ELSEVIER European Journal of Operational Research 101 (1997) 519-531

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

A multi-product loading problem: a model and solution method

Umit Yuceer

Faculty of Business Administration, Bilkent University, Ankara, Turkey

Received 15 July 1995; accepted 23 May 1996

Abstract

An important operational problem arises during the transportation and delivery of several products, which cannot be mixed, in the same vehicle at regular intervals. The vehicle has compartments to keep the products separately. Therefore, a scheme of allocation of compartments which we call vehicle loading problem to maximize the efficiency of the system while the demands for the products at the destination(s) are satisfied. A mixed binary model is developed for this multi-product loading problem. The solution method is based on simultaneously exploring the primal and dual structures derived from the Lagrangian relaxation. Subset sum problems are obtained as subproblems to the partial Lagrangian. An algorithm is developed and its convergence is proved. The efficiency of the method is demonstrated by running randomly chosen test problems. An initial solution finding method is also developed. © 1997 Elsevier Science B.V.

Keywords: Mathematical programming; Lagrangian relaxation; Distribution; Set partitioning

1. Introduction

A vehicle transports and delivers a number of products, that cannot be mixed, from a source to a destination. The products have constant demand rates at the destination. Simultaneous depletion of the products is accomplished by delivering the correct proportions. The time interval between two consecutive deliveries is called the replenishment time. The replenishment time is very much interrelated to the delivery quantities and conversely. The vehicle has compartments built to keep the products separated. The operational problem is then to allocate the compartments to the products in such a way to maximize the replenishment time for a given vehicle capacity and compartment sizes. This will maximize the capacity utilization, and therefore the efficiency of the delivery system.

An example of such an operation is the transportation and the delivery of petroleum products from a refinery to a number of regional distribution centers (depots). The products in this example are the regular gasoline, premium gasoline, aviation gasoline, kerosene, and diesel fuel. The vehicles may be tanker-trucks or sea tankers. Ronen (1995) points out that dispatching petroleum products may involve transportation and product characteristics, and operating rules of transportation units, and may also include the use of the vehicles with compartments.

During the transportation, different liquids or the chemical compounds in a single vehicle are not allowed to be mixed. Mixing of chemical compounds produces an undesirable product or worse causes catastrophic damages. A more practical example is the case of transporting premium gasoline in the same tank with regular gasoline; mixing results in a different product for which there is no use in the market. The same problem exists for stationary storage facilities, warehouses, etc. To avoid disasters, dangers, and economic losses the storage facilities are divided into smaller cabins, rooms, compartments, tanks to guard against different products coming into contact with each other.

The destination may be a set of destinations with aggregated demand rates. Determining the destination(s) to be supplied in a single trip of a given vehicle is investigated by Yuceer and Dogrusoz (1994). In this study, it is assumed that the destination(s) to be supplied by a given vehicle is given or predetermined. There is sufficient storage capacity at the destination(s) for every product, hence storage capacities do not form a constraint. Further, the source is capable of producing and/or supplying every product in required quantities at all times. The demand rates of the products at the destination(s), assumed given, are constant. The destination(s) agrees to the delivery schedule determined by the management of such a transportation and delivery operation as long as the demand for each product is met at all times.

Since mixing the products is not allowed, this complicates finding an optimal assignment of the compartments by making it a combinatorial problem. The problem is then to determine which compartments, cabins, rooms to assign to each product so that the time interval between two consecutive deliveries is maximum for an efficient operation of the delivery system. This problem may be called as a multi-product loading (of vehicles) or multi-product storing (at storage facilities) problem. This problem can also be viewed as a max-min allocation problem. Basically it can be modeled as a mixed 0-1 integer (binary) programming problem. The main thrust of this research is to develop a model and to find an efficient solution method to solve problems of this type. Solving the problems of this type by branch-and-bound method easily deteriorates into complete enumeration because of the special structure of the constraints in the problem.

Similar type of loading problems have been attempted before by Christofides et al. (1976) and Neebe et al. (1977). There are m liquids, that cannot be mixed, to be loaded into n tanks so that the total profit from these products is maximized. They modeled the problem as a multiple knapsack problem and solved accordingly. In contrast, the problem presented in this article tackles an operational problem and maximizes the replenishment time.

Tang (1988) describes a class of max-min allocation problems and provides a list of application areas in manufacturing and production. He proposes a non-simplex based algorithm which finds the optimum in $O(mn^2)$ operations. The model described in this article is quite different from his model, since, contrary to his assumption, it is not known in advance which compartments will be assigned to each product.

This problem can also be viewed as a set partitioning problem (but not a structured partitioning problem), since a number of compartments cabins, or rooms will be assigned to each product, and furthermore only one product can occupy each compartment. Set covering and set partitioning problems are binary programming problems. Fisher and Kedia (1990) present an algorithm for a mixed set covering and set partitioning model. They also provide a summary of the published research on these topics. Marsten (1974) (Fisher and Kedia as well) notices that the general strategy in solving the set covering/set partitioning problems relies on solving the linear programming relaxation. Fisher and Kedia use the dual of linear programming relaxation to provide lower bounds for a branch-and-bound algorithm. They also use the subgradient method to improve the Lagrangian bounds for the set partitioning problem.

A Lagrangian relaxation approach is recommended in the literature in solving the Generalized Assignment Algorithm (GAP). There exist numerous articles on Lagrangian relaxation in general and on GAP in particular. A few of those are cited here. Barcia and Jorsten (1990) combine the Lagrangian decomposition and bound improving sequences in converging to optimal. Gavish and Pirkul (1991) provide several different relaxations to Multi-Resource Generalized Assignment Problem and report an efficient branch-and-bound algorithm.

A general approach in solving mixed integer programming problems is Cross-Decomposition method proposed by Van Roy(1983) and implemented by Van Roy (1986). Cross-decomposition method exploits simultaneously both primal and dual structures of the relaxed Lagrangian. Computational experience with this approach has shown its efficiency. Holmberg (1994) investigates obtaining good lower bounds for the optimal objective function value of linear, pure integer programming problems by cross-decomposition.

Section 2 presents a mixed binary model for the multi-product loading problem. Dynamic programming solves this problem but lacks the efficiency. Another solution method of Section 3 is based on the strategy of cross-decomposition method in developing a solution method for the multi-product loading problem. The proposed algorithm however is quite different from the cross-decomposition algorithm. The algorithm generates subsetsum problems as subproblems. Each subsetsum problem can be solved by dynamic programming, branch and bound, exhaustive search or any other method in the literature. Subsetsum problems belong to a special class of knapsack problems, since each variable has exactly the same coefficient in the objective function and in the constraint. Specialized techniques are discussed by Martello and Toth (1987) for knapsack problem in general and subsetsum problems as a special case. Solution to each subsetsum problem yields the minimum required capacity allocation for a given replenishment time. A feasible compartment combination is sought by solving these subproblems. Based on a feasible solution obtained from these subproblems, a dual linear problem is solved to obtain a better feasible solution or an upper bound for the replenishment time. The algorithm keeps the lower bound of the interval on the replenishment time at a feasible solution and reduces the upper bound. Finding a new feasible solution reduces the interval by increasing the lower bound. The process is repeated until no more progress is possible and a final interval of uncertainty is obtained. An exhaustive search, developed from the primal structures of the partial Lagrangian, is performed in the final interval of uncertainty to obtain the optimal solution to the multi-product loading problem. The convergence of the algorithm in a finite number of steps is also proved. The efficiency of the algorithm is demonstrated by running randomly selected test problems. An illustration and computational results are given in Section 4. The case of equal size compartments and the results and findings of this research is summarized in the Section 5. In addition, a simple algorithm to obtain an initial solution is given in the Appendix.

2. A model for the multi-product loading problem

A vehicle with m compartments will transport n different products which are not allowed to be mixed from a source to a destination. It is assumed that there is no storage constraint at the destination. Feasibility requires $m \geq n$. The decision variables are given first.

t = the common replenishment time for all n products,

D_j = the delivery quantity of product $j \in J$,

$$x_{ij} = \begin{cases} 1 & \text{if compartment } i \text{ is assigned to the product } j, \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in I$ and $j \in J$, where $I = \{1, 2, \dots, m\}$ is the index set for the compartments, and $J = \{1, 2, \dots, n\}$ is the index set for the products.

The parameters of the problem, assumed constant, are given below.

q_i = the capacity of the compartment $i \in I$,

d_j = the demand rate of product $j \in J$.

Consequently the following relationships are obtained from these variables and/or parameters.

$D_j = td_j$ for all $j \in J$ is the delivery quantity,

$$A_j = \sum_{i \in I} q_i x_{ij}$$

is the total capacity of the compartments allocated to the product $j \in J$. $D_j \leq A_j$ for all $j \in J$. Then the cycle length to deplete quantity A_j is equal to $t_j = A_j/d_j = \sum_{i \in I} q_i/d_j \, x_{ij}$ for $j \in J$. The common replenishment time is obtained as $t = \min_{j \in J}\{t_j\}$. Hence the problem turns into the following.

$$\max t = \max \left\{ \min_{j \in J} \{t_j\} \right\} = \max \left\{ \min_{j \in J} \left\{ \sum_{i \in I} \frac{q_i}{d_j} x_{ij} \right\} \right\}.$$

If $\min_j \{(\sum_{i \in I} q_i x_{ij})/d_j\}$ is unique, then all the compartments will be assigned necessarily. If it is not unique, then the unassigned compartments (if any) can be assigned to some products arbitrarily since the objective is to maximize $\min_j \{(\sum_{i\in I} q_i x_{ij})/d_j\}$. In this case, alternating optimum solutions exist. In the simple example of a vehicle with three compartments $(q_i) = (60,50,30)$ and the demand rates $(d_j) = (10,12)$, the minimum ratio is not unique. $\min(50/10, 60/12) = 5$. Assigning the remaining comaprtment produces only alternating optimum solutions without increasing the replenishment time; $\min((50+30)/10, 60/12) = 5$ or $\min(50/10,(60+30)/12) = 5$. In order to secure that a compartment is allocated to one and only one product, the constraint $\sum_{j\in J} x_{ij} = 1$ for all $i \in I$ must be satisfied. Thus the decision model is stated as follows.

$$\max \min_{j\in J} \left\{ \sum_{i\in I} \frac{q_i}{d_j} x_{ij} \right\}, \tag{1}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \text{for all } i \in I, \tag{2}$$

$$x_{ij} = 0, 1 \quad \text{for all } i \in I \text{ and } j \in J. \tag{3}$$

The objective is clearly to maximize the common replenishment time. Subsequently, a more practical formulation is obtained as follows. Problem (P):

$$\max z = t, \tag{4}$$

subject to

$$td_j - \sum_{i\in I} q_i x_{ij} \le 0 \quad \text{for all } j \in J, \tag{5}$$

$$\sum_{j\in J} x_{ij} = 1 \quad \text{for all } i \in I,$$

$$x_{ij} = 0, 1 \quad \text{for all } i \in I \text{ and } j \in J,$$

$$t \ge 0. \tag{6}$$

The multi-product loading problem is basically a special class of set partitioning problem, but not a structured partitioning problem, since the partitioning is not done according to the ordering or the ranking of the compartments. A set of m objects (compartments) will be partitioned into n (number of products) non-empty subsets. Since each subset of compartments can be used to carry one product, the number of all possible ways of assigning m compartments into n products is given by $n!S_m^{(n)}$ where $S_m^{(n)}$ is a Stirling number of the Second Kind and represents the number of ways of partitioning a set of m objects into n non-empty subsets, (Abramowitz, and Stegun (1970)). For instance, the number of ways of assigning 7 compartments into 4 products is $4!S_7^{(4)} = 24 * 350 = 8400$.

3. Solution methods

The problem described by the expressions (4), (5), (2), (3) and (6) can be solved by dynamic programming. The state variable is defined by the vector $U = (u_1, u_2, \dots, u_m)$ where each

$$u_i = \begin{cases} 1 & \text{if compartment } i \text{ is assigned,} \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in I$. Further, $\|U\| = \sum_{i\in I} u_i$ is the number of assigned compartments. The number of state variables is $2^m - 1$ excluding the zero vector. The stage $j \in J$ of the dynamic programming corresponds to assigning products $1, 2, \dots, j$ to some compartments and is defined as follows.

Stage 1. All compartment combinations of the form $1 \le \|U\| \le m - (n-1)$ and $t_1(U^1) = \sum_{i\in I} q_i u_i^1 / d_1$.

Stage j. All compartment combinations of the form $j \le \|U\| \le m - (n-j)$ for $2 \le j < n$, $U^j = U^{j-k} + U^k$ and $t_j(U^j) = \min\{t_{j-1}(U^k), \sum_{i\in I} q_i u_i^{j-k}/d_j\}$ for $1 \le k \le (j-1)$.

Stage n. All the compartment combinations of the form $n \le \|U\| \le m$, $t_n(U^n) = \min\{t_{n-1}(U^{n-k}), \sum_{i\in I} q_i u_i^{n-k}/d_n)\}$ for $1 \le k \le (n-1)$.

Dynamic programming generates the exact solution of this problem, but it is very time consuming. An efficient solution method will be described next and the computational comparison of both methods will be discussed in the next section.

Consider the relaxed problem ($0-1$ requirement on x_{ij} is ignored), problem (RP);

$$\max z_R = t, \tag{7}$$

subject to

$$td_j - \sum_{i\in I} q_i x_{ij} \le 0 \quad \text{for all } j \in J, \tag{8}$$

$$\sum_{j\in J} x_{ij} = 1 \quad \text{for all } i \in I, \tag{9}$$

$$x_{ij} \geq 0 \quad \text{for all } i \in I, j \in J, \tag{10}$$

$$t \geq 0 \tag{11}$$

and its dual, problem (RD);

$$\min w_R = \sum_{i \in I} \lambda_i, \tag{12}$$

subject to

$$\sum_{j \in J} d_j \mu_j \geq 1, \tag{13}$$

$$-q_i \mu_j + \lambda_i \geq 0 \quad \text{for all } i \in I, j \in J, \tag{14}$$

$$\mu_j \geq 0 \quad \text{for all } j \in J, \tag{15}$$

$$\lambda_i \quad \text{for all } i \in I \text{ is unrestricted.} \tag{16}$$

The dual variables μ_j for $j \in J$ correspond to the set of constraints (8). The dual variables λ_i for $i \in I$ correspond to the set of constraints (9). The dual variable λ_i may be interpreted as the marginal contribution of compartment $i \in I$ to the replenishment time. Therefore, $\sum_{i \in I} \lambda_i$ is the total marginal contribution of all compartments, or briefly the replenishment time. The dual variable μ_j may be interpreted as the marginal value (in terms of time) of the capacity allocated per unit volume of product $j \in J$.

A feasible solution to the dual problem (RD) can be obtained very easily by setting $\mu_j = 1/\sum_{j \in J} d_j$ for all $j \in J$ and $\lambda_i = \mu_j q_i = q_i/\sum_{j \in J} d_j$ for all $i \in I$. The value of the dual objective function at this solution is equal to $w_R = \sum_{i \in I} q_i/\sum_{j \in J} d_j$. On the other hand, the optimal solution to the relaxed problem (RP) is equal to $z_R = \sum_{i \in I} q_i/\sum_{j \in J} d_j$. This makes sense, because the total capacity would be utilized fully if the mixing of products were allowed. Since $w_R = z_R$, this dual solution is optimal. The value of z_R constitutes an upper bound on the value of objective function of (P). If there exists a feasible solution (t, X) such that $t = z_R$, then it is optimal. Since the mixing of products is not allowed, then there is a loss in capacity utilization (not all compartments are fully loaded) implies $t^* < z_R$.

The following partial Lagrangian function is considered for the solution of (P) and is called the problem (LPP(λ)).

$$\max \mathcal{L}(t, X, \lambda) = t + \sum_{i \in I} \lambda_i \left(1 - \sum_{j \in J} x_{ij} \right), \tag{17}$$

subject to (5), (3) and (6).

The partial Lagrangian (17) can be decomposed into the following subproblem for a given t and (λ) and each one will be called problem (SP(t)).

$$\min \sum_{j \in J} \sum_{i \in I} \lambda_i x_{ij}, \tag{18}$$

subject to

$$\sum_{i \in I} q_i x_{ij} \geq t d_j \quad \text{for all } j \in J \tag{19}$$

and (3).

This problem can further be decomposed into n knapsack problems (SP$_j(t)$) of the type, for each $j \in J$,

$$\min \sum_{i \in I} \lambda_i x_{ij}, \tag{20}$$

subject to (3) and (19).

The knapsack problems turn into subsetsum problems by setting $\lambda_i = q_i$ as the result of the relationship between the dual variables (λ_i) and the compartment sizes (q_i) of the relaxed problem. Each subsetsum problem can be solved by an exhaustive search, dynamic programming or branch-and-bound and produces a solution to the partial Lagrangian (LPP(λ)) of Expression (17). Solving each SP$_j(t)$ yields the minimum capacity allocation requirement for each product $j \in J$. Further, solving each SP$_j(t)$ may produce several alternating solutions, among those a feasible solution (X) is sought by trying all possibilities. A feasible solution (t, X) has the property that the total capacity allocated to each product will be at least equal to the respective minimum requirement determined by solving each SP$_j(t)$ (stated below formally as a Lemma). If such a solution (X), satisfying the expression (2) exists, then a feasible solution to (P) is obtained.

Lemma 1. *Let $(A_1, A_2, \ldots, A_n)$ be the minimum required capacity allocations determined by solving each SP$_j(t)$ for a given t. $A_j = \sum_{i \in I} q_i x'_{ij}$ for each $j \in J$ is an assignment of compartments without regard to the feasibility requirement of expression (2). Further, let j^* be the index such that $A_{j^*} = t d_{j^*}$ and $A_j > t d_j$ for all $j \neq j^*$. Then if there is a feasible solution (t, X), then $A_{j^*} = \sum_{i \in I} q_i x_{ij^*}$ and $A_j \leq \sum_{i \in I} q_i x_{ij}$ for all $j \neq j^*$.*

This lemma expresses the fact that the minimum allocation requirements on the products except the dominant product need to be relaxed for a solution of the general problem (P).

Corollary 2. *For any sufficiently small $\epsilon > 0$, $t' = t + \epsilon$ implies that the minimum requirement A_{j^*} must be increased by resolving $SP_{j^*}(t')$, the other A_j's will remain the same.*

Corollary 3. *Let $(A_1, A_2, \dots, A_n)$ correspond to minimum capacity allocation requirements for a given t. $A_{j^*} = td_{j^*}$. Let A'_{j^*} be the minimum capacity allocation which exceeds A_{j^*}. Then $(A_1, A_2, \dots, A'_{j^*}, \dots, A_n)$ corresponds to the minimum capacity allocation requirements for a replenishment time t' where $t' = \min\left(A'_{j^*}/d_{j^*}, \min\{A_j/d_j \mid j \neq j^*\}\right)$. Further, there are no solutions (t_1, X_1) such that $t < t_1 < t'$.*

The second term of Eq. (17),

$$S = \sum_{i\in I} q_i / \sum_{j\in J} d_j (1 - \sum_{j\in J} x_{ij})$$

may be interpreted as the potential marginal contribution of the solution (X) to the replenishment time. If the solution is feasible to (P) for a given (λ), then S is zero. It gives a measure of infeasibility when the solution is infeasible. If $S < 0$, then the total capacity requirement of the solution (t, X) exceeds C (over utilization of the compartments), then the potential marginal contribution of the solution (X) to the replenishment time is negative and t should be decreased. On the other hand, if $S > 0$, then the total capacity requirement of the solution (t, X) is less than C (under utilization), then the potential marginal contribution of the solution (X) is positive and t should be increased. If $S = 0$ or negligible in magnitude for an infeasible solution, then no useful information can be derived about the marginal contribution.

Lemma 4. *Let (t, X) correspond to an infeasible solution (constraint set (4) is not satisfied) and (t', X') correspond to a feasible solution such that $t < t'$, then $S > 0$.*

Proof. The delivery quantities $td_j < t'd_j$ for all $j \in J$ implies that the compartments allocated for the products should satisfy the following relation for all $j \in J$, and at least one of them is a strict inequality. If all are equal, then it means a different combination of compartment allocations provides a feasible solution which is contrary to the assumption. Summing over all $j \in J$ yields that $\sum_{i\in I}\sum_{j\in J} q_i x_{ij} < \sum_{i\in I}\sum_{j\in J} q_i x'_{ij}$. Total capacity required by the solution (t, X) is strictly less than that of (t', X') when $t < t'$. Multiplying by (-1) and adding $\sum_{i\in I} q_i / \sum_{j\in J} d_j$ to both sides yields the following relation.

$$\sum_{i\in I} \frac{q_i}{\sum_{j\in J} d_j}\left(1 - \sum_{j\in J} x_{ij}\right) > \sum_{i\in I} \frac{q_i}{\sum_{j\in J} d_j}\left(1 - \sum_{j\in J} x'_{ij}\right). \tag{21}$$

The right hand side of the inequality 21 is zero since (t', X') is a feasible solution. The left hand side of the inequality gives the value of S for the solution (t, X). Thus $S > 0$. □

Corollary 5. *If $S < 0$ for an infeasible solution (t, X) in the interval of search $t_{\mathrm{LB}} < t < t_{\mathrm{UB}}$, then there is no feasible solution in the interval (t, t_{UB}).*

Given a feasible solution (t, X) to LLP() with $S = 0$ and $t_{\mathrm{LB}} \le t < t_{\mathrm{UB}}$, a new feasible solution will be sought by solving DSP(t, X), the dual of SP(t), and imposing an additional constraint as described below. Problem (DSP(t, X)):

$$\min u_0 = \sum_{j\in J} \nu_j \sum_{i\in I} q_i x_{ij}, \tag{22}$$

subject to

$$\sum_{j\in J} d_j \nu_j \ge 1, \tag{23}$$

$$\nu_j \ge 0 \quad \text{for all } j \in J. \tag{24}$$

The optimal solution to DSP(t, X) yields $u_0 = t$ and for the tight constraint $\sum_{i\in I} q_i x_{ij^*} = td_j^*$ in SP$_{j^*}(t)$ and $\nu_j = 0$ for all $j \neq j^*$. The current dominant product in the solution is determined by the index j^*, and a better solution (if exists) will be determined by replacing this dominant product with another one. This is accomplished very easily by imposing the constraint $\nu_{j^*} \leq 0$. Let u_0' be the solution of this upper bounded problem, then $u_0 < u_0'$. In case of ties, the process is repeated until this condition is obtained. If solving $\mathrm{SP}(u_0')$ yields a feasible solution (u_0', X), then $t_{\mathrm{LB}} = u_0'$. If this solution is infeasible and $S < 0$, then $t_{\mathrm{UB}} = u_0'$. Furthermore, if $u_0' + S < t_{\mathrm{LB}}$ or $u_0' > t_{\mathrm{UB}}$ then $(t_{\mathrm{LB}}, t_{\mathrm{UB}})$ is an untested interval or interval of uncertainty. The optimal value of the objective function will be searched in the final interval of uncertainty. Otherwise $\mathrm{SP}(u_0' + S)$ will be solved again.

The proposed algorithm finds the optimum solution to the multi-product loading problem. It is based on exploiting the primal and dual structures of relaxed Lagrangian simultaneously. It may be described briefly as follows. First a feasible initial solution is obtained. Step 1 solves the dual problem and generates a dual cut. In Step 2 the subset sum problems as subproblems are solved to obtain a minimum capacity allocation requirement A_j for each product j for a given t. Step 2 searches a feasible solution satisfying constraint set (4). If such a feasible solution exists, then a new lower bound is obtained. Otherwise Step 3 reduces the search interval by reducing the upper bound or readjusts the replenishment time and repeats the process. The process is repeated until no more progress is possible. Then a final interval of uncertainty is obtained and exhaustive search is carried out by dividing the interval into smaller intervals by the corollaries of Lemma 1. The algorithm is described fully below.

Step 0. Sort the compartments in descending order and the demand rates in ascending order. Obtain an initial solution (t_0, X_0) and set $k = 0$ (see the Appendix for obtaining an initial solution). Set $\lambda_i = q_i$ for $i \in I$. The lower bound is $t_{\mathrm{LB}} = t_0$, and the upper bound is $t_{\mathrm{UB}} = z_R$. (If $t_0 = z_R$, then the solution is optimal, terminate.) Go to Step 1.

Step 1. Solve $\mathrm{DSP}(t_k, X_k)$ to obtain (ν_j), then generate the values of (ν_j') and u_0' by imposing $\nu_{j^*} \leq 0$ where j^* corresponds to the active constraint in (P). Set $t_k = u_0'$ and go to Step 2.

Step 2. Solve $\mathrm{SP}(t_k)$ by solving each $\mathrm{SP}_j(t_k)$ for all $j \in J$. If $S > 0$, then impose the condition that the compartments allocated to j^* do not overlap with the compartments allocated to any $j \neq j^*$. Sort the compartment allocations with respect to the demand rates. If a feasible solution (t_k, X_k) is obtained by using Lemma 1, then set $k := k + 1$, and $t_{\mathrm{LB}} = t_k$ and go to Step 1. Otherwise, go to Step 3.

Step 3. If $S = 0$ or is negligible in magnitude and the solution is infeasible, then go to Step 3a. Otherwise, set $t = t_k + S$. If $S < 0$, the new upper bound is $t_{\mathrm{UB}} = t_k$. If $t_{\mathrm{LB}} < t < t_{\mathrm{UB}}$, then set $t_k = t$ and go to Step 2, otherwise go to Step 3a.

Step 3a. Exhaustive search in the final interval of uncertainty:

$t_0 = t_{\mathrm{LB}}$, $k = 0$; repeat.

Solve $\mathrm{SP}_j(t_k)$ to obtain A_j for each $j \in J$ and find j^* and A_{j^*} by calculating $\min\{A_j/d_j\}$. Then find A'_{j^*} by solving $\mathrm{SP}_{j^*}(t_k+\epsilon)$ for some sufficiently small $\epsilon > 0$, and $t_{k+1} = \min\{\min\{A_j/d_j \mid j \neq j^*\}, A'_{j^*}/d_{j^*}\}$.

$k := k + 1$;

until $(t_k > t_{\mathrm{UB}})$.

Repeat (starting from the largest such k).

If $\sum_{j\in J} \leq C$, then determine the set of all possible allocations for all $j \neq j^*$ $E_j = \left\{\sum_{i\in I} q_i x_{ij} \mid A_j \leq \sum_{i\in I} q_i x_{ij} < C - \sum_{j\neq j^*} A_j\right\}$, and $E_{j^*} = \{A_{j^*}\}$. Try all possibilities $E_1 * E_2 * \ldots * E_n$ to obtain a feasible solution. The condition that the compartments should not overlap with the compartments in A_{j^*} reduces the number of elements in the sets. A branch-and-bound procedure can decrease number of evaluations of such possibilities. Set $k := k - 1$;

until ($k = 0$ or a feasible solution is obtained).

If a feasible solution is obtained then it is optimal. If all the subintervals are tested but no feasible solution is obtained, then the solution corresponding to the lower bound t_{LB} is optimal.

This algorithm converges to the optimal in a finite number (however, may be large) of steps. At each iteration the algorithm reduces the interval $(t_{\mathrm{LB}}, t_{\mathrm{UB}})$ by increasing the lower bound or decreasing the upper bound in a positive amount by the Lemma 4 and its corollary. The lower bound is increased by finding a new feasible solution (t, X) where $t_{\mathrm{LB}} < t < t_{\mathrm{UB}}$. The algorithm always keeps the lower bound at a feasible solution. The dual cut produces $t > t_{\mathrm{LB}}$ after eliminating the ties. If (t,X) is not feasible in any subiteration as well, then the upper bound is decreased down to t if $S < 0$. Thus, after a finite number of trials a final interval of uncertainty will be obtained. The exhaustive search procedure divides this interval into smaller non-overlapping subintervals. In each subinterval, only one product is dominant, its depletion time determines the replenishment time. Starting from the subinterval of the largest replenishment time, all possibilities are tested. If a feasible solution is obtained, then it is optimal. Otherwise the subinterval with the next largest replenishment time is tested. The process is repeated until all subintervals are tested. If no feasible solution is obtained, then the solution corresponding to the t_{LB} is optimal. This is rather an efficient search procedure, instead of enumerating all possibilities in the interval $(t_{\mathrm{LB}}, t_{\mathrm{UB}})$.

4. An illustration

As an illustration of the method, a randomly selected problem will be considered. In this example, a sea-tanker of capacity of 6560 tons with 11 compartments will transport and deliver five different products, which can not be mixed, from a source to a destination. The daily demand rates are given as follows (in tons/day): $(d_j) = (66, 71, 72, 76, 81)$ with a total demand rate of 366 tons/day. The compartment capacities are as follows: $(q_i) = (844, 826, 764, 675, 661, 626, 626, 565, 373, 313, 287)$ (in tons). The common replenishment time for delivering these products and the delivery quantities will be sought. The set of 11 compartments will be partitioned into five products, and the number of all possibilities is equal to 29607600 $(5!S_{11}^{(5)})$.

Dynamic programming yields the solution $(1217 = q_1+q_8, 1252 = q_6+q_7, 1275 = q_4+q_{10}+q_{11}, 1301 = q_2+q_8, 1425 = q_3+q_5)$ with $t = 1425/81 = 17.593$. It takes 1776.51 CPU seconds on a 486 based PC to find this solution. There are $2^{11} - 1 = 2047$ state variables and it makes 465135 state evaluations.

The proposed algorithm is used in finding an optimal solution to this problem. An initial solution is obtained by the method described in the Appendix and given below (the capacity of the compartments allocated and the delivery quantity of each product). This is not a structured partitioning as can be observed below, since the partitioning of the set of compartments is not done according to their ordering or rankings. An initial solution is obtained as: $(1199 = q_2+q_9, 1261 = q_5+q_{10}+q_{11}, 1301 = q_4+q_7, 1329 = q_3+q_8, 1470 = q_1+q_6)$ with a replenishment time of $1329/76 = 17.486$ days.

A lower bound on the value of the replenishment time is $t_0 = 17.486$, and an upper bound is obtained by $t_{\mathrm{UB}} = 6560/366 = 17.923$ days. The dominant product is $j^* = 4$. The initial values of $\mu_j = 1/366$ for $j = 1,2,3,4,5$ and $\lambda_i = q_i/366$ for $i = 1,2,\ldots,11$. Solving $\mathrm{DSP}(t_0, X_0)$ and imposing the constraint $\nu_4 \leq 0$ yields $u_0 = 17.761$. PLP(17.761) is solved by first solving each $\mathrm{SP}_j(17.757)$ by listing all compartment combinations the finding the minimum allocation level satisfying $A_j \geq td_j$ for each $j = 1,2,3,4,5$. The delivery quantities at $t = 17.761$ are given by the following vector: $(D_j) = (td_j) = (1171.96, 1260, 75, 1278.50, 1349.53, 1438.32)$. Solving each $\mathrm{SP}_j(17.761)$ produces the following minimum capacity allocation requirements for products; $(A_j) = (1191, 1261, 1279, 1350, 1439)$. The dominant product has the index $j^* = 2$. The replenishment time is 17.761 days with these allocations (if feasible) by completely utilizing the compartments allocated to the product $j = 2$. Since $6538 < 6560$, $S = (6560 - 6538)/366 > 0$ and the condition that the compartments allocated to any product do not overlap with the compartments allocated to the dominant product is imposed. Solving $\mathrm{SP}_j(17.761)$ (except $j = 2$) with this condition again yields the minimum allocation requirements $(R_j) = (1191, 1261, 1301, 1390, 1439)$. Now $6582 > 6560$ and $S = (6560 - 6582)/366 = -0.060$, then $t_{\mathrm{UB}} = 17.761$ and the process will be repeated with $t = 17.761 - 0.060 = 17.701$. The delivery quantities are given by $(D_j) = (1168.27, 1256.77, 1274.47, 1345.27, 1433.78)$. Then the minimum allocation requirements, obtained by solving each $\mathrm{SP}_j(17.701 + \epsilon)$, are given by $(A_j) = (1191, 1261, 1275, 1347, 1439)$. The replenishment time is recalculated as $t = \min(18.045, 17.761, 17.708, 17.724, 17.765) = 17.708$, and the dominant product is $j^* = 3$. $\sum_{j \in J} A_j = 6513 < 6560$ implies imposing the nonoverlapping condition stated above and the minimum requirements then are given by $(A_j) = (1191, 1287, 1275, 1390, 1450)$ with

Table 1

(A_j)	$(t_j = A_j/d_j)$	$\min\{t_j\}$	j^*
(1157, 1251, 1261, 1335, 1424)	(17.530, 17.620, 17.514, 17.566, 17.580)	17.514	3
(1157, 1251, 1275, 1335, 1424)	(17.530, 17.620, 17.708, 17.566, 17.580)	17.530	1
(1165, 1251, 1275, 1335, 1424)	(17.652, 17.620, 17.708, 17.566, 17.580)	17.566	4
(1165, 1251, 1275, 1336, 1424)	(17.652, 17.620, 17.708, 17.579, 17.580)	17.579	4
(1165, 1251, 1275, 1347, 1424)	(17.652, 17.620, 17.708, 17.724, 17.580)	17.580	5
(1165, 1251, 1275, 1347, 1425)	(17.652, 17.620, 17.708, 17.724, 17.593)	17.593	5
(1165, 1251, 1275, 1347, 1426)	(17.652, 17.620, 17.708, 17.724, 17.605)	17.605	5
(1165, 1251, 1275, 1347, 1439)	(17.652, 17.620, 17.708, 17.724, 17.765)	17.620	5

$\sum_{j\in J} A_j = 6593$. $S = (6560 - 6593)/366 = -0.090$, and $t_{\text{UB}} = 17.708$ and $t = 17.708 - 0.090 = 17.618$. Repetition of the process goes as follows. The delivery quantities are given by the vector $(D_j) = (1162.79, 1250.88, 1268.50, 1338.97, 1427.06)$, consequently the minimum allocation requirements are $(Aj) = (1165, 1251, 1275, 1347, 1439)$, obtained by solving each $\text{SP}_j(17.618)$. The replenishment time is recalculated and $t = \min(17.652, 17.620, 17.708, 17.724, 17.765) = 17.620$. Since $\sum_{j\in J} A_j = 6477$, the nonoverlapping condition is imposed and the minimum requirements are now given by $(A_j) = (1252, 1251, 1287, 1390, 1439)$. $\sum_{j\in J} A_j = 6619$ implies that $S = (6560 - 6619)/366 = -0.161$ and $t_{\text{UB}} = 17.620$ but $t = 17.620 - 0.161 = 17.459 < 17.486$. Hence, the interval from 17.486 to 17.620 is left an interval of uncertainty. An exhaustive search will be carried in this interval as described below.

At $t = 17.486$, $(A_j) = (1157, 1251, 1261, 1329, 1424)$ computation of the time t yields $t = \min(17.530, 17.620, 17.514, 17.486, 17.580) = 17.486$ with $j^* = 4$. Thus, the next lowest allocation to 1329 for product $j = 4$ will be obtained by solving $\text{SP}_4(t + \epsilon)$ and it is 1335. The same reasoning is repeated until t becomes equal to or exceeds 17.620 and the results are listed in Table 1.

It is already known that there are no feasible solutions at $t = 17.620$, hence the last interval is ignored. In the interval at $t = 17.605$, the sets of possible allocations will be determined first. There may be several compartment combinations to obtain an allocation level, for instance $1286 = q_6 + q_8$ or $1286 = q_7 + q_8$. The superscript above the allocation level represents the number of different compartment combinations. $E_1 = \{1191^{(2)}\}$, $E_2 = \{1252\}$, $E_3 = \{1287^{(2)}, 1301^{(2)}\}$, $E_4 = \{1390^{(2)}\}$, and finally $E_5 = \{1426\}$ since $j^* = 5$. Noticing that $1191 = q_6 + q_8 = q_7 + q_8$ and $1252 = q_6 + q_7$ implies that there exists no feasible solutions and the interval at $t = 17.593$ is tested next. The sets of allocations are as follows: $E_1 = \{1165, 1191^{(2)}, 1199, 1217, 1225, 1226^{(2)}\}$, $E_2 = \{1251, 1252, 1275, 1286^{(2)}, 1301^{(2)}, 1312^{(2)}\}$, the allocations for $j = 3$ are in the set $E_3 = \{1275, 1286^{(2)}, 1301^{(2)}, 1312^{(2)}, 1335\}$, and the allocations for $j = 4$ are in the set $E_4 = \{1361, 1391, 1409, 1426, 1444\}$ and $E_5 = \{1425\}$. Employing a branch and bound finds a feasible solution in 108 trials and $t^* = 17.593$. Another optimal solution is obtained and is given below. The execution time of this problem is 32.10 CPU seconds.

$$(x_{i1}) = (0,1,0,0,0,0,0,0,1,0,0)$$
$$A_1 = 1199 \quad D_1 = 17.593 * 66 = 1161.14,$$
$$(x_{i2}) = (0,0,0,0,0,1,1,0,0,0,0)$$
$$A_2 = 1252 \quad D_2 = 17.593 * 71 = 1249.10,$$
$$(x_{i3}) = (0,0,0,1,0,0,0,0,0,1,1)$$
$$A_3 = 1275 \quad D_3 = 17.593 * 72 = 1266.70,$$
$$(x_{i4}) = (1,0,0,0,0,0,0,1,0,0,0)$$
$$A_4 = 1409 \quad D_4 = 17.593 * 76 = 1337.07,$$
$$(x_{i5}) = (0,0,1,0,1,0,0,0,0,0,0)$$
$$A_5 = 1425 \quad D_5 = 17.593 * 81 = 1425.$$

This delivery schedule will be repeated every 17.593 days and the product 5 is the dominant product.

If all the possibilities in the interval (17.486, 17.620) is tested without this efficient method, the number of possibilities tested would be 99792 ($= 12 * 3 * 12 * 11 * 21$). This search procedure has tested only 108 possibilities. In the worst case it would test

$2*1*4*2*1+8*9*8*5*1=2880+16=2896$ which is 2.90% of 99792 and it is approximately 0.0098% of all 29607600 possibilities. Computational experience of the algorithm is tested by running several randomly generated problems and results are summarized in Table 2.

Various practical size problems are randomly generated and solved on a 486 based PC. In the generation of the random problems, the demand rate is assumed uniformly distributed in the interval from 20 to 100, and the compartment capacities are also uniform from the interval 150 to 950. The random numbers are then rounded off to the nearest integer. Average execution time and its standard deviation for each group of 5 problems are given in Table 2. Because of the binary character of the problems, there is a high variability in the execution times. A further regression analysis yields that the execution time can be expressed as an exponential function of n (number of products) and m (number of compartments) as follows: $time = 0.0002(1.3198^n)(2.7229^m)$. The method finds the optimal solutions (optimal replenishment time and the optimal assignment of the compartments to the products) to the problems. Ronen (1995) points out that there are 13 compartments or less in the vehicles used in the petroleum distributing firms. Larger size problems will require much more memory capacity and execution speed.

5. Case of equal size compartments and conclusions

If all the compartments are of the same size, then the problem is very much simplified. Let $q = q_i$ for $i \in I$, then the constraints (5) of the problem (P) will be reexpressed as follows:

$$td_j - q\sum_{i\in I} x_{ij} \le 0 \quad \text{for all } j \in J, \tag{25}$$

where the sum $\sum_{i\in I} x_{ij}$ can be interpreted as the number of compartments assigned to the product j, and let $y_j = \sum_{i\in I} x_{ij}$ for $j \in J$. Then (25) is rewritten as follows.

$$td_j - qy_j \le 0 \quad \text{for all } j \in J. \tag{26}$$

Further $y_j \ge td_j/q$ for all $j \in J$. Since y_j is an integer, and is the smallest integer greater or equal to td_j/q for all $j \in J$, then $y_j = \langle td_j/q\rangle$ for all $j \in J$. Recalling that there are only m compartments available, then the problem (P) is transformed into the following problem.

$$\max z = t,$$

subject to

$$\sum_{j\in J}\langle\frac{td_j}{q}\rangle = m,$$

$$t \ge 0. \tag{27}$$

The constraint (27) is equivalent to decomposing m into n non-empty subsets without regard to order. The number of ways of decomposing m into n is equal to $n!F(n-m)$ where $F(n-m)$ is the Fibonacci number of $(m-n)$. If $m = 7$ and $n = 4$, then there are only 72 ways of assigning 7 compartments into 4 products. Listing of all possibilities is not practical in all cases. This problem is identical to voter's college or political districting problem in the literature. A simple algorithm exists to obtain the optimal answer. An initial estimate of $t = C/\sum_{j\in J} d_J$, and $q = C/m$, then $td_j/q = md_j/\sum_{j\in J} d_j$. Each product $j \in J$ is allocated to $[md_j/\sum_{j\in J} d_j]$ compartments. If the sum of the allocated compartments is less than m, then $md_j/\sum_{j\in J} d_j - [md_j/\sum_{j\in J} d_j]$ are ranked in descending order and the index j with the largest value gets the first available compartment (break the ties arbitrarily), and the index j with the second largest value gets the second available compartment, and so on, until all the compartments are assigned.

This article presents an operational problem. Transporting several products, which cannot be mixed, in a single vehicle yields a combinatorial problem as a special case of a set partitioning problem.

A mixed binary programming model is developed to represent this operational problem mathematically. A solution method is developed by simultaneously exploiting primal and dual structures of relaxed Lagrangian. An efficient algorithm is developed and its convergence in a finite number of steps is shown. Efficiency of the algorithm is tested by running randomly chosen problems of various sizes. Optimal solutions to these random problems are obtained by this method.

A mathematically feasible solution to the problem however may not always be operationally meaningful. If the replenishment time is very short, then the ve-

Table 2
Computational performance of the algorithm in CPU seconds

n	m	9	10	11	12	13
4	mean	4.11	13.68	27.55	101.52	
	std	1.42	4.01	6.38	27.44	
5	mean	8.39	17.58	53.94	180.14	
	std	4.34	5.67	12.03	117.59	
6	mean		32.81	67.29	350.68	
	std		16.29	52.91	164.48	
7	mean				215.74	399.22
	std				31.54	194.79

hicle must repeat the trip to the destination(s) quite often and the time to transport to the destination, to return back to the source, loading and unloading times may far exceed the replenishment time. That means a vehicle of small capacity with respect to the total demand rate of the products. Such a situation may be prevented by using a vehicle of reasonable capacity as compared to the total demand rate of the products and the total traveling distance in a trip.

Appendix A. Finding an initial solution

The following algorithm finds an initial solution to the problem (P). Some further notation is required for this algorithm. Let $I(j) = \{i \mid x_{ij} = 1\}$ for a given $j \in J$. $I(j)$ is the set of compartments assigned to product $j \in J$. Initially, $I(j) = \emptyset$ for all $j \in J$. $I(j') \bigcap I(j) = \emptyset$ for $j \neq j'$. Let $r_j = (\sum_{t \in I(j)} q_t + q_i)/d_j$ for all $j \in J$ and for a given $i \in I$. The term r_j represents the proposed depletion time for product $j \in J$, if all the allocated compartments including q_i are filled completely. The product with the minimum depletion time dictates the replenishment time, then for a given number of compartments (from 1 to m) the term r_j is minimized to obtain an approximate replenishment time. This procedure, however, does not maximize replenishment time. Thus it only provides a lower bound on the value of the replenishment time. The algorithm is described as follows.

Step 0. Sort the compartments in descending order and the demand rates in ascending order.

Step 1. $i = 1$;
repeat
$r_{j^*} = \min\{r_j \mid j \in J\}$ (break ties arbitrarily). Set $x_{ij^*} = 1$ and all other $x_{ij} = 0$ for $j \neq j^*$. Update the set $I(j^*) = I(j^*) + \{i\}$.
$i := i + 1$;
until $i > m$. Then go to Step 2.

Step 2. If $I(j) \neq \emptyset$ for all $j \in J$, then terminate, an initial solution is obtained. If there is some unassigned product $j \in J$ such that $I(j) = \emptyset$, then count the number of unassigned products (nup), and let $J_u = \{j \mid I(j) = \emptyset\}$. For each pair of (i,j) if $j \in J_u$ and $i \in \{m+1, \ldots, m+nup\}$ set $x_{ij} = 1$, otherwise $x_{ij} = 0$. Let j^* be such that $x_{ij^*} = 1$ for $m+1 \leq i \leq m+nup$. For each i' from 1 to m, find $j \neq j^*$ such that $x_{i'j} = 1$. Then determine $\max_{i'} \{\min\{q_{i'}/d_{j^*}, \min_j \{r_j - q_{i'}/d_j\}\}\}$ and set $x_{i'j^*} = 1$ and $x_{i'j} = 0$, $x_{ij} = 1$, $x_{ij^*} = 0$. Update the set $I(j^*) = \{i'\}$ and $J_u = J_u - \{j^*\}$. Hence, a feasible solution is obtained.

There are some optional steps to improve the current initial solution by exchanging the compartments between two products at a time. A few more notations are required for that purpose. Let Δ represent the amount of increase in t and j^* be the index of the tight constraint $\sum_{i \in I} q_i x_{ij^*} = t d_{j^*}$ (break ties arbitrarily), $nol = \sum_{i \in I} x_{ij^*}$ be the number of compartments allocated to the product j^*. $I(j^*) = \{i_1, i_2, \ldots, i_{nol}\}$. The optional steps are given below.

Repeat

$\Delta = 0, \quad k = 1,$

repeat
i from i_{k-1} to $i_k - 1$ (if $i_{k-1} < i_k - 1$),
if $x_{ij} = 1$ for $i \neq j^*$, then set $w_{ij^*} = 1$, $w_{ij} = 0$, $w_{i_kj} = 1$, $w_{i_kj^*} = 0$ and $w_{ij} = x_{ij}$ for all others. $t_k = \min_{j \in J} \{\sum_{i \in I} q_i w_{ij}/d_j\}$. If $t_k > t$, then $\Delta > 0$, $W^k =$

i	r_1	r_2	r_3	r_4	r_5	
1	12.79	11.89	11.72	11.11	10.42	then $x_{15} = 1$, $I(5) = \{1\}$
2	12.52	11.63	11.47	10.87	20.62	then $x_{24} = 1$, $I(4) = \{2\}$
3	11.58	10.76	10.61	20.92	19.85	then $x_{33} = 1$, $I(3) = \{3\}$
4	10.23	9.51	19.99	19.75	18.75	then $x_{42} = 1$, $I(2) = \{4\}$
5	10.02	18.82	19.79	19.57	18.58	then $x_{51} = 1$, $I(1) = \{5\}$
6	19.50	18.33	19.30	19.11	18.15	then $x_{65} = 1$, $I(5) = \{1,6\}$
7	19.50	18.33	19.30	19.11	25.88	then $x_{72} = 1$, $I(2) = \{4,7\}$
8	18.58	26.29	18.46	18.30	25.13	then $x_{84} = 1$, $I(4) = \{2,8\}$
9	15.67	23.58	15.79	23.21	22.75	then $x_{91} = 1$, $I(1) = \{5,9\}$
10	20.41	22.74	14.96	22.42	22.01	then $x_{10,3} = 1$, $I(3) = \{3,10\}$
11	20.02	22.37	18.95	22.08	21.69	then $x_{11,3} = 1$, $I(3) = \{3,10,11\}$

W, $j' = arg\{t_k\}$. (Sort the allocations in ascending order if t_k does not decrease.) Set $k := k+1$;
until $(k > nol)$;
if $\Delta > 0$ then $t' = \max\{t_k\}$ and $X := W^k$; $I(j^*) := I(j^*) + \{i\}$, $I(j') := I(j') - \{i\}$,
until $(\Delta = 0)$.

Example 6. An initial solution to the problem described in the Illustration will be obtained. There are eleven compartments with capacities; $(q_i) = (844, 826, 764, 675, 661, 626, 626, 565, 373, 313, 287)$ and five products with the demand rates $(d_j) = (66, 71, 72, 76, 81)$.

Initialization: $I(j) = \emptyset$ for $j = 1,2,3,4,5$ and all $x_{ij} = 0$ for all i, j.

Now $(\sum_{i \in I} q_i x_{ij}) = (1034, 1301, 1364, 1391, 1470)$. Hence an initial solution is obtained, and $t = \min(15.667, 18.324, 118.944, 18.303, 14.148) = 15.667$ with $j^* = 1$.

$$(x_{i1}) = (0,0,0,0,1,0,0,0,1,0,0)$$
$$(x_{i2}) = (0,0,0,1,0,0,1,0,0,0,0)$$
$$(x_{i3}) = (0,0,1,0,0,0,0,0,0,1,1)$$
$$(x_{i4}) = (0,1,0,0,0,0,0,1,0,0,0)$$
$$(x_{i5}) = (1,0,0,0,0,1,0,0,0,0,0)$$

This solution can further be improved by using the optional steps. The first iteration goes as follows. x_{51} can be interchanged with x_{15} and

$$t = \min\left(\frac{1217}{66}, 18.324, 18.944, 18.303, \frac{1287}{81}\right) = 15.889,$$

or x_{51} can be interchanged with x_{24} and

$$t = \min\left(\frac{1199}{66}, 18.324, 18.944, \frac{1226}{76}, 18.148\right) = 16.132,$$

or x_{51} can be interchanged with x_{33} and

$$t = \min\left(\frac{1137}{66}, 18.324, \frac{1261}{72}, 18.303, 18.148\right) = 17.227,$$

or x_{51} can be replaced by x_{42} and

$$t = \min\left(\frac{1048}{66}, \frac{1287}{71}, 18.944, 18.303, 18.148\right) = 15.879.$$

The maximum of these values is $t = 17.227$ and the amount of improvement $\Delta = 17.227 - 15.667 = 1.560$, hence x_{51} is interchanged with x_{33}, and the new solution is given as follows.

$$(x_{i1}) = (0,0,1,0,0,0,0,0,1,0,0)$$
$$\text{with } A_1 = 1137, \text{ and } t_1 = \frac{1137}{66} = 17.227,$$
$$(x_{i2}) = (0,0,0,1,0,0,1,0,0,0,0)$$
$$\text{with } A_2 = 1301, \text{ and } t_2 = \frac{1301}{71} = 18.324,$$
$$(x_{i3}) = (0,0,0,0,1,0,0,0,0,1,1)$$
$$\text{with } A_3 = 1261, \text{ and } t_3 = \frac{1261}{72} = 17.514,$$
$$(x_{i4}) = (0,1,0,0,0,0,0,1,0,0,0)$$
$$\text{with } A_4 = 1391, \text{ and } t_4 = \frac{1391}{76} = 18.303,$$

$$(x_{i5}) = (1,0,0,0,0,1,0,0,0,0,0)$$
$$\text{with } A_5 = 1470, \text{ and } t_5 = \frac{1470}{81} = 18.148.$$

The sorting is allowed since $j^* = 1$, and $1301/72 = 18.069$, $1261/71 = 17.761$ both exceed 17.223. The second iteration is given after sorting.

$$(x_{i1}) = (0,0,1,0,0,0,0,0,1,0,0)$$
$$\text{with } A_1 = 1137,\ t_1 = 1137/66 = 17.227,$$
$$(x_{i2}) = (0,0,0,0,1,0,0,0,0,1,1)$$
$$\text{with } A_2 = 1261,\ t_2 = 1261/71 = 17.761,$$
$$(x_{i3}) = (0,0,0,1,0,0,1,0,0,0,0)$$
$$\text{with } A_3 = 1301,\ t_3 = 1301/72 = 18.069,$$
$$(x_{i4}) = (0,1,0,0,0,0,0,1,0,0,0)$$
$$\text{with } A_4 = 1391,\ t_4 = 1391/76 = 18.303,$$
$$(x_{i5}) = (1,0,0,0,0,1,0,0,0,0,0)$$
$$\text{with } A_5 = 1470,\ t_5 = 1470/81 = 18.148.$$

If x_{31} is interchanged with x_{15}, then

$$t = \min\left(1217/66, 17.761, 18.069, 18.303, 1390/81\right) = 17.160,$$

or if x_{31} is interchanged with x_{24}, then

$$t = \min\left(1199/66, 17.761, 18.069, 1329/76, 18.148\right) = 17.486.$$

The maximum of those is $t = 17.486$, $j^* = 4$ and the amount of improvement is $\Delta = 17.486 - 17.227 = 0.259$ and the new solution is given below.

$$(x_{i1}) = (0,1,0,0,0,0,0,0,1,0,0)$$
$$\text{with } A_1 = 1199,\ t_1 = 1199/66 = 18.167,$$
$$(x_{i2}) = (0,0,0,0,1,0,0,0,0,1,1)$$
$$\text{with } A_2 = 1261,\ t_2 = 1261/71 = 17.761,$$
$$(x_{i3}) = (0,0,0,1,0,0,1,0,0,0,0)$$
$$\text{with } A_3 = 1301,\ t_3 = 1301/72 = 18.069,$$
$$(x_{i4}) = (0,0,1,0,0,0,0,1,0,0,0)$$
$$\text{with } A_4 = 1329,\ t_4 = 1329/76 = 17.486,$$
$$(x_{i5}) = (1,0,0,0,0,1,0,0,0,0,0)$$
$$\text{with } A_5 = 1470,\ t_5 = 1470/81 = 18.148.$$

An investigation shows that no further improvement ($\Delta = 0$ now) is possible. Therefore, this solution is delivered as an initial solution.

References

[1] Abramowitz, M., and Stegun, I.A., eds. (1970), *Handbook of Mathematical Functions* (Dover Publications).

[2] Barcia, P., and Jorsten, K. (1990), "Improved Lagrangian Decomposition: An Application to the Generalized Assignment Problem", *European Journal of Operational Research* 46, 84-92.

[3] Christofides, N., Carpaneto, G., Mingozzi, A., and Toth, P. (1976), "Loading of Liquids into Tanks", Imperial College Report.

[4] Fisher, M.L., and Kedia, P. (1990), "Optimal Solution of the Set Covering/Partitioning Problems Using Dual Heuristics", *Management Science* 36/6, 674-688.

[5] Gavish, B., and Pirkul, H. (1991), "Multi-Resource Generalized Assignment Algorithm", *Management Science* 37/6, 695-713.

[6] Holmberg, K. (1994), "Cross Decomposition Applied to Integer Programming Problems: Duality Gaps and Convexification in Parts", *Operations Research* 42, 657-668.

[7] Marsten, R.E. (1974), "An Algorithm for Large Set Partitioning Problems", *Management Science* 20, 779-787.

[8] Martello, S., and Toth, P. (1987), "Algorithms for Knapsack Problems", *Annals of Discrete Mathematics* 31, 213-258.

[9] Neebe, A., and Danenbring, D. (1977), "Algorithms for a Specialized Segregated Storage Problem", Technical Report No:77-5, University of North Carolina, NC.

[10] Ronen D. (1995), "Dispatching Petroleum Products", *Operations Research* 43, 379-387.

[11] Tang, C.S. (1988), "A Max-Min Allocation Problem: Its Solutions and Applications", *Operations Research* 36/2, 359-367.

[12] Van Roy, T.J. (1983), "Cross Decomposition For Mixed Integer Programming", *Mathematical Programming* 25, 46-63.

[13] Van Roy, T.J. (1986), "A Cross Decomposition Algorithm for Capacitated Facility Location", *Operations Research* 34/1, 145-163.

[14] Yuceer, U., and Dogrusoz, H. (1994), "Designing Physical Distribution Systems With Interacting Operating Policies", submitted to *OR Transactions* for publication and a Technical Report 94-06-F at the Faculty of Administration, Bilkent University, Ankara, Turkey.