Computers & Operations Research 41 (2014) 1–11

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

# An effective matheuristic for the capacitated total quantity discount problem

Daniele Manerba *, Renata Mansini

*Department of Information Engineering, University of Brescia, Italy*

ARTICLE INFO

Available online 29 July 2013

*Keywords:*
Supplier selection
Total Quantity Discount
Matheuristic
ILP refinement

ABSTRACT

New generation algorithms focus on hybrid miscellaneous of exact and heuristic methods. Combining meta-heuristics and exact methods based on mathematical models appears to be a very promising alternative in solving many combinatorial optimization problems. In this paper we introduce a matheuristic method exploiting the optimal solution of mixed integer linear subproblems to solve the complex supplier selection problem of a company that needs to select a subset of suppliers so to minimize purchasing costs while satisfying products demand. Suppliers offer products at given prices and apply discounts according to the total quantity purchased. The problem, known as Total Quantity Discount Problem (TQDP), is strongly $\mathcal{NP}$−hard thus motivating the study of effective and efficient heuristic methods. The proposed solution method is strengthened by the introduction of an Integer Linear Programming (ILP) refinement approach. An extensive computational analysis on a set of benchmark instances available in the literature has shown how the method is efficient and extremely effective allowing us to improve the existing best known solution values.



## 1. Introduction

Supplier selection problem is usually formalized in terms of two joint decisions concerning which suppliers should be selected and how much should be ordered from each of them. As supplier selection represents a critical task in any organization, the problem has been studied since long (Weber et al. [34]), and in the last few decades has gained increasing attention both in practice and theory mainly fueled by its web counterpart (see e.g. [22]).

Real procurement problems are usually complicated by side constraints taking into account both qualitative and quantitative factors. The introduction of discount policies and/or traveling costs are among the most complicating features of real procurement settings. In this paper we deal with a supplier selection problem where suppliers apply volume discounts depending on the total quantity ordered. The problem can be described as follows. Let $S:=\{1,\ldots,n\}$ be a set of suppliers, and let $K:=\{1,\ldots,m\}$ be a set of products. Each product $k\in K$ can be purchased in a subset $S_k \subseteq S$ of suppliers at a positive price $p_{ik}$, potentially different for each supplier $i\in S_k$. For each product $k\in K$ a positive integer demand $d_k$ is defined, and for each product $k\in K$ and each supplier $i\in S_k$ a positive quantity availability $q_{ik}$ is specified, such that $\sum_{i\in S_k} q_{ik} \geq d_k$.

Each supplier $i\in S$ defines a set $R_i=\{1,\ldots,r_i\}$ of $r_i$ consecutive and non-overlapping discount intervals $[l_i^r, u_i^r]$, where $l_i^r$ and $u_i^r$ are the minimum and the maximum number of product units, respectively, that have to be purchased to lay in interval $r$ of supplier $i$ (w.l.o.g. we assume that $\sum_{k\in K} q_{ik} \leq u_i^{r_i}$ and $l_i^1=0$ for each $i\in S$). A non-negative discount rate $\delta_i^r$ is associated to each interval $r\in R_i$ such that $\delta_i^{r+1} \geq \delta_i^r$, $r=1,\ldots,r_i-1$. The interval in which the total quantity purchased lies determines the discount applied by the supplier to the total purchase cost (*total quantity discount policy*). For the sake of simplicity we can convert the discount rates into non-increasing unit prices $p_{ik}^r > 0$, i.e. $p_{ik}^{r+1} \leq p_{ik}^r$, $i\in S_k$, $r=1,\ldots,r_i-1$. The Capacitated Total Quantity Discount Problem (CTQDP) looks for a subset of suppliers so that the total demand is satisfied at a minimum purchasing cost.

In the classical Total Quantity Discount Problem (TQDP) suppliers are assumed to have unlimited availability for offered products. The problem is introduced in Goossens et al. [18] where the authors show its $\mathcal{NP}$−completeness by reduction from the 3-Dimensional Matching Problem (proof is done assuming $l_i^1=0$ for all the suppliers) and demonstrate that it cannot be solved by a polynomial-time approximation algorithm with a constant ratio (unless $\mathcal{P}=\mathcal{NP}$). They also study four variants of TQDP that involve market share constraints, the possibility of buying more than products demand, a limited number of winning suppliers and multi-period scenarios. Finally, they propose a branch-and-bound algorithm based on a min-cost flow problem formulation on

* Corresponding author. Tel.: +39 0303715935.
*E-mail addresses:* daniele.manerba@ing.unibs.it (D. Manerba), renata.mansini@ing.unibs.it (R. Mansini).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.07.019

instances with up to 50 suppliers, 5 discount intervals and 100 products. Nowadays, all these instances can be solved to optimality by a standard commercial solver in a matter of seconds, ceasing to be challenging.

More recently, Manerba and Mansini [24] develop a branch-and-cut approach for the Capacitated TQDP exploiting simple polyhedral results. The authors define a new set of hard-to-solve benchmark problems including up to 100 suppliers, 5 discount intervals and 500 products. Instances are divided into two classes and can be downloaded at the page http://www.ing.unibs.it/~orgroup/instances.html. The largest ones of these instances have not been solved to optimality yet.

In the present paper we use a straightforward heuristic that takes advantage of the information provided by the linear programming (LP) relaxation of the original problem to construct an initial solution, and study a stand-alone matheuristic for the CTQDP based on the optimal solution of mixed integer subproblems. The framework of the algorithm is similar to a Variable Neighborhood Decomposition Search (VNDS) as described in Hansen et al. [20]. Iteratively, some features of a current feasible solution are kept fixed and a Mixed Integer Linear Programming (MILP) subproblem is formulated and solved exactly. Computing optimal solutions, although computationally intractable, can become a valid tool provided that the subproblem solved is reasonably small. The proposed method is named HELP2 since it extends an embrional procedure called HELP (Heuristic Enhancement from LP) proposed in Manerba and Mansini [24]. We also introduce an ex post improvement procedure based on an Integer Linear Programming (ILP) refinement that enhances the performance of the basic algorithm. Heuristics behavior has been tested on the benchmark instances described above.

The paper provides the following contributions. HELP2 and its variant including ILP refinement represent new matheuristics for the CTQDP. Computational experiments show that the methods perform effectively for a variety of problem instances, including those taken from a real world application. In particular, HELP2 finds the optimal solution in about 48% of benchmark instances and in the remaining ones the total average error is equal to 0.16% for a class of instances and to 1.18% for the other one, whereas HELP2 with the ILP refinement further reduces the total average error to 0.56% on the most difficult instances improving 10 best known values. The heuristics owe their strength to simple basic ideas that could be generalized also to other MILP problems presenting a similar combinatorial structure. Finally, from a practical point of view, while the importance of the CTQDP is widely recognized, no solution methods can be found in the literature to tackle this real problem. To the best of our knowledge, HELP2 represents the first stand-alone heuristic methods for the CTQDP and thus provides the first reliable tool for decision makers handling similar procurement problems.

The paper is organized as follows. Section 2 is devoted to the literature review, focused either on existing algorithms for procurement problems or on general matheuristics. In Section 3 we describe a mathematical model for the problem, including useful valid inequalities added to strengthen the LP formulation. In Section 4 the procedure used to find an initial feasible solution as well as HELP2 and its extension including the ILP refinement (called HELP2+) is analyzed in detail. Extensive computational results are described in Section 5. Finally, some conclusions are drawn in Section 6.

## 2. Literature review

We first review solution approaches proposed for procurement problems, then we discuss contributions on hybrid methodologies (matheuristics) for more general problems.

In the literature, supplier selection problems are mainly faced in terms of quality criteria (Dickson [12] identifies 23 different critical factors for selecting suppliers). In the last few years a part of the specialized literature was focused on mathematical modeling and heuristic approaches for procurement problems with deterministic or stochastic demand. Some of these works take into account discount policies, but none of them introduces a solution approach for the CTQDP. Benton [7] proposes a Lagrangian relaxation based heuristic to solve a supplier selection problem with multiple item, multiple supplier and quantity limitation. Another Lagrangian relaxation based heuristic is developed in Güder et al. [19] to solve a multiple item material cost minimization problem with incremental discount offered by a single supplier. Chauhan and Proth [9] propose a heuristic solution for a variant with concave purchasing cost and thresholds on ordering quantities in a multi-supplier system in the case of single product and in the multi-item one. Burke et al. [8] introduce a simple heuristic method for solving a quantity allocation decision in a multiple supplier setting characterized by three different pricing schemes (linear discounts, incremental units discounts and all units discounts), whereas Awasthi et al. [6] propose a method for a problem with limitation on minimum and maximum order sizes in a multi-source system.

Procurement problems real setting may be complicated by the inclusion of transportation costs either as truckload shipping costs or as cost of visiting suppliers in one or more tours. A classical and well-studied procurement problem involving both traveling and purchasing costs is the Travelling Purchaser Problem (TPP). Several heuristic and exact methods have been proposed for the problem. In Voss [33] the author investigates two dynamic tabu search strategies, whereas in Riera-Ledesma and Salazar-González [29] the authors propose a local search heuristic based on node chains removal. More recently, in Angelelli et al. [4,5] the authors analyze a dynamic extension of the TPP where quantities offered by the suppliers are assumed to decrease over time. Two groups of heuristics are described and compared. The first group consists of simplified approaches based on greedy criteria, the second one includes heuristics based on a look-ahead approach taking future prediction into account. Mansini and Tocchella [27] analyze a version of the TPP where total purchasing costs do not exceed a predefined threshold. The authors provide an enhanced local search heuristic and a Variable Neighborhood Search (VNS) tested in a multi-start variant. Finally, in Mansini et al. [26] the TQD procurement problem is extended to include truckload shipping costs. The authors develop integer programming based heuristics to solve the problem and demonstrate their efficacy on a large set of randomly generated instances.

Over the last few years, the best results found for many practical or academic optimization problems have been obtained using hybrid algorithms. A hybrid method can be thought as an approach combining the principles of different solution methods, usually represented by meta-heuristics (see [31] for a taxonomy). Among the hybrid methods, matheuristics are receiving an increasing attention from research community, thanks also to the great advance in Integer Programming solvers. Matheuristics combine successfully meta-heuristics and mathematical programming techniques to solve complex combinatorial optimization problems (see [25]). Different forms of cooperation between heuristics and exact techniques can be figured out. Some methods use exact algorithms to explore large neighborhoods in local search algorithms. Other methods perform several runs of a local search and exploit information in high quality solutions to define smaller problems that can be easily solved with exact algorithms. Finally, some matheuristics use information from relaxation of integer programming problems to guide local search. All methods described below belong to one of these categories.

Glover et al. [17] investigate whether principles of a Tabu Search algorithm can be embedded into a branch-and-bound structure to solve a graph coloring problem. Haouari and Ladhari [21] analyze a

local search branch-and-bound to solve a flow shop problem. Nagar et al. [28] use a branch-and-bound to generate some information, which in turn is used to guide the search of a genetic algorithm for a two-machine flowshop scheduling problem. A similar approach is analyzed in French et al. [16] where the authors combine genetic algorithms and a branch-and-bound approach to solve MAX-SAT problems. In Woodcock and Wilson [35] a general-purpose mixed integer programming solver is used to tackle sub-problems generated by a Tabu Search as a guiding strategy to solve the Generalized Assignment Problem. De Franceschi et al. [11] propose an ILP-refinement approach for the Distance-Constrained Capacitated Vehicle Routing Problem that uses nodes extracted from a solution to generate a large number of new sequences and then exploits an ILP model for the reallocation of some of these sequences. Framinan and Pastor [15] propose a procedure called Bound Driven Search that combines features of exact enumeration methods (branch-and-bound) with approximate procedures. Subramanian et al. [30] analyze a hybrid algorithm composed of an Iterated Local Search (ILS) based heuristic and a set partitioning (SP) formulation. The SP model is solved by means of an ILP solver that interactively calls the ILS heuristic during its execution. Alvarenga et al. [1] introduce a robust heuristic approach for the Vehicle Routing problem with time windows using an efficient genetic algorithm and a set partitioning formulation. Labadie et al. [23] propose a procedure for the team orienteering problem with time windows that uses information provided by a problem relaxation to guide the search of a Variable Neighborhood Search. Also the Relaxation Induced Neighborhood Search (RINS) introduced by Danna et al. [10] constructs a promising neighborhood using information contained in the continuous relaxation of a MILP problem, whereas Fischetti and Lodi [14] study the use of a MILP solver as a tool to explore solution subspaces defined and controlled by a simple external branching framework. The neighborhoods are obtained through the introduction in the MILP problem of linear inequalities called local branching cuts. Finally, Kernel Search (see [2,3]) proposed for the solution of mixed integer linear programming problems with binary variables consists of a high level heuristic framework with the main task of identifying promising subsets of variables on which to construct MILP subproblems optimally solved by a general MILP solver. We also refer interested readers to Doerner and Schmid [13] for a survey on matheuristics for Rich Vehicle Routing Problems, and to the chapter by Toth and Tramontani [32] on a matheuristic for the Capacitated Vehicle Routing Problem.

## 3. Problem formulation

Let us introduce the following sets of decision variables:

- $z_{ik}^r$:= number of units of product $k$ purchased from supplier $i$ in interval $r$, $i\in S_k$, $k\in K$, $r\in R_i$;
- $y_i^r := \begin{cases} 1 & \text{if } \sum_{k\in K} z_{ik}^r \in [l_i^r, u_i^r] \\ 0 & \text{otherwise} \end{cases}$ $i\in S$, $r\in R_i$.

Then, the Capacitated Total Quantity Discount Problem can be formulated as follows:

$$(CTQDP) \quad v := \min \sum_{k\in K}\sum_{i\in S_k}\sum_{r\in R_i} p_{ik}^r z_{ik}^r \tag{1}$$

subject to

$$\sum_{i\in S_k}\sum_{r\in R_i} z_{ik}^r = d_k, \quad k\in K \tag{2}$$

$$\sum_{r\in R_i} z_{ik}^r \le q_{ik}, \quad k\in K, \ i\in S_k \tag{3}$$

$$\sum_{r\in R_i} y_i^r \le 1, \quad i\in S \tag{4}$$

$$l_i^r y_i^r \le \sum_{k\in K} z_{ik}^r \le u_i^r y_i^r, \quad i\in S, \ r\in R_i \tag{5}$$

$$z_{ik}^r \le \min\{q_{ik}, u_i^r\} y_i^r, \quad k\in K, \ i\in S_k, \ r\in R_i \tag{6}$$

$$z_{ik}^r \ge 0, \quad k\in K, \ i\in S_k, \ r\in R_i \tag{7}$$

$$y_i^r \in \{0,1\}, \quad i\in S, \ r\in R_i \tag{8}$$

Objective function (1) establishes the minimization of the purchasing costs. Constraints (2) ensure that demand $d_k$ is satisfied for each product $k$, whereas constraints (3) state that it is not possible to purchase from supplier $i$ an amount of product $k$ larger than the available quantity $q_{ik}$. Constraints (4) guarantee that at most one interval for each supplier is selected. Constraints (5) define interval bounds for each supplier. If interval $r$ for supplier $i$ is selected ($y_i^r = 1$), then the total amount purchased has to lie between the lower bound $l_i^r$ and the upper bound $u_i^r$. On the contrary, if interval $r$ is not selected ($y_i^r = 0$), then $\sum_{k\in K} z_{ik}^r = 0$. Constraints (6) bounds the quantity that can be purchased for each product in a given interval by the minimum between quantity available for that product and the upper bound of the interval. Finally, constraints (7) and (8) state non-negativity and binary conditions.

In problem formulation integer conditions on $z$-variables are not required. If all input data are integral then, assuming the existence of a feasible solution, there always exists an optimal solution of CTQDP with integral $z$ values (see [18]). Moreover, constraints (6) are valid inequalities we added to strengthen the problem formulation. The optimal solution of the continuous relaxation of problem *(CTQDP)* without valid inequalities (6) chooses the last discount interval for each selected supplier, ignoring the quantities effectively purchased. The introduction of inequalities (6) excludes the systematic selection of the last intervals providing tighter lower bounds.

A simple preprocessing can also be implemented to reduce the number of variables of the original formulation. Let $S^* := \{i\in S$ : there exists $k\in K$ such that $\sum_{j\in S_k\setminus\{i\}} q_{jk} < d_k\}$ be the set of suppliers that has to be necessarily selected in any feasible solution of the problem, then $\sum_{r\in R_i} y_i^r = 1$, $i\in S^*$. Moreover, let $K^* := \{k\in K : \sum_{i\in S_k} q_{ik} = d_k\}$ be the set of products for which all suppliers are necessary to satisfy demand, then constraints (3) become strict inequalities for all $k\in K^*$. Finally, let $R_i^* := \{r\in R_i : \sum_{k\in K} q_{ik} < l_i^r\}$ be the set of intervals for each supplier $i\in S$ that can never be reached also buying all quantities available, then $y_i^r = 0, i\in S, r\in R_i^*$ and $z_{ik}^r = 0$, $k\in K$, $i\in S_k$, $r\in R_i^*$. Furthermore, since demand requirements cannot be exceeded, each coefficient $q_{ik}$ can be replaced everywhere by $\min\{q_{ik}, d_k\}$. A detailed analysis on valid inequalities, properties and preprocessing for the CTQDP can be found in Manerba and Mansini [24].

## 4. Solution algorithms

This section is devoted to the description of the proposed matheuristic and its variant. The method creates neighborhoods and explores them by optimally solving subproblems where all complicating variables but a small subset are fixed as in the best available feasible solution. Variable fixing is made guaranteeing the feasibility of the resulting subproblems. The selection of complicating variables entering each subproblem is made randomly by a VNS approach.

In CTQDP formulation binary variables $y$ are the complicating variables. Different methods can be figured out to fix $y$-variables. Some of them may not guarantee that the optimal solution of the resulting subproblem will be feasible. For instance, the simple ceiling of the optimal value of $y$-variables in the continuous relaxation problem could lead to an infeasible subproblem as shown in the following example.

**Example 1.** Let us consider a problem where $|S|=2, |K|=2$ and $|R_i|=2, \forall i \in S$. Demands $d_k$ for the two products are 2 and 3. For each product $k$ and each supplier $i$, $q_{ik}=4$ and $p_{ik}=8$. Finally, both suppliers have the first discount interval with $l_i^1=0$ and $u_i^1=5$, and the second interval with $l_i^2=6$ and $u_i^2=10$ where $\delta_i^1=0$ and $\delta_i^2=10\%$. In the optimal solution of the continuous relaxation variable $y_1^2$ takes value 0.75, whereas all remaining $y$ variables have value equal to zero. Rounding the value of $y_1^2$ to 1 and keeping the null value for all the other $y$ variables leads to a subproblem with no feasible solution.

As we will describe in the following, the initial solution is obtained by fixing all $y$ variables to predefined integer values, whereas in the matheuristic we formulate MILP subproblems by fixing only subsets of $y$ variables, i.e. selecting predefined discount intervals for some suppliers.

### 4.1. Initial solution

Let $(z^{(LP)}, y^{(LP)})$ be the values assigned to variables $(z,y)$ by the optimal solution of the continuous relaxation (LP relaxation) of the CTQDP. We define as $CTQDP(\overline{y})$ the subproblem obtained from formulation CTQDP by fixing all $y$ variables to some feasible value $\overline{y}$. Since $l_i^1=0$ for all $i \in S$, the following proposition is trivially true:

**Proposition 1.** *Let $(\overline{z},\overline{y})$ be the optimal solution of the subproblem CTQDP $(\overline{y})$ when $\overline{y}_i^r=1$ if $l_i^r \le \sum_{k\in K} z_{ik}^{r(LP)} \le u_i^r$ and zero otherwise. Then $(\overline{z},\overline{y})$ is a feasible solution for the CTQDP.*

The proposition states that selecting the discount intervals identified by quantities purchased in the optimal solution of the LP relaxation, and then optimally solving the resulting subproblem, give rise to a feasible solution for the CTQDP. This solution $(\overline{z},\overline{y})$ represents our initial feasible solution.

### 4.2. Heuristic Enhancement from LP (HELP2)

HELP2 is a matheuristic, globally structured like a Variable Neighborhood Decomposition Search (VNDS). The basic idea of the procedure is that of iteratively solving MILP subproblems of reasonable size. Subproblems are obtained fixing the value of almost all complicating variables $y$ and then solving the resulting subproblem by means of a standard MILP solver. Given a current feasible solution $s^C$, the procedure defines a neighborhood containing all the solutions that differ from $s^C$ for a bounded number of selected intervals. Since $l_i^1=0$, $i \in S$, when nothing is bought from a supplier, this corresponds to select its first interval. Thus, all feasible solutions will select exactly $n$ intervals.

Pseudo-code of HELP2 is described in Algorithm 1. The procedure receives as input the initial feasible solution $s^I=(z^I,y^I)$, and provides as output a possibly improved integer feasible solution $s^F=(z^F,y^F)$.

**Algorithm 1.** HELP2.

**Require**: integer feasible solution $s^I=(z^I,y^I)$ with value $v^I$
**Ensure**: integer feasible solution $s^F=(z^F,y^F)$ with value $v^F$
1: $s^F \leftarrow s^I$, $v^F \leftarrow v^I$
2: **while** $T_{MAX}$ is not reached **do**
3: $h \leftarrow h_{START}$, $h' \leftarrow h'_{START}$, $s^C \leftarrow s^I$, $v^C \leftarrow v^I$
4: **while** $(h>0$ **and** $h'>0)$ **do**
5: Select randomly a set $H \subset S$ of $h$ suppliers
6: Select randomly a set $H' \subset S \backslash H$ of $h'$ suppliers
7: Construct subproblem $MILP(s^C,H,H')$
8: Solve optimally $MILP(s^C,H,H')$. Let $s'$ be its optimal solution with value $v'$
9: **if** $v' < v^C$ **then**
10: $s^C \leftarrow s'$, $v^C \leftarrow v'$
11: $h \leftarrow h_{START}$, $h' \leftarrow h'_{START}$
12: **if** $v' < v^F$ **then**
13: $s^F \leftarrow s^C$, $v^F \leftarrow v^C$
14: **end if**
15: **else**
16: $h \leftarrow h - h_{STEP}$, $h' \leftarrow h' - h'_{STEP}$
17: **end if**
18: **end while**
19: **end while**

After the initialization of parameters $h$ and $h'$, the algorithm randomly selects a set $H$ consisting of $h$ suppliers (Step 5). Next, in Step 6, $h'$ suppliers are randomly selected in $S \backslash H$ defining set $H'$. In Step 7, given a current solution $s^C$, a MILP subproblem is constructed using the sets $H$ and $H'$ as follows. The procedure starts computing the sets $I$ and $I'$. $I$ is the set of all discount intervals of suppliers in $H$, i.e. $I := \bigcup_{i\in H} R_i$. $I'$ is the set of intervals selected by $s^C$ for suppliers in $H'$ plus the intervals, if any, consecutive to these ones, i.e. $I' := \bigcup_{i\in H'} \{r(C,i), r(C,i)+1\}$, where $r(C,i)$ is the interval selected by the current solution $s^C$ for supplier $i \in H'$. If $r(C,i)=r_i$ then only this interval is added to $I'$ for supplier $i$. Subproblem $MILP(s^C,H,H')$ is obtained from the original problem by setting all $y$ variable values as in $s^C$ but for those associated with intervals in $I$ and $I'$. The resulting mixed integer linear programming subproblem will contain $O(\sum_{i\in H}|R_i|+2|I'|)$ variables $y$ and all $z$ variables. Thanks to this variable fixing, the optimal solution of the subproblem will never terminate to be infeasible providing, in the worst case, the feasible solution $s^C$. Hence, the method does not require an additional procedure to verify and possibly restore feasibility.

The subproblem is then solved optimally (Step 8) by a MILP solver. This corresponds to optimally search a neighborhood $N(s^C,h,h')$ consisting of all solutions that differ from $s^C$ for at most $h+h'$ selected intervals and including $s^C$. Given $H$ and $H'$, the neighborhood contains $O(2^{|I'|}\prod_{i\in H}|R_i|)$ solutions, where $\prod_{i\in H}|R_i|$ represents all different ways to select $h$ feasible intervals in $I$, whereas $2^{|I'|}$ all different ways to select $h'$ intervals in $I'$.

If the optimal solution $s'=(z',y')$ has a better cost than solution $s^C$, then $s^C$ is updated (Step 10) and $h$ and $h'$ are reset to their starting values $h_{START}$ and $h'_{START}$ (Step 11). If $s'$ improves the best incumbent solution $s^F$, then $s^F$ is updated too (Step 13). If subproblem $MILP(s^C,H,H')$ does not find a better solution, $h$ and $h'$ are decreased by $h_{STEP}$ and $h'_{STEP}$, respectively (Step 16). The inner while loop is repeated until both $h$ and $h'$ reach value zero. A maximum computing time $T_{MAX}$ is set as the stopping rule of the whole procedure (external while loop, Steps 2–19). The number of times the external loop is computed within the time limit $T_{MAX}$ is identified as the number of *descents*. At the beginning of each descent, the same starting solution $s^I$ is used. In some preliminary experiments we notice that using the best solution found instead of the initial one is usually less effective (the method gets easily stuck into local minima). We say that a *relative improvement* occurs each time $s^C$ is updated, and define as *relative improvement solution* the updated $s^C$ solution. We call *descent solution* the feasible solution provided at the end of a descent and corresponding to the best relative improvement solution found in that descent.

Efficiency and effectiveness of the procedure strongly depend on the selection of the two parameters that guide the search and control the size (computational complexity) of the solved subproblems, namely $h$ and $h'$. More precisely, parameter $h$ controls procedure diversification: the higher the $h$ value, the larger the number of suppliers and thus of intervals that may differ from those selected in the current solution $s^C$. Parameter $h'$ has an intensification role driving the search toward possibly better

feasible solutions. It defines the number of suppliers for which it will be possible to select the interval already selected by the current feasible solution or the consecutive one, moving to larger and thus more convenient discount intervals.

It is evident how smaller values for $h$ and $h'$ lead to less cumbersome subproblems. Nevertheless, we decide to start with large initial values and then to decrement them. The rationale is that we solve at the beginning more complex problems so to possibly identify promising regions of the solutions space and then we intensify the search. In the reverse case the number of subproblems solved would be higher, but most of them may result to be ineffective, searching neighborhoods containing solutions too close to the incumbent one. The efficiency of the solution of subproblems created and optimally solved is also crucial. For this reason we decide not to include inequalities (6) in the subproblem solution. These inequalities may overload the model formulation affecting global computational time when subproblems solved are many. As we have already noticed, these inequalities are instead very useful in the problem LP relaxation used to compute the initial feasible solution. Moreover, in each descent the resolution of a new subproblem can be sped up by using as initial solution the incumbent feasible solution.

Finally, the general approach used by our heuristic can be easily generalized and customized to other MILP problems in which, once decided the value of all or of a subset of complicating decision variables, the remaining subproblem becomes easier to solve.

### 4.3. An ILP refinement

A simple ILP refinement is applied sequentially to HELP2. This refinement mainly exploits the common structure among different feasible solutions found during the HELP2 search of the solutions space. However, the algorithm is general enough to be applied as refinement of any alternative procedure one may use instead of HELP2 to find a set of feasible solutions. Algorithm 2 provides the general pseudo-code of the proposed ILP refinement.

**Algorithm 2.** ILP refinement.

**Require**: two sets $A$ and $B$ of feasible solutions
**Ensure**: integer feasible solution $s^F=(z^F,y^F)$ with value $v^F$
1: **for all** $(s_h,s_p)$ such that $h,p\in A, p>h$ **do**
2: Construct subproblem $MILP_A(s_h,s_p)$
3: Solve it optimally. Let $s'$ be its optimal solution with value $v'$
4: **if** $v'<v^F$ **then**
5: $s^F\leftarrow s'$, $v^F\leftarrow v'$
6: **end if**
7: **end for**
8: Construct subproblem $MILP_B(B,v^F)$.
9: Solve it optimally. Let $s'$ be its optimal solution with value $v'$
10: **if** $v'<v^F$ **then**
11: $s^F\leftarrow s'$, $v^F\leftarrow v'$
12: **end if**

The algorithm receives as input two sets $A$ and $B$ of feasible solutions. In Steps 1–7, for each pair of solutions $(s_h,s_p)$ in $A$, we formulate and solve a MILP subproblem ($MILP_A$) constructed as follows. Let $s_1$ and $s_2$ be a pair of feasible solutions and let $r_1$ and $r_2$ be the interval selected for supplier $i$ by the two solutions, respectively. Then, if $r_1=r_2$ we add the constraint $y_i^{r_1}=1$, otherwise we add the constraint $y_i^{r_1}+y_i^{r_2}=1$. Similar constraints are added for each supplier $i\in S$. When solving $MILP_A$ both $s_h$ and $s_p$ are provided as starting solutions to the MILP solver.

Set $B$ is used to formulate (Step 8) and solve exactly (Step 9) a final MILP problem ($MILP_B$). In particular, such a problem uses the value $v^F$ of the best incumbent solution as a cut-off value and is constructed adding to the original formulation the following set of constraints:

$$y_i^r=0,\quad i\in S,\ r\in G_i,$$

where $G_i$ is the set of intervals of supplier $i$ that have been selected in the solutions of the set $B$ a number of times strictly lower than $\gamma|B|$, where $0<\gamma<1$.

In our implementation of the ILP refinement the sets $A$ and $B$ include the $l$ best descent solutions and all the relative improvement solutions found by HELP2, respectively. It is evident how, in this case, $A\subseteq B$.

**Table 1**
Parameters value for HELP2.

| Parameter | Value |
|---|---|
| $h_{START}$ | 10 |
| $h_{STEP}$ | $h_{START}/5$ |
| $h'_{START}$ | $\lfloor\frac{n}{100}\rfloor * h_{START}$ |
| $h'_{STEP}$ | $\lfloor\frac{n}{100}\rfloor * h_{STEP}$ |
| $T_{MAX}$ | $\phi_1 m+\phi_2\lfloor\frac{n}{100}\rfloor$ |
| $\phi_1$ (for Class1-$\lambda=0.8$ instances) | 1/2 |
| $\phi_1$ (for all other instances) | 2 |
| $\phi_2$ (for Class1-$\lambda=0.8$ instances) | $m/2$ |
| $\phi_2$ (for all other instances) | 200 |

**Table 2**
Subproblems analysis for HELP2.

| $n$ | $m$ | class | $\lambda$ | #MILPs | avg $t$(s) |
|---|---|---|---|---|---|
| 50 | 100 | 1 | 0.1 | 197 | 1.02 |
| 50 | 100 | 1 | 0.8 | 160 | 0.31 |
| 50 | 100 | 2 | 0.1 | 135 | 1.49 |
| 50 | 100 | 2 | 0.8 | 303 | 0.66 |
| 50 | 200 | 1 | 0.1 | 131 | 3.05 |
| 50 | 200 | 1 | 0.8 | 91 | 1.14 |
| 50 | 200 | 2 | 0.1 | 98 | 4.12 |
| 50 | 200 | 2 | 0.8 | 301 | 1.34 |
| 50 | 300 | 1 | 0.1 | 91 | 6.65 |
| 50 | 300 | 1 | 0.8 | 60 | 2.59 |
| 50 | 300 | 2 | 0.1 | 78 | 7.76 |
| 50 | 300 | 2 | 0.8 | 204 | 2.96 |
| 50 | 500 | 1 | 0.1 | 49 | 20.61 |
| 50 | 500 | 1 | 0.8 | 50 | 4.98 |
| 50 | 500 | 2 | 0.1 | 40 | 25.31 |
| 50 | 500 | 2 | 0.8 | 152 | 6.65 |
| 100 | 100 | 1 | 0.1 | 290 | 1.36 |
| 100 | 100 | 1 | 0.8 | 158 | 0.61 |
| 100 | 100 | 2 | 0.1 | 167 | 2.38 |
| 100 | 100 | 2 | 0.8 | 397 | 0.99 |
| 100 | 200 | 1 | 0.1 | 101 | 5.94 |
| 100 | 200 | 1 | 0.8 | 103 | 1.90 |
| 100 | 200 | 2 | 0.1 | 63 | 9.50 |
| 100 | 200 | 2 | 0.8 | 144 | 4.16 |
| 100 | 300 | 1 | 0.1 | 78 | 10.20 |
| 100 | 300 | 1 | 0.8 | 64 | 4.71 |
| 100 | 300 | 2 | 0.1 | 39 | 20.50 |
| 100 | 300 | 2 | 0.8 | 83 | 10.02 |
| 100 | 500 | 1 | 0.1 | 29 | 41.86 |
| 100 | 500 | 1 | 0.8 | 29 | 17.40 |
| 100 | 500 | 2 | 0.1 | 28 | 44.84 |
| 100 | 500 | 2 | 0.8 | 79 | 15.40 |
| | | | | 124.75 | 8.83 |

As described the ILP refinement consists of two main parts. In Part I, including Steps 1–7, $\binom{l}{2}$ $MILP_A$ subproblems are constructed and solved. Since the best descent solutions have a large number of common intervals, the resulting $MILP_A$ problems are usually quite small and thus quite easy to solve to optimality. The rationale behind Part I is to look for improvements in solution quality never requiring high computational times.

In very large instances, HELP2 often cannot even complete a single descent within the time limit $T_{MAX}$, making useless Part I of the algorithm. Part II (Steps 8–12) tries to partially overcome this drawback providing a different improvement procedure with the solution of problem $MILP_B$. To speed up computational time required to solve $MILP_B$, the objective function value of the best incumbent solution $v^F$ is used as a cut-off value.

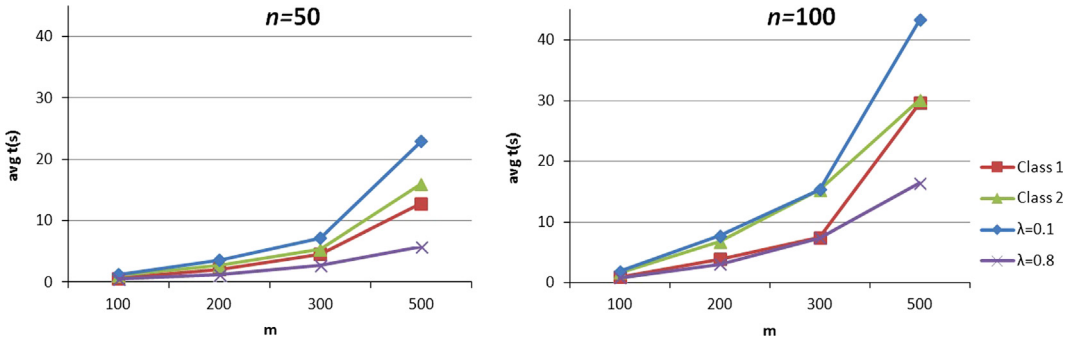


**Fig. 1.** MILP subproblems average solution time (in seconds).

**Table 3**
HELP2 results for Class 1 instances.

| $n$ | $m$ | # | $\lambda = 0.1$ | | | | | | | $\lambda = 0.8$ | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Best known | | HELP2 | | | | | Best known | | HELP2 | | | | |
| | | | $t^*$ | $v^*$ | $t$ | init | max | avg | min | $t^*$ | $v^*$ | $t$ | init | max | avg | min |
| 50 | 100 | 1 | 270 | 1,065,331.43 | 201 | 1.59 | 0.09 | 0.04 | **0.00** | 61 | 259,244.06 | 50 | 3.09 | 0.06 | 0.02 | **0.00** |
| 50 | 100 | 2 | 270 | 1,053,147.65 | 202 | 1.66 | 0.11 | 0.02 | **0.00** | 83 | 239,212.53 | 51 | 2.99 | 0.13 | 0.11 | **0.00** |
| 50 | 100 | 3 | 244 | 1,063,178.97 | 202 | 0.59 | **0.00** | **0.00** | **0.00** | 72 | 255,411.50 | 51 | 0.82 | **0.00** | **0.00** | **0.00** |
| 50 | 100 | 4 | 234 | 933,882.86 | 201 | 1.03 | 0.01 | 0.00 | **0.00** | 60 | 245,170.07 | 50 | 0.62 | **0.00** | **0.00** | **0.00** |
| 50 | 100 | 5 | 234 | 994,708.65 | 201 | 1.15 | 0.04 | 0.01 | **0.00** | 68 | 257,293.66 | 50 | 1.65 | **0.00** | **0.00** | **0.00** |
| 50 | 200 | 1 | 523 | 1,952,152.07 | 404 | 0.77 | 0.01 | 0.00 | **0.00** | 201 | 504,369.86 | 102 | 2.77 | 0.62 | 0.22 | **0.00** |
| 50 | 200 | 2 | 607 | 2,132,512.11 | 409 | 1.60 | 0.19 | 0.08 | **0.00** | 199 | 501,271.64 | 104 | 2.73 | 0.14 | 0.04 | **0.00** |
| 50 | 200 | 3 | 522 | 1,998,714.50 | 406 | 0.52 | **0.00** | **0.00** | **0.00** | 177 | 546,469.73 | 102 | 2.95 | 0.17 | 0.04 | **0.00** |
| 50 | 200 | 4 | 508 | 2,103,277.09 | 405 | 1.41 | 0.09 | 0.05 | **0.00** | 175 | 520,573.11 | 101 | 3.24 | 0.23 | 0.09 | **0.00** |
| 50 | 200 | 5 | 517 | 1,935,851.94 | 409 | 1.62 | 0.18 | 0.08 | **0.00** | 164 | 475,366.42 | 101 | 3.49 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 1 | 958 | 2,710,818.62 | 607 | 0.96 | 0.10 | 0.02 | **0.00** | 338 | 810,963.79 | 157 | 2.38 | 0.13 | 0.05 | **0.00** |
| 50 | 300 | 2 | 772 | 2,762,370.39 | 609 | 0.33 | 0.04 | 0.01 | **0.00** | 198 | 801,325.23 | 152 | 0.44 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 3 | 975 | 3,150,592.95 | 610 | 0.15 | 0.03 | 0.02 | 0.02 | 264 | 754,396.40 | 153 | 2.88 | 0.27 | 0.05 | **0.00** |
| 50 | 300 | 4 | 1093 | 3,180,894.49 | 604 | 1.91 | 0.53 | 0.14 | **0.00** | 208 | 785,817.78 | 155 | 4.88 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 5 | 805 | 3,206,291.08 | 612 | 0.35 | 0.08 | 0.02 | **0.00** | 204 | 743,753.17 | 151 | 2.40 | 0.25 | 0.10 | **0.00** |
| 50 | 500 | 1 | 1571 | 5,087,519.13 | 1076 | 1.32 | 0.25 | 0.11 | **0.00** | 738 | 1,320,857.89 | 274 | 0.41 | 0.12 | 0.02 | **0.00** |
| 50 | 500 | 2 | 1534 | 4,816,132.22 | 1057 | 0.31 | 0.03 | 0.01 | **0.00** | 491 | 1,267,401.38 | 255 | 2.83 | 0.99 | 0.34 | **0.00** |
| 50 | 500 | 3 | 1711 | 5,073,596.64 | 1023 | 1.44 | 0.23 | 0.08 | 0.02 | 377 | 1,347,170.84 | 257 | 0.05 | **0.00** | **0.00** | **0.00** |
| 50 | 500 | 4 | 1517 | 4,976,074.66 | 1083 | 1.81 | 0.74 | 0.32 | **0.00** | 436 | 1,219,496.74 | 269 | 3.73 | 0.52 | 0.10 | **0.00** |
| 50 | 500 | 5 | 1678 | 5,053,157.68 | 1023 | 1.20 | 0.16 | 0.07 | 0.02 | 400 | 1,228,369.20 | 264 | 3.19 | 0.41 | 0.08 | **0.00** |
| 100 | 100 | 1 | 456 | 2,040,173.12 | 401 | 0.12 | 0.00 | 0.00 | **0.00** | 184 | 459,820.43 | 101 | 1.31 | 0.63 | 0.24 | **0.00** |
| 100 | 100 | 2 | 481 | 1,912,898.23 | 401 | 0.12 | 0.02 | 0.01 | **0.00** | 204 | 422,275.23 | 101 | 2.47 | 0.17 | 0.09 | **0.00** |
| 100 | 100 | 3 | 453 | 1,873,337.80 | 402 | 0.28 | **0.00** | **0.00** | **0.00** | 149 | 445,686.18 | 101 | 2.08 | 0.12 | 0.04 | **0.00** |
| 100 | 100 | 4 | 503 | 1,922,911.67 | 402 | 0.95 | 0.06 | 0.02 | **0.00** | 156 | 472,285.13 | 101 | 2.37 | 0.95 | 0.40 | 0.03 |
| 100 | 100 | 5 | 536 | 2,155,736.71 | 402 | 0.14 | 0.01 | 0.01 | **0.00** | 207 | 416,908.89 | 101 | 1.06 | 0.36 | 0.15 | **0.00** |
| 100 | 200 | 1 | 873 | 4,360,648.52 | 607 | 0.76 | 0.15 | 0.05 | 0.01 | 570 | 883,411.52 | 204 | 1.97 | 0.03 | 0.01 | **0.00** |
| 100 | 200 | 2 | 933 | 4,154,529.60 | 605 | 0.47 | 0.11 | 0.05 | 0.00 | 501 | 875,510.12 | 202 | 1.45 | 0.31 | 0.19 | **0.00** |
| 100 | 200 | 3 | 1100 | 3,797,749.45 | 616 | 1.11 | 0.06 | 0.03 | **0.00** | 391 | 895,753.50 | 202 | 1.77 | 0.64 | 0.20 | **0.00** |
| 100 | 200 | 4 | 879 | 3,926,225.47 | 610 | 0.34 | 0.02 | 0.01 | **0.00** | 358 | 883,419.34 | 203 | 1.53 | **0.00** | **0.00** | **0.00** |
| 100 | 200 | 5 | 962 | 3,668,949.23 | 611 | 0.14 | 0.01 | 0.00 | **0.00** | 421 | 878,740.30 | 204 | 2.08 | 0.63 | 0.36 | **0.00** |
| 100 | 300 | 1 | 1504 | 5,719,285.13 | 812 | 0.14 | 0.02 | 0.01 | **0.00** | 2941 | 1,429,177.24 | 319 | 1.76 | 0.54 | 0.29 | 0.03 |
| 100 | 300 | 2 | 1429 | 5,711,788.37 | 813 | 0.71 | 0.18 | 0.08 | 0.01 | 1241 | 1,433,131.13 | 317 | 3.66 | 1.71 | 1.00 | 0.58 |
| 100 | 300 | 3 | 1763 | 5,584,860.56 | 821 | 0.49 | 0.10 | 0.04 | 0.01 | 1098 | 1,365,427.33 | 312 | 1.70 | 0.52 | 0.15 | **0.00** |
| 100 | 300 | 4 | 1583 | 6,143,443.91 | 821 | 0.19 | 0.07 | 0.02 | 0.00 | 1480 | 1,282,178.92 | 301 | 2.68 | 0.55 | 0.30 | 0.16 |
| 100 | 300 | 5 | 1338 | 6,135,946.34 | 814 | 0.31 | 0.08 | 0.04 | 0.01 | 1221 | 1,255,026.29 | 308 | 4.13 | 1.08 | 0.80 | 0.41 |
| 100 | 500 | 1 | 2666 | 9,240,585.56 | 1238 | 1.24 | 0.50 | 0.24 | 0.01 | 3591 | 2,264,190.57 | 543 | 1.02 | 0.62 | 0.34 | 0.11 |
| 100 | 500 | 2 | 2621 | 9,971,412.11 | 1269 | 1.34 | 0.14 | 0.07 | 0.01 | 4035 | 2,290,165.46 | 542 | 0.59 | 0.40 | 0.21 | 0.02 |
| 100 | 500 | 3 | 3317 | 9,861,834.02 | 1232 | 0.67 | 0.21 | 0.09 | 0.01 | 4808 | 2,229,422.19 | 544 | 4.93 | 2.58 | 1.73 | 0.66 |
| 100 | 500 | 4 | 5319 | 9,632,126.38 | 1268 | 0.70 | 0.27 | 0.11 | **0.00** | 3644 | 2,025,046.59 | 514 | 3.94 | 1.32 | 0.57 | 0.04 |
| 100 | 500 | 5 | 5143 | 9,851,746.34 | 1282 | 1.73 | 0.42 | 0.27 | 0.14 | 3976 | 2,218,089.75 | 530 | 4.20 | 3.33 | 1.92 | 0.92 |
| | | | | | | 0.84 | 0.13 | 0.06 | 0.01 | | | | 2.36 | 0.51 | 0.26 | 0.07 |

## 5. Computational results

The performance of all algorithms has been tested on the set of benchmark problems proposed in Manerba and Mansini [24]. The set consists of two classes of instances that differ only for the discount policy used: in Class 1 suppliers can have 3–5 discount intervals with random discount rates, whereas in Class 2 suppliers have exactly 3 intervals and the same discount rates equal to {0%,10%,50%}. For each class, the number of suppliers $n$ and the number of products $m$ are set to {50, 100} and to {100, 200, 300, 500}, respectively. Products demand is computed according to a parameter $\lambda$ set equal to $\{0.1, 0.8\}$. This parameter is strictly related to the expected number of suppliers in a feasible solution ($\lambda = 0.1$ represents the case in which most part of suppliers are necessary to satisfy demand, whereas $\lambda = 0.8$ that in which only a restricted part of suppliers are necessary). Moreover, 5 random instances have been generated for each combination $(n, m, \lambda)$. This means 160 instances all together in the two classes. Up to this date, 37 out of these 160 instances have not been solved to optimality yet.

All computational tests have been executed on an *Intel Core Duo* 2 GHz computer, with 2 GB RAM and running *Windows Vista*32-bit operating system. We use CPLEX 12.2 and C++ Concert Technology to solve all MILP subproblems. These are exactly the same machine and the same MILP solver version used in Manerba and Mansini [24].

After some preliminary tests, we have decided to set the parameters of HELP2 to the values reported in Table 1. The integer parameter $h_{START}$ has been set to 10 for all instances. This value is the result of a tuning process where the parameter has been varied in the range $\left[5, \lceil \frac{3}{10} n \rceil\right]$. Remaining parameters value depends on $h_{START}$ and on the input size. In particular, the time limit $T_{MAX}$ increases linearly with the number of products $m$, and has an additional bonus time when the number of suppliers $n$ is equal to 100. Class 1 instances with $\lambda = 0.8$, which seem to be the most easy to solve, have a reduced $T_{MAX}$.

The goodness of all these parameter values is also confirmed by the average time required to solve MILP subproblems. In Table 2 we show the number of subproblems solved by HELP2 (#MILPs) and the corresponding average solution time in seconds (avg t(s)) for a subset of 32 instances, one for each combination of $n, m$, *class* and $\lambda$. The number of subproblems solved tends to reduce with the size of the input. Nevertheless, also for most difficult instances, this number is never lower than 28, whereas the total average computational time is about 8 s. Fig. 1 shows how instances with $n = 50$ of Class 2 require, on average, more time respect those of Class 1, even if this difference becomes evident just for $m = 500$.

**Table 4**
HELP2 results for Class 2 instances.

| $n$ | $m$ | # | $\lambda = 0.1$ | | | | | | | $\lambda = 0.8$ | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Best known | | HELP2 | | | | | Best known | | HELP2 | | | | |
| | | | $t^*$ | $v^*$ | $t$ | init | max | avg | min | $t^*$ | $v^*$ | $t$ | init | max | avg | min |
| 50 | 100 | 1 | 1954 | 888,275.88 | 202 | 20.09 | 0.62 | 0.45 | 0.09 | 526 | 209,340.00 | 201 | 29.30 | 2.65 | 0.97 | **0.00** |
| 50 | 100 | 2 | 573 | 750,361.55 | 202 | 10.59 | 0.86 | 0.48 | 0.10 | 383 | 221,929.00 | 202 | 25.29 | 1.45 | 0.62 | **0.00** |
| 50 | 100 | 3 | 442 | 784,005.14 | 202 | 9.93 | 1.12 | 0.75 | 0.50 | 419 | 204,096.50 | 204 | 32.40 | 2.44 | 1.37 | **0.00** |
| 50 | 100 | 4 | 487 | 828,232.43 | 203 | 9.93 | 0.64 | 0.46 | 0.25 | 450 | 233,650.00 | 203 | 18.99 | 0.96 | 0.42 | **0.00** |
| 50 | 100 | 5 | 483 | 752,930.08 | 202 | 13.27 | 1.01 | 0.65 | **0.00** | 328 | 208,702.61 | 201 | 27.87 | 0.21 | 0.12 | **0.00** |
| 50 | 200 | 1 | 4258 | 1,706,913.61 | 412 | 21.50 | 1.16 | 0.77 | 0.42 | 1254 | 425,279.00 | 401 | 28.78 | 1.34 | 0.27 | **0.00** |
| 50 | 200 | 2 | 4530 | 1,542,150.28 | 409 | 20.94 | 0.85 | 0.67 | 0.41 | 1628 | 470,957.00 | 411 | 36.85 | 2.11 | 1.32 | 0.79 |
| 50 | 200 | 3 | 2414 | 1,554,503.81 | 403 | 21.83 | 1.88 | 1.11 | 0.56 | 1084 | 387,964.50 | 409 | 25.62 | 0.90 | 0.18 | **0.00** |
| 50 | 200 | 4 | 6951 | 1,703,115.96 | 406 | 11.69 | 1.07 | 0.91 | 0.66 | 1676 | 405,405.50 | 406 | 25.36 | 0.08 | 0.03 | **0.00** |
| 50 | 200 | 5 | 6888 | 1,569,847.52 | 406 | 16.85 | 1.15 | 0.91 | 0.61 | 663 | 416,658.50 | 402 | 15.61 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 1 | 7793 | 2,447,712.05 | 615 | 16.47 | 1.70 | 1.18 | 0.79 | 2841 | 632,539.00 | 614 | 25.45 | 2.42 | 0.67 | **0.00** |
| 50 | 300 | 2 | 7008 | 2,461,007.61 | 613 | 15.66 | 1.41 | 1.12 | 0.87 | 6981 | 599,533.50 | 603 | 26.21 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 3 | – | 2,333,156.03 | 630 | 20.78 | 2.54 | 1.99 | 1.44 | 2762 | 657,022.00 | 610 | 25.44 | 0.27 | 0.05 | **0.00** |
| 50 | 300 | 4 | 4371 | 2,164,848.92 | 609 | 9.94 | 1.02 | 0.91 | 0.73 | 3760 | 624,807.00 | 615 | 28.43 | 1.62 | 0.91 | **0.00** |
| 50 | 300 | 5 | – | 2,358,755.50 | 614 | 19.07 | 1.45 | 0.79 | 0.22 | 4418 | 631,879.00 | 610 | 26.60 | 0.81 | 0.28 | **0.00** |
| 50 | 500 | 1 | – | 4,099,394.12 | 1049 | 17.55 | 0.72 | 0.59 | 0.43 | – | 1,172,101.50 | 1079 | 22.79 | 0.69 | 0.25 | **0.00** |
| 50 | 500 | 2 | – | 3,847,179.41 | 1054 | 29.76 | 1.89 | 1.34 | 0.94 | 11,830 | 978,685.50 | 1043 | 24.43 | 0.63 | 0.22 | **0.00** |
| 50 | 500 | 3 | 14,777 | 4,037,572.32 | 1048 | 23.25 | 3.91 | 2.60 | 1.85 | – | 1,178,217.00 | 1028 | 25.66 | 1.43 | 0.86 | **0.00** |
| 50 | 500 | 4 | – | 3,860,036.57 | 1031 | 20.34 | 0.59 | 0.34 | 0.08 | – | 1,135,915.50 | 1022 | 24.90 | **0.00** | **−0.15** | **−0.25** |
| 50 | 500 | 5 | – | 3,846,026.49 | 1022 | 23.79 | 1.42 | 0.77 | 0.39 | – | 1,069,179.00 | 1019 | 24.56 | 0.36 | 0.14 | **−0.16** |
| 100 | 100 | 1 | 835 | 1,560,407.87 | 406 | 12.88 | 1.46 | 1.27 | 1.02 | 1553 | 364,230.50 | 404 | 12.15 | 0.41 | 0.23 | **0.00** |
| 100 | 100 | 2 | 3505 | 1,430,206.02 | 404 | 9.57 | 0.84 | 0.59 | 0.40 | 3290 | 383,295.50 | 402 | 31.19 | 2.75 | 1.60 | 0.88 |
| 100 | 100 | 3 | 2280 | 1,655,161.62 | 404 | 11.19 | 1.04 | 0.87 | 0.76 | 1774 | 367,575.50 | 402 | 20.01 | 1.94 | 1.09 | 0.60 |
| 100 | 100 | 4 | 1596 | 1,459,152.50 | 403 | 8.19 | 1.15 | 0.85 | 0.60 | 1729 | 387,625.50 | 402 | 25.62 | 3.34 | 2.41 | 0.83 |
| 100 | 100 | 5 | 1313 | 1,570,368.95 | 403 | 16.47 | 2.13 | 1.30 | 0.92 | 1874 | 374,945.00 | 403 | 31.21 | 2.25 | 1.72 | 1.05 |
| 100 | 200 | 1 | – | 3,386,881.90 | 608 | 13.34 | 1.10 | 0.81 | 0.62 | 11,584 | 708,833.50 | 612 | 30.28 | 5.00 | 2.92 | 1.91 |
| 100 | 200 | 2 | – | 3,353,533.05 | 609 | 4.82 | 0.18 | 0.14 | 0.09 | – | 748,779.00 | 614 | 28.25 | 1.70 | 1.10 | 0.13 |
| 100 | 200 | 3 | – | 3,359,507.13 | 612 | 6.97 | 0.62 | 0.50 | 0.37 | – | 775,076.00 | 616 | 35.54 | 4.04 | 3.14 | 1.54 |
| 100 | 200 | 4 | 10,297 | 3,285,983.24 | 621 | 11.57 | 2.20 | 1.44 | 0.90 | 5004 | 768,933.00 | 615 | 26.61 | 4.36 | 3.93 | 3.65 |
| 100 | 200 | 5 | – | 3,254,487.50 | 611 | 17.08 | 1.85 | 1.46 | 0.89 | – | 765,640.00 | 607 | 30.26 | 2.28 | 2.08 | 1.91 |
| 100 | 300 | 1 | – | 4,867,747.01 | 831 | 12.83 | 1.60 | 1.18 | 0.86 | – | 1,096,232.00 | 813 | 29.55 | 5.18 | 4.01 | 2.02 |
| 100 | 300 | 2 | – | 4,918,446.29 | 815 | 12.83 | 1.84 | 1.54 | 0.77 | – | 1,164,793.00 | 842 | 29.38 | 2.99 | 2.63 | 2.35 |
| 100 | 300 | 3 | – | 4,680,882.21 | 842 | 16.32 | 2.20 | 1.44 | 0.80 | – | 1,158,439.50 | 837 | 25.19 | 4.91 | 4.45 | 3.79 |
| 100 | 300 | 4 | – | 4,942,760.09 | 818 | 13.25 | 2.37 | 1.14 | 0.59 | – | 1,187,005.00 | 831 | 22.80 | 0.52 | 0.27 | 0.08 |
| 100 | 300 | 5 | – | 4,852,938.98 | 857 | 11.01 | 2.46 | 1.58 | 0.75 | – | 1,214,561.00 | 837 | 26.95 | 2.30 | 1.42 | 0.93 |
| 100 | 500 | 1 | – | 8,202,625.72 | 1268 | 17.55 | 2.64 | 1.96 | 1.62 | – | 1,821,572.00 | 1382 | 27.01 | 5.24 | 4.18 | 2.00 |
| 100 | 500 | 2 | – | 8,252,585.36 | 1278 | 12.54 | 4.01 | 2.09 | 0.74 | – | 1,827,191.50 | 1283 | 20.87 | 2.24 | 1.60 | 0.87 |
| 100 | 500 | 3 | – | 8,522,604.60 | 1279 | 14.28 | 1.29 | 0.87 | 0.14 | – | 1,917,578.50 | 1288 | 21.33 | 0.56 | **−0.04** | **−0.32** |
| 100 | 500 | 4 | – | 8,805,448.70 | 1262 | 9.11 | 1.83 | 0.95 | 0.29 | – | 1,888,163.50 | 1331 | 27.18 | 4.60 | 3.48 | 0.82 |
| 100 | 500 | 5 | – | 8,277,027.11 | 1299 | 9.16 | 1.76 | 0.47 | **−0.06** | – | 1,851,845.00 | 1239 | 27.57 | 5.20 | 3.75 | 1.87 |
| | | | | | | 14.85 | 1.54 | 1.03 | 0.61 | | | | 26.24 | 2.05 | 1.36 | 0.68 |

**Table 5**
number of optimal (or improved) results in HELP2.

| $n$ | $m$ | # | Class 1 | | Class 2 | |
|---|---|---|---|---|---|---|
| | | | $\lambda=0.1$ | $\lambda=0.8$ | $\lambda=0.1$ | $\lambda=0.8$ |
| 50 | 100 | 1 | 1 | 3 | 0 | 3 |
| 50 | 100 | 2 | 1 | 1 | 0 | 2 |
| 50 | 100 | 3 | 5 | 5 | 0 | 1 |
| 50 | 100 | 4 | 2 | 5 | 0 | 2 |
| 50 | 100 | 5 | 2 | 5 | 1 | 2 |
| 50 | 200 | 1 | 3 | 2 | 0 | 4 |
| 50 | 200 | 2 | 1 | 2 | 0 | 0 |
| 50 | 200 | 3 | 5 | 3 | 0 | 4 |
| 50 | 200 | 4 | 2 | 3 | 0 | 3 |
| 50 | 200 | 5 | 2 | 5 | 0 | 5 |
| 50 | 300 | 1 | 3 | 3 | 0 | 3 |
| 50 | 300 | 2 | 4 | 5 | 0 | 5 |
| 50 | 300 | 3 | 0 | 4 | 0 | 4 |
| 50 | 300 | 4 | 1 | 5 | 0 | 2 |
| 50 | 300 | 5 | 3 | 3 | 0 | 3 |
| 50 | 500 | 1 | 1 | 4 | 0 | 2 |
| 50 | 500 | 2 | 3 | 2 | 0 | 3 |
| 50 | 500 | 3 | 0 | 5 | 0 | 1 |
| 50 | 500 | 4 | 1 | 4 | 0 | 5 |
| 50 | 500 | 5 | 0 | 4 | 0 | 2 |
| 100 | 100 | 1 | 4 | 2 | 0 | 2 |
| 100 | 100 | 2 | 3 | 1 | 0 | 0 |
| 100 | 100 | 3 | 5 | 1 | 0 | 0 |
| 100 | 100 | 4 | 2 | 0 | 0 | 0 |
| 100 | 100 | 5 | 2 | 2 | 0 | 0 |
| 100 | 200 | 1 | 0 | 3 | 0 | 0 |
| 100 | 200 | 2 | 0 | 2 | 0 | 0 |
| 100 | 200 | 3 | 1 | 1 | 0 | 0 |
| 100 | 200 | 4 | 1 | 5 | 0 | 0 |
| 100 | 200 | 5 | 1 | 1 | 0 | 0 |
| 100 | 300 | 1 | 2 | 0 | 0 | 0 |
| 100 | 300 | 2 | 0 | 0 | 0 | 0 |
| 100 | 300 | 3 | 0 | 3 | 0 | 0 |
| 100 | 300 | 4 | 1 | 0 | 0 | 0 |
| 100 | 300 | 5 | 0 | 0 | 0 | 0 |
| 100 | 500 | 1 | 0 | 0 | 0 | 0 |
| 100 | 500 | 2 | 0 | 0 | 0 | 0 |
| 100 | 500 | 3 | 0 | 0 | 0 | 4 |
| 100 | 500 | 4 | 1 | 0 | 0 | 0 |
| 100 | 500 | 5 | 0 | 0 | 1 | 0 |
| | | | 1.58 | 2.35 | 0.05 | 1.55 |

**Table 6**
HELP2+ results for Class 2 and $\lambda=0.1$ instances.

| $n$ | $m$ | # | Best known | | HELP2+ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $t^*$ | $v^*$ | $t_{tot}$ | $t_+$ | #impr | max | avg | min |
| 50 | 100 | 1 | 1954 | 888,275.88 | 215 | 14 | 0 | 0.24 | 0.07 | 0.03 |
| 50 | 100 | 2 | 573 | 750,361.55 | 211 | 11 | 1 | 0.54 | 0.29 | **0.00** |
| 50 | 100 | 3 | 442 | 784,005.14 | 217 | 15 | 0 | 0.41 | 0.27 | 0.11 |
| 50 | 100 | 4 | 487 | 828,232.43 | 211 | 11 | 0 | 0.46 | 0.29 | 0.10 |
| 50 | 100 | 5 | 483 | 752,930.08 | 211 | 10 | 1 | 0.70 | 0.22 | **0.00** |
| 50 | 200 | 1 | 4258 | 1,706,913.61 | 435 | 33 | 1 | 1.26 | 0.42 | **0.00** |
| 50 | 200 | 2 | 4530 | 1,542,150.28 | 437 | 35 | 0 | 0.62 | 0.35 | 0.18 |
| 50 | 200 | 3 | 2414 | 1,554,503.81 | 441 | 39 | 0 | 0.70 | 0.33 | 0.08 |
| 50 | 200 | 4 | 6951 | 1,703,115.96 | 431 | 27 | 0 | 0.70 | 0.59 | 0.46 |
| 50 | 200 | 5 | 6888 | 1,569,847.52 | 462 | 59 | 1 | 0.60 | 0.28 | **0.00** |
| 50 | 300 | 1 | 7793 | 2,447,712.05 | 665 | 60 | 0 | 0.89 | 0.70 | 0.55 |
| 50 | 300 | 2 | 7008 | 2,461,007.61 | 678 | 72 | 0 | 0.98 | 0.69 | 0.54 |
| 50 | 300 | 3 | – | 2,333,156.03 | 648 | 45 | 0 | 0.39 | 0.18 | 0.08 |
| 50 | 300 | 4 | 4371 | 2,164,848.92 | 664 | 61 | 0 | 0.99 | 0.75 | 0.63 |
| 50 | 300 | 5 | – | 2,358,755.50 | 677 | 73 | 0 | 0.18 | 0.08 | 0.01 |
| 50 | 500 | 1 | – | 4,099,394.12 | 1152 | 143 | 0 | 0.63 | 0.32 | 0.09 |
| 50 | 500 | 2 | – | 3,847,179.41 | 1092 | 72 | 0 | 1.09 | 0.35 | 0.09 |
| 50 | 500 | 3 | 14,777 | 4,037,572.32 | 1120 | 97 | 0 | 1.33 | 1.12 | 0.49 |
| 50 | 500 | 4 | – | 3,860,036.57 | 1337 | 315 | 2 | 0.42 | 0.08 | **−0.11** |
| 50 | 500 | 5 | – | 3,846,026.49 | 1267 | 245 | 0 | 0.46 | 0.36 | 0.30 |
| 100 | 100 | 1 | 835 | 1,560,407.87 | 430 | 28 | 0 | 0.68 | 0.58 | 0.50 |
| 100 | 100 | 2 | 3505 | 1,430,206.02 | 429 | 28 | 0 | 0.52 | 0.28 | 0.07 |
| 100 | 100 | 3 | 2280 | 1,655,161.62 | 451 | 50 | 0 | 0.47 | 0.38 | 0.26 |
| 100 | 100 | 4 | 1596 | 1,459,152.50 | 453 | 52 | 0 | 0.60 | 0.44 | 0.28 |
| 100 | 100 | 5 | 1313 | 1,570,368.95 | 437 | 35 | 0 | 0.67 | 0.40 | 0.22 |
| 100 | 200 | 1 | – | 3,386,881.90 | 649 | 45 | 0 | 0.64 | 0.38 | 0.13 |
| 100 | 200 | 2 | – | 3,353,533.05 | 656 | 50 | 0 | 1.35 | 0.71 | 0.26 |
| 100 | 200 | 3 | – | 3,359,507.13 | 661 | 55 | 0 | 1.24 | 0.84 | 0.45 |
| 100 | 200 | 4 | 10,297 | 3,285,983.24 | 723 | 119 | 0 | 0.67 | 0.58 | 0.34 |
| 100 | 200 | 5 | – | 3,254,487.50 | 822 | 216 | 0 | 0.86 | 0.59 | 0.35 |
| 100 | 300 | 1 | – | 4,867,747.01 | 999 | 181 | 0 | 1.11 | 0.64 | 0.10 |
| 100 | 300 | 2 | – | 4,918,446.29 | 1146 | 333 | 0 | 1.19 | 0.89 | 0.58 |
| 100 | 300 | 3 | – | 4,680,882.21 | 974 | 157 | 0 | 1.86 | 1.30 | 0.73 |
| 100 | 300 | 4 | – | 4,942,760.09 | 1013 | 199 | 0 | 1.03 | 0.73 | 0.58 |
| 100 | 300 | 5 | – | 4,852,938.98 | 1099 | 291 | 0 | 1.04 | 0.81 | 0.48 |
| 100 | 500 | 1 | – | 8,202,625.72 | 1568 | 329 | 0 | 1.97 | 1.05 | 0.73 |
| 100 | 500 | 2 | – | 8,252,585.36 | 1399 | 159 | 0 | 1.82 | 1.38 | 0.86 |
| 100 | 500 | 3 | – | 8,522,604.60 | 1478 | 243 | 0 | 1.16 | 0.82 | 0.43 |
| 100 | 500 | 4 | – | 8,805,448.70 | 1438 | 173 | 0 | 1.43 | 1.16 | 0.87 |
| 100 | 500 | 5 | – | 8,277,027.11 | 1552 | 306 | 0 | 1.16 | 0.81 | 0.60 |
| | | | | | | | | 0.88 | 0.56 | 0.31 |

The time gap among instances with $\lambda=0.1$ and $\lambda=0.8$ is larger already for $m=200$. Similar trends can be observed for $n=100$ instances.

HELP2 has a random structure thus we run the algorithm five times for each instance. Average results are presented in Tables 3 and 4 (one table for each class of instances). Each table is divided into two parts, one for each $\lambda$ value. For each instance (uniquely identified by the number of suppliers $n$, the number of products $m$ and the number of instance #), the following columns are shown:

- $v^*$, $t^*$: the best known solution value (possibly the optimal one) and its computational time as obtained from the literature. The symbol "–" used in column $t^*$ means that the time limit was reached or that the solver went out of memory. In both cases the corresponding solution value is not proved to be optimal;
- $t$: average computational time in seconds required by HELP2 on 5 trials;
- *init*: percentage gap of the initial solution from the value of the best known solution;
- *max*: max percentage gap from the value of the best known solution;
- *avg*: average percentage gap on 5 trials from the value of the best known solution;
- *min*: minimum percentage gap from the value of the best known solution.

Percentage gaps are computed as $(x-v^*)\times 100/v^*$, where $v^*$ is the best known solution value for the current instance and $x$ is the upper bound provided by the heuristic. If the solution value found by HELP2 is equal or better than the current best known value (i.e. the percentage gap is equal or less than 0) we have highlighted *max*, *avg* and *min* percentages in bold font. Aside these average results, in Table 5 we summarize for each instance the number of times (out of 5 trials) HELP2 reaches or improves the best known solution value.

A first sight analysis shows that Class 2 instances are more difficult to solve than Class 1, and $\lambda=0.1$ instances are harder than $\lambda=0.8$ ones. Considering Class 1, HELP2 has a good performance: for 58 instances out of 80, this method finds the optimal solution in at least one trial, and in 11 among these 58 instances, it obtains the optimal solution in all the five trials. In all these cases computational times for HELP2 are smaller than the optimal time $t^*$ required by the exact approach. When no optimal solution is available, gaps from the best known values are modest anyway, sometimes negligible. Indeed, the worst percentage gaps never exceed 0.74% for $\lambda=0.1$ and 3.33% for $\lambda=0.8$, and the maximum average gaps are 0.13% for $\lambda=0.1$ and 0.51% for $\lambda=0.8$.

Results for Class 2 are less impressive, but still quite good. Here, HELP2 is able to reach or improve best known values in at least one trial for 23 instances out of 80, and for 3 instances among these 23, the heuristic reaches this goal for all the five trials. In all these cases computational times are quite lower than $t^*$ values. In this set of instances, the worst percentage gap reaches 4.01% for $\lambda = 0.1$ and is equal to 5.24% for $\lambda = 0.8$, whereas the maximum average gap is equal to 1.54% for $\lambda = 0.1$ and to 2.05% for $\lambda = 0.8$, respectively. Average gaps for the two values of $\lambda$ settle around to 1%, anyway. Hence, the heuristic has some discontinuous performance for Class 2 instances, mostly for the hardest ones (100 suppliers), but if we compare computational times HELP2 is extremely efficient with respect to times required to get optimal solutions. As far as the 37 open instances are concerned, HELP2 has been able to improve the best known values (negative percentage gaps in the tables) in 4 instances and its computational time never exceeds 1382 s.

Since in Class 1 average percentage errors found by HELP2 are almost null, we decided to test the proposed ILP refinement procedure only on the most difficult instances of Class 2. At this aim, we repeated the computational tests for all Class 2 (that is obviously the critical one and that includes several non-optimally solved instances) adding to HELP2 the refinement procedure (as already mentioned we call this extended procedure HELP2+). Also in this case we compute 5 trials for each instance. According to some preliminary experiments, we have set $l = 5$ and $\gamma = 0.25$, and in order to compare the results obtained with those of plain HELP2 we left unchanged the parameters setting as in the non-refined version. Average results are presented in Tables 6 and 7 that show, in addition to already explained entries, the average time (on 5 trials) used by the whole procedure ($t_{tot}$) and by the sole refinement ($t_+$), and the number of times out of the 5 trials HELP2+ reaches or improves the best known values (*#impr*).

It is evident how the proposed refinement algorithm results to be extremely effective, strictly improving the standard HELP2's solution values in about the 75% of cases. Most of the remaining cases correspond to instances with $\lambda = 0.8$ and $n = 50$, where HELP2 was frequently able to find the optimal solution, and thus no room for improvements was available. Moreover, computational times required by the refinement procedure are very small (or reasonably proportional) with respect to the total procedure times. Looking at the solution quality, we can see that the results are better in general. The number of optimal solutions found has increased, as well as the number of improvements with respect of the best known solution values (in particular, for 7 instances a new best solution has been found). Percentage gaps for HELP2+ are definitely smaller than those found by HELP2 (compare the last

**Table 7**
HELP2+ results for Class 2 and $\lambda = 0.8$ instances.

| $n$ | $m$ | # | Best known | | HELP2+ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $t^*$ | $v^*$ | $t_{tot}$ | $t_+$ | #impr | max | avg | min |
| 50 | 100 | 1 | 526 | 209,340.00 | 203 | 2 | 4 | 2.65 | 0.53 | **0.00** |
| 50 | 100 | 2 | 383 | 221,929.00 | 204 | 2 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 100 | 3 | 419 | 204,096.50 | 204 | 3 | 3 | 0.65 | 0.26 | **0.00** |
| 50 | 100 | 4 | 450 | 233,650.00 | 203 | 2 | 3 | 0.58 | 0.23 | **0.00** |
| 50 | 100 | 5 | 328 | 208,702.61 | 204 | 2 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 200 | 1 | 1254 | 425,279.00 | 407 | 3 | 4 | 0.31 | 0.06 | **0.00** |
| 50 | 200 | 2 | 1628 | 470,957.00 | 408 | 5 | 2 | 0.94 | 0.50 | **0.00** |
| 50 | 200 | 3 | 1084 | 387,964.50 | 405 | 3 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 200 | 4 | 1676 | 405,405.50 | 405 | 3 | 4 | 0.08 | 0.02 | **0.00** |
| 50 | 200 | 5 | 663 | 416,658.50 | 404 | 3 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 1 | 2841 | 632,539.00 | 607 | 4 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 2 | 6981 | 599,533.50 | 611 | 6 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 3 | 2762 | 657,022.00 | 608 | 4 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 4 | 3760 | 624,807.00 | 609 | 4 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 300 | 5 | 4418 | 631,879.00 | 608 | 4 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 500 | 1 | – | 1,172,101.50 | 1016 | 7 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 500 | 2 | 11,830 | 978,685.50 | 1017 | 9 | 3 | 0.63 | 0.22 | **0.00** |
| 50 | 500 | 3 | – | 1,178,217.00 | 1011 | 9 | 5 | **0.00** | **0.00** | **0.00** |
| 50 | 500 | 4 | – | 1,135,915.50 | 1018 | 7 | 5 | **−0.25** | **−0.25** | **−0.25** |
| 50 | 500 | 5 | – | 1,069,179.00 | 1019 | 9 | 5 | **−0.16** | **−0.16** | **−0.16** |
| 100 | 100 | 1 | 1553 | 364,230.50 | 406 | 5 | 3 | 0.93 | 0.27 | **0.00** |
| 100 | 100 | 2 | 3290 | 383,295.50 | 411 | 9 | 0 | 1.00 | 0.68 | 0.41 |
| 100 | 100 | 3 | 1774 | 367,575.50 | 409 | 9 | 1 | 1.61 | 0.76 | **0.00** |
| 100 | 100 | 4 | 1729 | 387,625.50 | 410 | 9 | 0 | 2.06 | 1.59 | 0.01 |
| 100 | 100 | 5 | 1874 | 374,945.00 | 411 | 10 | 0 | 2.83 | 1.72 | 0.75 |
| 100 | 200 | 1 | 11,584 | 708,833.50 | 622 | 19 | 0 | 1.56 | 1.13 | 0.80 |
| 100 | 200 | 2 | – | 748,779.00 | 628 | 22 | 2 | 0.54 | 0.14 | **−0.15** |
| 100 | 200 | 3 | – | 775,076.00 | 637 | 34 | 0 | 2.24 | 1.96 | 1.77 |
| 100 | 200 | 4 | 5004 | 768,933.00 | 622 | 14 | 0 | 2.83 | 2.18 | 0.76 |
| 100 | 200 | 5 | – | 765,640.00 | 623 | 21 | 0 | 1.22 | 0.71 | 0.35 |
| 100 | 300 | 1 | – | 1,096,232.00 | 836 | 30 | 0 | 3.73 | 1.17 | 0.14 |
| 100 | 300 | 2 | – | 1,164,793.00 | 850 | 41 | 0 | 0.52 | 0.34 | 0.11 |
| 100 | 300 | 3 | – | 1,158,439.50 | 838 | 30 | 2 | 2.99 | 0.89 | **−0.16** |
| 100 | 300 | 4 | – | 1,187,005.00 | 839 | 18 | 5 | **−0.38** | **−0.66** | **−0.81** |
| 100 | 300 | 5 | – | 1,214,561.00 | 833 | 31 | 0 | 0.59 | 0.34 | 0.08 |
| 100 | 500 | 1 | – | 1,821,572.00 | 1268 | 53 | 0 | 3.45 | 2.23 | 0.23 |
| 100 | 500 | 2 | – | 1,827,191.50 | 1283 | 59 | 0 | 1.77 | 1.01 | 0.18 |
| 100 | 500 | 3 | – | 1,917,578.50 | 1278 | 53 | 5 | **−1.33** | **−2.01** | **−3.59** |
| 100 | 500 | 4 | – | 1,888,163.50 | 1274 | 51 | 3 | 1.54 | 0.24 | **−0.38** |
| 100 | 500 | 5 | – | 1,851,845.00 | 1286 | 65 | 1 | 1.05 | 0.56 | **−0.08** |
| | | | | | | | | 0.90 | 0.42 | 0.00 |

**Table 8**
ILP refinement contribution.

| $n$ | $m$ | # | $\lambda = 0.1$ | | | $\lambda = 0.8$ | | |
|---|---|---|---|---|---|---|---|---|
| | | | $\Delta$(%) | Part I | Part II | $\Delta$(%) | Part I | Part II |
| 50 | 100 | 1 | 0.82 | 3 | 2 | 0.00 | 0 | 0 |
| 50 | 100 | 2 | 1.08 | 4 | 1 | 0.00 | 0 | 0 |
| 50 | 100 | 3 | 0.87 | 4 | 1 | 0.13 | 1 | 0 |
| 50 | 100 | 4 | 1.17 | 2 | 3 | 0.00 | 0 | 0 |
| 50 | 100 | 5 | 0.84 | 4 | 1 | 0.00 | 0 | 0 |
| 50 | 200 | 1 | 0.72 | 2 | 2 | 0.00 | 0 | 0 |
| 50 | 200 | 2 | 0.60 | 3 | 2 | 0.06 | 1 | 0 |
| 50 | 200 | 3 | 0.98 | 5 | 0 | 0.00 | 0 | 0 |
| 50 | 200 | 4 | 0.96 | 4 | 1 | 0.00 | 0 | 0 |
| 50 | 200 | 5 | 0.92 | 3 | 2 | 0.00 | 0 | 0 |
| 50 | 300 | 1 | 0.64 | 3 | 1 | 0.00 | 0 | 0 |
| 50 | 300 | 2 | 0.49 | 5 | 0 | 0.00 | 0 | 0 |
| 50 | 300 | 3 | 1.01 | 3 | 2 | 0.00 | 0 | 0 |
| 50 | 300 | 4 | 0.63 | 4 | 1 | 0.00 | 0 | 0 |
| 50 | 300 | 5 | 1.17 | 4 | 1 | 0.00 | 0 | 0 |
| 50 | 500 | 1 | 1.26 | 3 | 2 | 0.00 | 0 | 0 |
| 50 | 500 | 2 | 0.83 | 3 | 2 | 0.00 | 0 | 0 |
| 50 | 500 | 3 | 0.68 | 3 | 1 | 0.07 | 1 | 0 |
| 50 | 500 | 4 | 0.61 | 4 | 1 | 0.05 | 1 | 0 |
| 50 | 500 | 5 | 0.46 | 3 | 2 | 0.03 | 2 | 0 |
| 100 | 100 | 1 | 1.35 | 5 | 0 | 0.43 | 4 | 0 |
| 100 | 100 | 2 | 1.36 | 4 | 1 | 0.91 | 5 | 0 |
| 100 | 100 | 3 | 1.00 | 2 | 3 | 1.86 | 5 | 0 |
| 100 | 100 | 4 | 1.81 | 3 | 2 | 1.55 | 5 | 0 |
| 100 | 100 | 5 | 1.77 | 3 | 2 | 1.47 | 5 | 0 |
| 100 | 200 | 1 | 1.07 | 3 | 2 | 0.75 | 5 | 0 |
| 100 | 200 | 2 | 1.13 | 4 | 1 | 0.22 | 4 | 0 |
| 100 | 200 | 3 | 1.05 | 4 | 1 | 1.02 | 5 | 0 |
| 100 | 200 | 4 | 1.69 | 4 | 1 | 1.04 | 4 | 0 |
| 100 | 200 | 5 | 1.48 | 2 | 3 | 0.89 | 5 | 0 |
| 100 | 300 | 1 | 1.31 | 2 | 3 | 0.45 | 4 | 0 |
| 100 | 300 | 2 | 0.62 | 0 | 5 | 0.86 | 5 | 0 |
| 100 | 300 | 3 | 1.52 | 1 | 4 | 1.06 | 5 | 0 |
| 100 | 300 | 4 | 1.22 | 0 | 5 | 0.43 | 5 | 0 |
| 100 | 300 | 5 | 1.46 | 1 | 4 | 0.54 | 5 | 0 |
| 100 | 500 | 1 | 1.10 | 0 | 4 | 0.60 | 5 | 0 |
| 100 | 500 | 2 | 0.47 | 0 | 5 | 1.21 | 5 | 0 |
| 100 | 500 | 3 | 0.97 | 0 | 5 | 0.43 | 5 | 0 |
| 100 | 500 | 4 | 1.10 | 0 | 5 | 0.33 | 3 | 0 |
| 100 | 500 | 5 | 1.00 | 0 | 5 | 0.69 | 5 | 0 |
| Avg: | | | 1.03 | | | 0.43 | | |
| Sum: | | | | 107 | 89 | | 100 | 0 |

**Table 9**
Heuristics comparison on real case instances.

| $n$ | $m$ | # | $v^*$ | greedy% | HELP2% | HELP2+% |
|---|---|---|---|---|---|---|
| 20 | 100 | 1 | 323,518.50 | 39.77 | 0.00 | 0.00 |
| 20 | 100 | 2 | 275,461.74 | 37.71 | 0.00 | 0.00 |
| 20 | 100 | 3 | 277,815.00 | 33.83 | 0.00 | 0.00 |
| 20 | 100 | 4 | 360,279.00 | 43.20 | 0.35 | 0.00 |
| 20 | 100 | 5 | 362,353.30 | 32.98 | 0.00 | 0.00 |
| 20 | 100 | 6 | 297,981.69 | 39.46 | 0.00 | 0.00 |
| 20 | 100 | 7 | 312,408.13 | 21.42 | 0.00 | 0.00 |
| 20 | 100 | 8 | 333,121.00 | 40.19 | 0.56 | 0.56 |
| 20 | 100 | 9 | 320,630.48 | 34.76 | 0.53 | 0.34 |
| 20 | 100 | 10 | 326,619.00 | 40.69 | 0.00 | 0.00 |
| | | | | 36.40 | 0.14 | 0.09 |

line of Table 4 with that of Tables 6 and 7). Indeed, for $\lambda = 0.1$ instances gaps have been halved (from 1.54, 1.03 and 0.61 to 0.88, 0.56, and 0.31 for *max*, *avg* and *min* respectively), while for $\lambda = 0.8$ instances gaps have been further reduced from 2.05, 1.36 and 0.68 to 0.90, 0.42 and 0.00, respectively. Notice that these gaps never exceed 1%.

Finally, in Table 8 we analyze the contribution provided by the ILP refinement. Column $\Delta(\%)$ reports the average percentage improvement with respect to plain HELP2, columns *Part I* and *Part II* show the number of time out of 5 trials each part has determined the best solution value. For $\lambda = 0.1$ the two parts gets the best solution in almost the same number of trials. On the contrary for $\lambda = 0.8$ Part II of the refinement is not effective at all, even if it should be noticed that in many instances HELP2 already found the optimal solution.

To conclude, both HELP2 and HELP2+ provide a robust behaviour also when the complexity of instances increases (larger number of products and suppliers are considered). Moreover, HELP2+ comes out to be extremely effective outperforming basic HELP2 in the most difficult instances, while maintaining the efficiency of the standard version.

### 5.1. Real case instances

We also test our algorithms on a set of 10 real case instances obtained by a consortium involved with a procurement problem corresponding to a CTQDP. The performance of HELP2 and HELP2+ is compared to a simple greedy heuristic adopted by the consortium in practice. This procedure, that simply buy products in the markets offering them at the lowest prices, and the real instances are described in [24]. In Table 9 we can see how HELP2+ has an average gap of 0.09% with respect to the optimal solution, within the time limit of 60 seconds, whereas the greedy heuristic terminates with an average gap larger than 36%. Our method would allow to the consortium an average saving of about 116,000 Euros.

## 6. Conclusions

In this paper an effective matheuristic is proposed to solve the CTQDP problem. The algorithm, called HELP2, combines a VNDS-like meta-heuristic structure with the exact solution of MILP subproblems. The method, tested on a large set of benchmark instances, is shown to be very effective (reaching the best known value in 77 instances and improving it in 4 instances) and quite efficient. A further version of this algorithm (HELP2+), including an ILP refinement, is also tested. HELP2+ has been able to strongly improve the solutions found by HELP2 for the hardest instances, without loosing in efficiency with respect to the basic version.

## Acknowledgements

The authors are grateful to two anonymous referees for their valuable comments.

## References

[1] Alvarenga G, Mateus G, de Tomi G. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. Computers & Operations Research 2007;34(6):1561–84.
[2] Angelelli E, Mansini R, Speranza M. Kernel search: a general heuristic for the multi-dimensional knapsack problem. Computers and Operations Research 2010;37:2017–26.
[3] Angelelli E, Mansini R, Speranza M. Kernel search: a new heuristic framework for portfolio selection. Computational Optimization and Applications 2012;51(1):345–61.
[4] Angelelli E, Mansini R, Vindigni M. Exploring greedy criteria for the dynamic traveling purchaser problem. Central European Journal of Operations Research 2009;17:141–58.
[5] Angelelli E, Mansini R, Vindigni M. Look-ahead heuristics for the dynamic traveling purchaser problem. Computers and Operations Research 2011;38:1867–76.
[6] Awasthi A, Chauhan S, Goyal S, Proth J-M. Supplier selection problem for a single manufacturing unit under stochastic demand. International Journal of Production Economics 2009;117:229–33.
[7] Benton WC. Quantity discount decisions under conditions of multiple items. International Journal of Production Research 1991;29:1953–61.
[8] Burke GJ, Carrillo J, Vakharia AJ. Heuristics for sourcing from multiple suppliers with alternative quantity discounts. European Journal of Operational Research 2008;186(1):317–29.
[9] Chauhan S, Proth J-M. The concave cost supply problem. European Journal of Operational Research 2003;148:374–83.
[10] Danna E, Rothberg E, Pape CL. Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming 2005;102(1):71–90.
[11] De Franceschi R, Fischetti M, Toth P. A new ILP-based refinement heuristic for vehicle routing problems. Mathematical Programming 2006;105:471–99.
[12] Dickson GW. An analysis of vendor selection systems and decisions. Journal of Purchasing 1966;2:5–17.
[13] Doerner K, Schmid V. Survey: matheuristics for rich vehicle routing problems. In: Blesa M, Blum C, Raidl G, Roli A, Sampels M, editors. Hybrid Metaheuristics. Lecture Notes in Computer Science, vol. 6373. Berlin/Heidelberg: Springer; 2010. p. 206–21.
[14] Fischetti M, Lodi A. Local branching. Mathematical Programming 2003;98:23–47.
[15] Framinan JM, Pastor R. A proposal for a hybrid meta-strategy for combinatorial optimization problems. Journal of Heuristics 2008;14:375–90.
[16] French A, Robinson A, Wilson J. Using a hybrid genetic-algorithm/branch and bound approach to solve feasibility and optimization integer programming problems. Journal of Heuristics 2001;7:551–64.
[17] Glover F, Parker M, Ryan J. Coloring by tabu branch & bound. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 1996;26:285–307.
[18] Goossens D, Maas A, Spieksma F, van de Klundert J. Exact algorithms for a procurement problem with total quantity discounts. European Journal of Operational Research 2007;178(2):603–26.
[19] Güder F, Zydiak J, Chaudhry S. Capacitated multiple item ordering with incremental quantity discounts. Journal of the Operational Research Society 1994;45(10):1197–205.
[20] Hansen P, Mladenovi N, Perez D. Variable neighborhood decomposition search. Journal of Heuristics 2001;7:335–50.
[21] Haouari M, Ladhari T. A branch-and-bound based local search method for the flow shop problem. Journal of the Operational Research Society 2003;54:1076–84.
[22] Hou J, Su D. Integration of web services technology with business models within the total product design process for supplier selection. Computers in Industry 2006;57(8–9):797–808.
[23] Labadie N, Mansini R, Melechovsky` J, Calvo RW. The team orienteering problem with time windows: an LP-based granular variable neighborhood search. European Journal of Operational Research 2012;220:15–27.
[24] Manerba D, Mansini R. An exact algorithm for the capacitated total quantity discount problem. European Journal of Operational Research 2012;222(2):287–300.
[25] Maniezzo V, Stützle T, Voss S. MatheuristicsHybridizing Metaheuristics and Mathematical Programming. Annals of Information Systems. Heidelberg: Springer; 2009.
[26] Mansini R, Savelsbergh MVP, Tocchella B. The supplier selection problem with quantity discounts and truckload shipping. Omega 2012;40:445–55.
[27] Mansini R, Tocchella B. The traveling purchaser problem with budget constraint. Computers & Operations Research 2009;36:2263–74.
[28] Nagar A, Heragu S, Haddock J. A combined branch and bound and genetic algorithm based for a flowshop scheduling algorithm. Annals of Operations Research 1996;63:397–414.

[29] Riera-Ledesma J, Salazar-Gonzàlez J. A heuristic approach for the travelling purchaser problem. European Journal of Operational Research 2005;162 (1):142–52.
[30] Subramanian A, Penna PHV, Uchoa E, Ochi LS. A hybrid algorithm for the heterogeneous fleet vehicle routing problem. European Journal of Operational Research 2012;221(2):285–95.
[31] Talbi E-G. A taxonomy of hybrid metaheuristics. Journal of Heuristics 2002;8:541–64.
[32] Toth P, Tramontani A. An integer linear programming local search for capacitated vehicle routing problems. In: Golden B, Raghavan S, Wasil E, editors. The Vehicle Routing ProblemLatest Advances and New Challenges Operations Research/ Computer Science Interfaces. Springer US; 2008. p. 275–95.
[33] Voss S. Dynamic tabu search strategies for the traveling purchaser problem. Annals of Operations Research 1996;63:253–75.
[34] Weber CA, Current JR, Benton WC. Vendor selection criteria and methods. European Journal of Operational Research 1991;50:2–18.
[35] Woodcock A, Wilson J. A hybrid tabu search/branch & bound approach to solving the generalized assignment problem. European Journal of Operational Research 2010;207(2):566–78.