and it is likely that potential customers will choose the offer and be redirected to a particular shop. If an offer is not on top positions in price ranking it is very unlikely that customers would be interested in such an offer. Price ranking based on single criterion – the lower price, the better offer –force sellers to reduce their margins to a few percent to have a chance to be on the top of the ranking. The simple method of comparing offers leaves no space for information about quality of service provided by the sellers. The problem has been noticed in some price comparators and it is currently becoming a common practice that customers may feed the database with their assessment and impressions of on-line shop presented by the system. The low-margin problem is important for merchandisers, because, according to research, reported in 2007 by Mediarun.pl, even 85% of on-line shops may be not indexed by price comparators.

Although the current state of price comparators is rather primitive those systems are considered to be among the most promising web applications.

5. An approach to improve the functionality of price comparison sites

Facing the databases of price comparators used nowadays, we should agree that almost all information required for shopping optimization is collected there. However, as we mentioned before, in many applications products are not identified by manufacturer's code (e.g. barcode) but identical products are recognized because they are identically named and described in offer records coming from different shops. We find this method not sufficient for tracing identical products offered by different providers, and thus, we assume that products in all offers collected in price comparator database must be identified by a universal code. In that were the case, the optimization application could recommend a change in the customer's basket, replacing product *p* offered by shop A with an identical product *p* offered by shop B. The rule of using universal identifiers applies to all products whose quality and value does not depend on the point of sale (e.g. the same book bought in Poznan or in Paris, or an airplane ticket bought on-line in an electronic form is equivalent to the ticket for the same flight bought in material-paper form). We can also assume that we allow collecting offers from on-line shops as well as conventional shops. Moreover we can register physical shop locations to give the customer a chance to visit the shop personally (if he accepts the distance) and buy required products without delay of delivery by post. To simplify conditions of the experiment we can also assume that personal collection of items from point of sale carries no extra cost for the customer. Taking the above conditions together we need a database consisting of several components:

- a catalogue of products (barcode + other descriptive attributes, not important in optimization process)
- price offers from shops indexed by the basket optimizer. Each price offer is a triple: (shop_id, product_barcode, price);
- information about shops (shop_id, shop_GPS_position, other descriptive attributes)

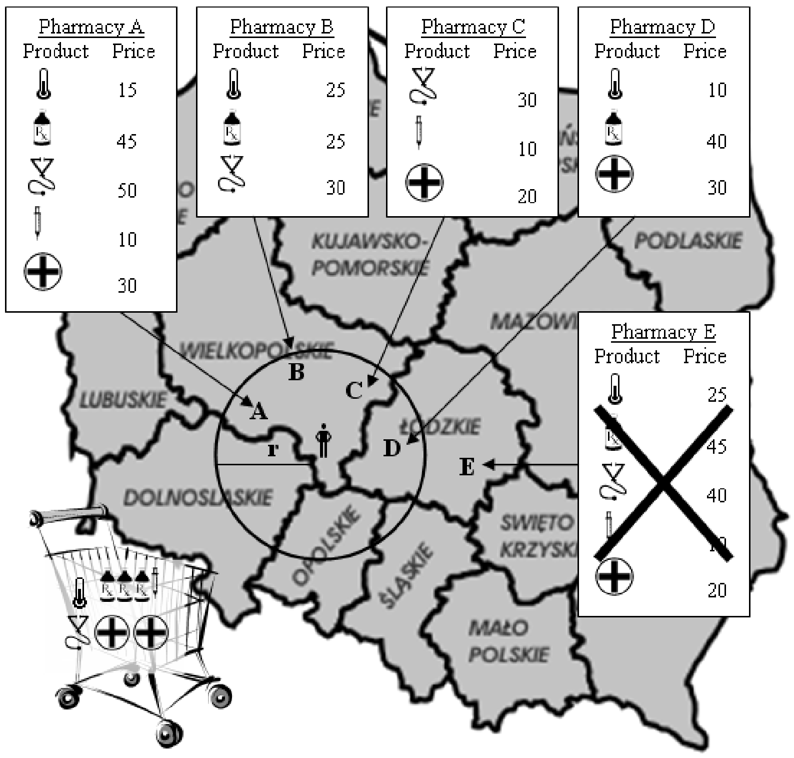

Figure 1. Basket optimizer database content and initial shop selection process.

The structure proposed above guarantees that each product that appears in a price offer of any shop is stored in the product catalogue. A customer who knows his current geographical position may decide about range where he can go to buy required products personally. Shops located outside the user-defined range may be taken into account during basket optimization only if the customer allows on-line shopping and the shop offers on-line sale. For demonstration use we assumed that delivery cost is negligible, is not calculated. Applying the algorithm to pharmacy market we are close to reality, where customers need to visit a particular pharmacy to buy prescribed medicine, which is not available in on-line shopping with delivery. We also assume that a barcode uniquely identifies each product, however we know that in some product families (e.g. clothes) it would be necessary to extend attribute list by size, colour etc. in order to precisely express customers requirements and allow accurate product replacement (in fact it is change of provider) in customer's basket. Information collected in basket optimizer database is shown on fig. 1, where unique barcodes are represented by icons. Identical product icons appearing in offers of different shops depict identical products.

6. Multiple-item shopping list division into minimal number of baskets satisfying the condition of minimal cost of buying all the products with zero delivery charge – *mini-mini* algorithm

The aim of the algorithm is to propose such a division of non-empty customer's shopping list into minimal number of baskets where each basket is fully realized in a single shop that realization of all baskets guarantees minimal cost of the entire shopping. In other words the logic of the *mini-mini* algorithm corresponds to the behaviour of a thrifty consumer, who would like to buy all the products, pay as little as necessary, and, in the best case, do all his shopping in one location. Although the main optimization criteria is minimal cost of entire shopping realization, the *mini-mini* algorithm works in dialog mode with the customer and may ask if the customer can accept increasing number of shops to visit in order to save money. Savings potential is known on every step of the optimization process.

Products on customer's shopping list are represented by barcodes. Customer's preferences about shops, where he can go for shopping come firstly from the customer's location $\boldsymbol{(x_0, y_0)}$ and the defined range $\boldsymbol{r}$ within which potential supplier must be located. On this basis the algorithm selects a set of shops $\boldsymbol{S}$ which are taken into account as potential shopping locations. Products on the customer's shopping list must be within a superset of all product offers of shops in $\boldsymbol{S}$. In other words, every product on the shopping list must be available in at least one shop in $\boldsymbol{S}$.

Lower limit of cost at which entire shopping list can be bought, $\mathbf{c_{opt}}$, may be computed by simple queries to database asking for minimal price among offers from shops in $\boldsymbol{S}$ for each product on shopping list: $\boldsymbol{min\ p_{is}}$, where $\boldsymbol{p_{is}}$ is price of product $\boldsymbol{i}$ in shop $\boldsymbol{s \in S}$. Sum of all $\boldsymbol{min\ p_{is}}$ multiplied by the quantity of product $\boldsymbol{i}$ on the shopping list: $\boldsymbol{q_i}$ counted for all products on the shopping list gives the optimal cost of buying all products, but it says nothing on how many shops must be visited to do the shopping at this cost.

In the next steps the algorithm does the optimization. First, *mini-mini* searches for an optimal realization of the shopping list in one shop ($\mathbf{n}$=1). If the entire shopping list can be realized in a single location (the set of realizations is not empty), the best (cheapest) realizations in a single shop are presented to the customer, and the customer knows the difference between the best price for the whole of the shopping in a single shop: $\mathbf{c_{1opt}}$ and the optimal realization (in any number of shops) $\mathbf{c_{opt}}$ computed previously. If the customer is satisfied with the solution obtained at this stage, he stops the computation; otherwise the algorithm increases the number of shops for shopping realization: $\mathbf{n}$ and repeats the procedure of searching for an optimal shopping realization within n shops. The algorithm stops either on the customer's request or when an optimal realization of shopping list $\mathbf{L}$ at price $\mathbf{c_{opt}}$ is found. $\mathbf{L}$ is defined to be the set of products selected from superset of all offers from shops in $\mathbf{S}$. The quantity of product i on list L is identified in the algorithm by q_i.

Realization of a shopping list $\mathbf{L}$ in shops $s_1, s_2, \ldots, s_n$, $s_i \in \boldsymbol{S}$ for i=1, …, $|\boldsymbol{S}|$ is such a set of triples: {(shop$_x$, product$_y$, quantity$_z$)} that all products from shopping list $\mathbf{L}$ appear in the set, sum of quantities of particular product assigned to shops in realization equals quantity of that product on shopping list, every shop in realization is in set $\mathbf{S}$ (accepted shops).

6.1. The mini-mini algorithm

```
1.  Define customer's position: $(x_0, y_0)$.
2.  Define range r: maximum distance to shopping location if distance criteria matters
     to the customer
     otherwise $r := \infty$
3.  Select set of potential shopping locations: S.
     e.g. geographical coordinates of shops in set S must comply to rule:
     distance between $(x_s, y_s)$ and $(x_0, y_0)$ is not greater than r, for each $s \in S$
4.  Build shopping list L.
5.  Calculate lower limit/optimal cost of purchase of all products on list L: $c_{opt}$
     $c_{opt} := \sum_{i=1}^{|L|} q_i \min(p_{is}),\ s \in S$
6.  $n := 1$
7.  $c_{n\ min} := \infty$
8.  $R := \varnothing$
9.  Select set R of the cheapest realizations of shopping list L in n shops.
     $c_{n\ min}$ := cost of the cheapest (optimal) realization of shopping list L in n shops
10. If set R (optimal realizations of L in n shops) is not empty
     then display R to the customer
11. If $c_{n\ min} = c_{opt}$ then go to 15.
12. If customer is satisfied with R (realizations of L at cost $c_{n\ min}$)
     then go to 15.
13. $n := n+1$
14. Go to 7.
15. STOP
```

In steps 1 to 4 of the *mini-mini* algorithm given above, the customer builds a shopping list from the products available in shops selected to be considered as potential shopping locations. This procedure is a guarantee that all products on shopping list **L** can be purchased and the algorithm stops, giving optimal solution. If we assume that a properly built shopping list **L** is given to the algorithm as an input parameter the logic of the optimization procedure should start at step 5.

6.2. Computational experiment

In order to verify the use of the algorithm in a practical application we prepared a prototype application [10] and ran a computational experiment in which we used to optimise shopping list consisting of between 2 and 25 products. The number of shops was gradually increased from 2 to 9. The experiment was performed on an IBM 1.6 GHz computer equipped with 1GB RAM running Windows XP.

Table 1. Mini-mini computation time [s] in function of number of products on shopping list and number of potential shopping locations (shops).

# shops	# products on shopping list 2	3	4	5	10	15	25	average comput. time
2	0.0037	0.0022	0.0027	0.0028	0.0027	0.0027	0.0034	0.0028
3	0.0055	0.0044	0.0045	0.0059	0.0055	0.0058	0.0065	0.0054
4	0.0085	0.0103	0.0107	0.0098	0.0120	0.0125	0.0154	0.0114
5	0.0192	0.0195	0.0207	0.0216	0.0274	0.0275	0.0323	0.0239
6	0.0494	0.0582	0.0538	0.0682	0.0636	0.0658	0.0768	0.0612
7	0.2161	0.2302	0.2390	0.2488	0.2603	0.2575	0.2870	0.2485
8	1.3402	1.3128	1.2792	1.4132	1.4336	1.4305	1.4418	1.3948
9	12.5353	12.7659	12.9186	12.9062	13.0959	12.9467	13.0271	12.9349

Experimental results (see table 1) clearly show that computational time is minimally influenced by the number or products on the shopping list, but strongly depends on the number of shops which are potential shopping locations. Considering on-line shopping optimization in a number of shops grater than 8 would require either using a heuristic algorithm or queuing the optimization task into batch processing system and sending optimal solution to the customer after computation [11]. A typical heuristic optimisation may produce a single result – recommended realization of the customer's shopping list. The weakness of such heuristic approach lies in the fact that the customer is not presented alternative solutions: how to buy entire shopping basket at total cost slightly higher than heuristic best realisation but in smaller number of shops. In the *mini-mini* algorithm we show the customer optimal realizations of his shopping list in increasing number of shops starting from in a single shop if all the products are available in at least one shop. Such an approach increases the customer's confidence when he can observe how much he can save on entire shopping list realisation by increasing number of shopping locations. It is worth to notice that the higher number of shopping locations the more effort must be paid to manage multiple orders. The additional shopping management cost (effort) should compared with savings coming from realisation of the customer's shopping list in increased shopping locations.

Our research over the Internet and literature study showed no similar applications to optimal management of a multiple-item shopping cart. Thus it is hard to compare our results to other algorithms or models. Decision support system built upon the *mini-mini* algorithm computes all possible shopping cart realizations among all k-combinations (each of size k) from a set of S shops, where k={1,…,S}. The system presents the customer all possible solutions (computed within a k-element combination from a set of S shops) of the problem ordered by price. The existing and commonly used price comparison sites offer much weaker functionality, which is restricted to a ranking of offers for a single product only.

Computational complexity analysis of proposed algorithm is a subject of our research developed in parallel to designing applications that utilize the business value of the multiple-item shopping list management.

7. Summary

Price comparison applications are very popular among Internet shoppers. However, quality of results provided by those applications is hard to assess and compare because the applications use various data feeds: they provide information on product prices in different sets of shops. In the paper we proposed an approach based on simple metrics which could provide information on the quality of service assessed from customer's and seller's perspective. Applying those metrics in an experimental comparison is a subject for further research.

One of the main drawbacks of price comparison applications is a lack of support for multiple-item basket optimization. In this research work we proposed the *mini-mini* algorithm, solving the optimization problem. Solution discussed in this paper is designed for a particular case of the shopping list optimization problem where every product from shopping list is available in at least one shop considered as a shopping location, and delivery costs are negligible. The performed experiment showed that solving the problem in condition where the number of shops is greater than 8 requires either using a heuristic approach to deliver an approximate solution on-line (with known distance from optimum) or the problem may be solved by queuing the optimization task into a batch processing system and sending the optimal solution to the customer after the computation.

References

[1] Gemius, S.A. E-Commerce in Poland, 2008, http://gemius.pl/pl/raporty/2008-06/03.

[2] Pew Internet & American Life Project. On-line Shopping, 2008, http://www.pewinternet.org/pdfs/PIP On-line Shopping.pdf.

[3] Klein S., The Emergence of Auctions on the World Wide Web, in: Shaw M. et al. (eds.) Handbook on Electronic Commerce, Springer-Verlag, Berlin-Heidelberg, 2000, 627-645.

[4] Vulkan N. The Economics of E-Commerce. A Strategic Guide to Understanding and Designing the On-line Marketplace, Princeton University Press, Princeton, New Jersey 2003, 149-178.

[5] Liang T.P., Huang J.S. An Empirical Study on Consumer Acceptance of Products in Electronic Markets: A Transactional Cost Model, Decision Support Systems, 21, 1998.

[6] Lee, H.G. Do Electronic Marketplaces Lower the Prices of Goods?, in: Communications of the ACM 41 (1), 1998, 73-80.

[7] Satzger B., Endres M., Kielssing W. A Preference-Based Recommender System in: Baukhnecht K. et al. (eds.), E-Commerce and Web Technologies, LNCS 4082, Springer-Verlag, Berlin Heidelberg, 2006.

[8] Klopotek M., Wierzchon S, et al., Map-Based Recommendation of Hyperlinked Document Collection in Bauknecht K. et al. (eds.), EC-Web 2006, LNCS 4082, Springer–Verlag, Berlin Heidelberg 2006, 1-10.

[9] Wojciechowski A., Supporting Social Networks by Event-Driven Mobile Notification Services in Meersman Z. et al. (eds.), OTM 2007 Ws, Part I, LNCS 4805, Springer–Verlag, Berlin Heidelberg 2007, 398-406.

[10] Virtual pharmacy shopping optimiser, www.free.linuxpl.com/pharmacist, April 2008.

[11] Czechowski J., Distributed System for Customer Basket Optimization, master thesis, supervisor: Wojciechowski A., Poznan University of Technology, Poland, 2007 [in Polish].

Received October, 2008.