A Tabu Search Approach for solving the Travelling Purchase Problem

Ramadan Abdel-Hamed Zean El-Dean
Operations Research Department, ISSR,
Cairo University

Abstract

The traveling purchaser problem (TPP) consists of determining a simple cycle in graph passing through the initial start node and a subset of markets so that all products are purchased at a minimum total cost obtained by adding the routing (travel) cost and the purchasing cost. The TPP is classified as NP-Hard and can be seen as a generalization of the Traveling Salesman Problem (TSP). Tabu Search (TS) has been successfully applied to a variety of combinatorial problems. In this paper, we develop a solution for the TPP which consists of three phases: the first phase is to find the shortest path between every two nodes on the graph using Floyd's algorithm, the second phase is to find an initial solution using the shortest path associated with cost at each market. The last phase is to apply the TS to find the best solution.

Key words: *Travelling Purchaser Problem, Tabu Search, Heuristic, Shortest Path.*

1. Introduction

Traveling Purchaser Problem (TPP) is defined as follows [14]. Let us consider a source node (*depot)* v_0, a set of markets $M = \{v_1,....,v_n\}$, and a set of products $K = \{p_1,....,p_m\}$. Let $G = (V, E)$ be a graph where $V = \{v_0\} \cup M$ is the vertex set and $E = \{[v_i, v_j] : v_i, v_j \in V, i < j\}$ is the arc set. A purchaser originally in v_0 must select and visit a subset of markets to buy a given amount of each product in *K*. Each product p_k can be purchased at a subset M_k of markets. Let d_k be the units of the product p_k that must be purchased, and let q_{ki} be the units of the product p_k that are available at market $v_i \in M_k$. We assume that q_{ki} and d_k satisfy $0 < q_{ki} \le d_k$ *and* $\sum_{v_j \in M_k} q_{kj} \ge d_k$ for all $p_k \in K$ and $v_i \in M_k$. Let us denote the price of a unit of product p_k at market v_i by b_{ki}, and the travel cost between v_i and v_j is equal to the cost of the edge $e = [v_i, v_j]$ denoted by c_e. The TPP consists of determining a simple cycle in *G* passing through the depot and a subset of markets so that all products are purchased at a minimum total cost obtained by adding the routing cost and the purchasing price. The TPP is classified as NP-Hard and can be seen as a generalization of the Traveling Salesman Problem (TSP).

Tabu Search (TS) is a search procedure designed to escape from a local optimum in pursuit of a global optimal solution. This search procedure incorporates flexible memory functions to forbid transitions in solutions (moves) that reinstate certain attributes of past solutions for the purpose [11].

The procedure of TS is simple. At each iteration, the best move is selected. If this move deteriorates the objective function value, it is only performed if the inverse move does not have the tabu status, i.e. if it is not in the tabu list. If it is in the list, then the next best move not in the tabu list is selected and performed. This process is repeated until a stopping criterion is reached. [1]. It has been successfully applied to a variety of combinatorial problems, which include maximum clique problem, traveling salesperson problem, quadratic assignment problem, the bandwidth packing problem, and the bin packing problems [15].

The paper is organizes as follows. The problem definition and formulation is

INFOS2008, March 27-29, 2008 Cairo-Egypt

presented in section 2. A brief overview about tabu search is presented in section 3. The proposed approached is discussed in section 4. Section 5 is assigned to the computational analysis. Finally the conclusions are presented in section 6.

2. Problem formulation

The TPP is defined and formulated as an integer linear programming problem [3, 9, 10] as follows:

For $S \subset V$,define

$E(S) = \{[v_i, v_j] \in E : v_i, v_j \in S, i < j\}$

and

$\delta(S) = \{[v_i, v_j] \in E : v_i \in S, v_j \in V \setminus S\}$

.

Also define

$M^* = \{v_0\} \cup \{v_i \in M : \text{there exist } p_k \in K$ such that $\sum_{v_j \in M_k \setminus \{v_i\}} q_{kj} < d_k\}$,

the set of vertices that must necessarily be part of any feasible TPP solution, and $K^* = \{ p_k \in K : \sum_{v_i \in M_k} q_{ki} = d_k\}$. the set of products without market decision options. There are three types of decision variables:

$$x_e = \begin{cases} 1 & \text{if the edge } e \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } e \in E$$

$$y_i = \begin{cases} 1 & \text{if the vertex } v_i \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } v_i \in V$$

z_{ki} = the amount of product p_k is purchased at market v_i for all $p_k \in K$ and $v_i \in M_k$.

Then the TPP formulation is

$$w^{OPT} = \min \sum_{e \in E} c_e x_e + \sum_{p_k \in K} \sum_{v_i \in M_k} b_{ki} z_{ki} \qquad (1)$$

Subject to

$$\sum_{e \in \delta(\{v_i\})} x_e = 2y_i \quad \text{for all } v_i \in V \qquad (2)$$

$$\sum_{e \in \delta(S)} x_e \geq 2y_i \quad \text{for all } S \subseteq M \text{ and all } v_i \in S \qquad (3)$$

$$\sum_{v_i \in M_k} z_{ki} = d_k \quad \text{for all } p_k \in K \qquad (4)$$

$$z_{ki} \leq q_{ki} y_i \quad \text{for all } p_k \in K \text{ and all } v_i \in M_k \qquad (5)$$

$$x_e \in \{0,1\} \quad \text{for all } e \in E \qquad (6)$$

$$y_i \in \{0,1\} \quad \text{for all } v_i \in M \qquad (7)$$

$$y_0 = 1 \qquad (8)$$

$$z_{ki} \geq 0 \quad \text{for all } p_k \in K \text{ and all } v_i \in M_k \qquad (9)$$

Constraints (2) are degree constraints, and constraints (3) ensure connectivity. Equations (4) guarantee that each product p_k is purchased, and inequalities (5) mean that a product p_k cannot be purchased at an unvisited v_i. Constraints (6) to (9) impose bounds and integrality condition on the variables.

The literature on TPP is mostly directed towards the development of heuristic or near optimal methods. In [3], they present a branch and cut algorithm for the undirected traveling purchaser problem. The algorithm is tested over several classes of randomly generated instances.

In [9], they present a new heuristic approach based on a local-search scheme, exploring a new neighborhood structure. The proposed heuristic is evaluated on a broad class of instances from literature.

In [10], they present the travelling purchaser problem as a biobjective problem and they present an iterative procedure to compute the efficient set in the objective space. For each efficient point in the objective space, a Pareto optimal solution in the decision space is computed. A basic algorithm generates both supported and non-supported efficient points, and a variant generates only supported efficient points. The procedure combines the classical methods in bicriteria programming with a standard branch-and-cut algorithm to solve single-objective subproblems. The algorithm was implemented and tested on three families of test-bed instances from literature proving the good performance of the proposal.

In [12], they proposed metaheuristics based on GRASP (general random Adaptive Search Procedure) and VNS (Variable

Neighborhood Search) and require two algorithms, one of them for generation of an initial solution and the other for local search. In [2], they describe improvement subprocedures that can be incorporated in known heuristics. Computational results show the effectiveness of these subprocedures in general savings, commodity adding and tour reduction heuristics for test instances up to size 200 and four different cost types, including both symmetric and non symmetric costs as well as coordinate based costs.

3. An Overview about Tabu Search

Glover [4, 5], (1986) is the initiator of Tabu Search. Similar views were developed by Hansen (1986) who formulated a steepest ascent/mildest descent heuristic principle.

Tabu Search (TS) is a search procedure designed to escape from a local optimum in pursuit of a global optimal solution. This search procedure incorporates flexible memory functions to forbid transitions in solutions (moves) that reinstate certain attributes of past solutions for the purpose [11].

Tabu Search constitutes a meta-procedure that organizes and directs the operations of a subordinate method. Tabu search has been applied across a wide range of problem settings in which it consistently has found better solutions than methods previously applied to these problems. The literature about TS (mainly reports on applications) is now growing at a very fast rate. TS has been used to 'solve' a large variety of combinatorial optimization problems like scheduling, vehicle routing, quadratic assignment, clustering and grouping, electronic circuit design, graph coloring, and the traveling salesman problem [6, 13].

Tabu search neither makes resource to randomization nor undertakes to proceed slowly to a local optima on the supposition that the proper rate of descent will make the local optimum, when finally reached, closer to a global optimum. Instead of terminating upon reaching a point of local optimality, tabu search structures the operation of its embedded heuristic in a manner that permits it to continue. This is accomplished by forbidding moves with certain attributes (making them tabu), and choosing moves from those remaining that the embedded heuristic assigns to a highest evaluation. In this respect, the method is a constrained search procedure, where each step consists of solving a secondary optimization problem (simple enough to be accomplished by inspection within the framework of most evaluation rules), admitting only those solutions, i.e., moves, that are not excluded by the currently reigning tabu conditions. The tabu conditions have the goal of preventing cycling and inducing the exploration of new regions. The need for a means to avoid cycling arises because, upon moving away from a local optimum, an unconstrained choice of moves (pursuing those with high evaluations) likely will lead back to the same local optimum. The tabu restrictions are designed to prevent this, and more generally to prevent any series of moves dominated by those forbidden at the primary level [7].

Tabu Search Description (Scheme)

It starts with an initial feasible solution, and for each solution (S) a new solution (S') is obtained with a function, called a move, that transforms S into S'. A simple TS makes the move to the best neighboring solution even though it is worse than the given solution, while taking steps to ensure that the search procedure does not re-visit solutions previously visited. This is accomplished by introducing restrictions on possible moves which discourage reversal and/or repetition of selected moves. That is, we define a set of moves that are tabu (forbidden) now, and these moves are sorted in a set called the *tabu list*. Elements in the list define all tabu moves that cannot be applied to the current solution. The size of the list is bounded by a parameter S_L, called the *tabu list size*. If the number of elements in the list is equal to S_L, one must remove an element in it, generally the oldest one, before adding a move to the tabu list. Classical TS algorithms generate all neighborhood solutions for a given solution to select a move. There may be several ways of defining tabu moves. The short term memory is used to avoid moves that will cancel the effect of moves made recently. As a criterion to terminate the algorithm, two rules are commonly used. One is to terminate if the iteration count (the number of moves made) reaches a predetermined number, and the other is to terminate if no improvement has been made of a certain number of iterations[11].

In the scheme, the strategy for escaping from local minima is the following one. Even if there is no better solution than the current one, x_n, in the neighborhood $V(x_n)$, one moves to the best possible solution x in $V(x_n)$ or a sub-neighborhood $V'(x_n) \subseteq V(x_n)$ in the case where $V(x_n)$ is too big to be explored efficiently. If the neighborhood structure is symmetric, i.e. if x_n belongs to the neighborhood $V(x)$ of x whenever $x \in V(x_n)$, there is a danger of cycling when we explore

V(x) during the next step; there is indeed a chance that x_n could be the best solution in V(x) in which case we would come back to x_n and from then on, oscillate between *x* and x_n. To avoid this situation and more general cycling situations, we could store in a list ($x_{n-1},\ldots,x_{n-L}$) called tabu list, a certain number *L* of the last solutions encountered. If *x* is in the list, the move $x_n \rightarrow x$ is forbidden.

The tabu list is usually a list of one or several attributes of the recently visited solutions or (of the converse) of the most recent moves: these attributes should be chosen accurately.

Tabu Search approach (TS) [1, 15]

1. Initialization: Select an initial solution x^1 in *X*; where *X* is the feasible solution region. Initialize the best value F^* of *F* and the corresponding solution x^* by letting:

$F(x^1) \rightarrow F^*$

$x^1 \rightarrow x^*$ Tabu list *TL* is set empty.

2. Iterative search: Let x^n denote the current solution for step *n= 1,2,….* $\bar{F}$ is used to store the best accessible value of *F* found in exploring $V(x^n)$, the subneighborhood of x^n. Let $\bar{x}$ be the solution in $V(x^n)$ for which $F(\bar{x}) = \bar{F}$.

$\bar{F}$ is initially set to ∞

For all $x \in V(x^n)$, If $F < \bar{F}$ and (if the move ($x^n \rightarrow x$) is not tabu or the move is tabu but corresponding *F* passes the aspiration criterion, then let

$F(x) \rightarrow \bar{F}$ and $x \rightarrow \bar{x}$.

Let $\bar{x} \rightarrow x^{n+1}$.

If $\bar{F} < F^*$, then $\bar{x} \rightarrow x^*$ and $\bar{F} \rightarrow F^*$.

A move is established: $x^n \rightarrow x^{n+1}$, with the modification of the associated information.

3. End: if the stopping criterion is reached, then stop.

4. The solution approach:

The proposed approach for solving the problem consists of three parts:

1- Finding the shortest path between every two nodes through the network using Floyd'a algorithm [8]

2- Finding an initial solution

3- Using tabu search to find the best solution

4.1 The initial solution:

The initial solution could be got randomly or according to a priority rule. We apply the least cost (travelling cost + purchasing cost) priority rule to find good initial solution.

1. We start from the source node (v0), finding the least travelling cost between this node and all other nodes Using Floyd's algorithm
2. At other nodes, we check the availability of the required products, and compute the purchasing cost and the travelling cost to that node
3. We select the node with the least total cost
4. If all quantities of all products are purchased, we add to this total cost the traveling cost to the source node, stop
5. Else we repeat the previous steps on the current node

4.2 The Tabu Search algorithm

After finding the initial solution, we apply the tabu search algorithm presented in section 3.

In the Tabu search process to solve our TPP, the entire neighborhoodof a local solution will be searched. It is based on the consideration that the whole neighborhood of a solution consists of manageable number of feasible solutions. We use several tabu list sizes according to the problem size and number of products.

5. Computational analysis

To evaluate the performance of our proposed approach for solving the TPP, we introduce two counter examples; the first one with five markets and a single product, and the other with 9 markets and five products. Our proposed approach is implemented in visual basic 6.

Example 1:

This is a simple problem consists of 5 markets and the source node v_0. A single product is available at the 5 markets with different quantities and prices. Let the travel cost c_e be as follows:

	1	2	3	4	5	6
1	-	2	3	5	-	-
2	2	-	-	4	-	-
3	3	-	-	-	5	-
4	-	4	-	-	-	6
5	-	-	5	-	-	7
6	-	-	-	6	7	-

Table 1. The travel cost for Example 1

Let the available amounts q_{1i} and the prices at each market b_{1i} as follows:

Market v_i	2	3	4	5	6
Available amount q_{1i}	5	6	5	7	4
The unit price b_{1i}	8	12	10	7	5

Table 2. The available amounts and prices for Example 1

And the required units d_1 are 20

The solution:
The table of the shortest path between every two nodes:

	1	2	3	4	5	6
1	-	2	3	5	8	11
2	2	-	5	4	10	10
3	3	5	-	8	5	12
4	5	4	8	-	13	6
5	8	10	5	13	-	7
6	11	10	12	6	7	-

Table 3. The shortest paths for Example 1

The initial solution is:

Markets v_i	6	2	4	5	
Units	4	5	5	6	
Price	5	8	10	7	
Purchasing cost	20	40	50	42	
Travel cost	11	10	4	13	8
Total cost	31	50	54	55	8

The total cost = 198

Table 4: The initial solution for Example 1

After applying the tabu search we could find the following solution, which is optimal:

Markets v_i	2	4	6	5	
Units	5	4	4	7	
Price	8	10	5	7	
Purchasing cost	40	40	20	49	
Travel cost	2	4	6	7	8
Total cost	42	44	26	56	8

The total cost = 176

Table 5: The TS solution for Example 1

Example 2:

This is a simple problem consists of 9 markets and the source node v_0. Five products are available at some of the markets with different quantities and prices. Let the travel cost c_e be as follows:

	1	2	3	4	5	6	7	8	9	10
1	-	2	4	-	3	-	-	-	-	-
2	2	-	-	5	-	-	-	-	-	-
3	4	-	-	-	-	8	-	-	-	-
4	-	5	-	-	-	-	6	-	-	-
5	3	-	-	-	-	7	1	-	-	-
6	-	-	8	-	7	-	-	3	5	-
7	-	-	-	6	1	-	-	3	-	9
8	-	-	-	-	-	3	3	-	7	-
9	-	-	-	-	-	5	-	7	-	2
10	-	-	-	-	-	-	9	-	2	-

Table 6: The travel cost for Example 2

Let the available amounts q_{ki} and the prices at each market b_{ki} as follows:

Market v_i	2	3	4	5	6	7	8	9	10
Available amount q_{1i}	3	5	4	6	5	5	4	6	5
The unit price b_{1i}	5	4	4	4	5	6	3	3	3
Available amount q_{2i}	7	2	6	7	4	3	6	7	5
The unit price b_{2i}	4	5	3	7	6	4	7	3	5
Available amount q_{3i}	3	4	5	0	5	3	8	10	20
The unit price b_{3i}	8	3	3	4	5	3	6	2	1
Available amount q_{4i}	2	2	0	6	2	2	5	4	3
The unit price b_{4i}	5	4	7	6	2	5	1	3	5
Available amount q_{5i}	1	6	3	7	0	6	7	8	2
The unit price b_{5i}	7	2	8	7	1	8	1	10	4

Table 7: The available amounts and prices for Example 2

And the required units d_k are:

Product (k)	1	2	3	4	5
Units (d_k)	30	30	30	20	20

Table 8: The required amounts of each product for Example 2

The solution:
The table of the shortest path between every two nodes:

	1	2	3	4	5	6	7	8	9	10
1	-	2	4	7	3	10	4	7	14	13
2	2	-	6	5	5	12	6	9	16	15
3	4	6	-	11	7	8	8	11	13	15
4	7	5	11	-	7	12	6	9	16	15
5	3	5	7	7	-	7	1	4	11	10
6	10	12	8	12	7	-	6	3	5	7
7	4	6	8	6	1	6	-	3	10	9
8	7	9	11	9	4	3	3	-	7	9
9	14	16	13	16	11	5	10	7	-	2
10	13	15	15	15	10	7	9	9	2	-

Table 9: The shortest paths for Example 2

The initial solution:

Markets v_i	2	3	4	5	7	6	8	9	10	
Units (q_{1i})	0	5	4	6	0	0	4	6	5	
Price (b_{1i})	5	4	4	4	6	5	3	3	3	
Units (q_{2i})	7	2	6	0	3	0	0	7	5	
Price (b_{2i})	4	5	3	7	4	6	7	3	5	
Units (q_{3i})	0	0	0	0	0	0	0	10	20	
Price (b_{3i})	8	3	3	4	3	5	6	2	1	
Units (q_{4i})	2	2	0	0	2	2	5	4	3	
Price (b_{4i})	5	4	7	6	5	2	1	3	5	
Units (q_{5i})	1	6	0	4	0	0	7	0	2	
1 Price (b_{5i})	7	2	8	7	8	1	1	10	4	
Travel cost	2	6	11	7	1	6	3	7	2	13
Total cost	47	56	45	59	23	10	27	78	85	

The total cost = 430 + 13 =443

Table 10: The initial solution for Example 2

The best tabu search solution:

Markets v_i	2	4	7	5	8	6	9	10	3	
Units (q_{1i})	0	4	0	6	4	0	6	5	5	
Price (b_{1i})	5	4	6	4	3	5	3	3	4	
Units (q_{2i})	7	6	3	0	0	0	7	5	2	
Price (b_{2i})	4	3	4	7	7	6	3	5	5	
Units (q_{3i})	0	0	0	0	0	0	10	20	0	
Price (b_{3i})	8	3	3	4	6	5	2	1	3	
Units (q_{4i})	2	0	2	0	5	2	4	3	2	
Price (b_{4i})	5	7	5	6	1	2	3	5	4	
Units (q_{5i})	1	0	0	4	7	0	0	2	6	
1 Price (b_{5i})	7	8	8	7	1	1	10	4	2	
Travel cost	2	5	6	1	4	3	5	2	15	4
Total cost	47	39	28	53	28	7	76	85	65	

The total cost = 428 + 4 =432

Table 11: The Tabu Search solution for Example 2

6. Conclusions:

We have presented the travelling purchaser problem (TPP), which is classified as NP-Hard and can be seen as a generalization of the Traveling Salesman Problem (TSP). A Tabu Search based approach is developed, and coded in Visual basic 6. The proposed approach is implemented and tested on several problems. Two counter examples are introduced and solved using the proposed approach.

References

[1] Alex Van Breedam, *Comparing descent heuristics and metaheuristics for the vehicle routing problem*, Computers & Operations Research 28, 2001.

[2] A. Teeninga, and A. Volgenant, *Improved heuristics for the traveling purchaser problem*, Computers & Operations Research 31,139–150, 2004.

[3] Gilbert Laporte, Jorge Riera-Ledesma, and Juan-José Salazar-Gonzalez, *A Branch-and-Cut Algorithm for the Undirected Traveling Purchaser Problem*, Operations Research, 51 (6) 940-951, 2003.

[4] Glover F. *Tabu search } Part I*. ORSA Journal of Computing 1, 190, 1989.

[5] Glover F. Tabu search *} Part II*. ORSA Journal of Computing 2, 4, 1990.

[6] Glover, F., *"Tabu Search – Wellsprings and Challenges"*, European Journal of Operational Research 106, 221-225, 1998

[7] Glover, F. and Greenberg, H. J., *"New approaches for heuristic search: A bilateral linkage with artificial intelligence"*, European Journal of Operational Research 39, 119-130, 1989

[8] Hamdy A. Taha; *Operations Research An Introduction*, Seventh Edition, Prentice Hall; 2003.

[9] Jorge Riera-Ledesma, and Juan-José Salazar-Gonzalez, *A heuristic approach for the Travelling Purchaser Problem*, European Journal of Operational Research 162, 142–152, 2005

[10] Jorge Riera-Ledesma, and Juan-José Salazar-Gonzalez, *The biobjective travelling purchaser problem*, European Journal of Operational Research 160, 599–613, 2005.

[11] Lee, J. k. and Kim, Y. D., *"Search heuristics for resource constrained project scheduling"*, Journal of Operational Research Society 47, 5, 678-689, 1996.

[12] Luiz Satoru Ochi, Mozar B. Silva, and Lucia Drummond, *Metaheuristics Based on GRASP and VNS for Solving the Traveling Purchaser Problem*, MIC'2001 - 4th Metaheuristics International Conference

[13] Pirlot, M., *"General local search methods"*, European Journal of Operational Research 92, 493-511, 1996.

[14] Ramesh T. *Traveling purchaser problem.* Operations Research,18:78–91, 1981.

[15] Wun-Hwa Chen, Chin-Shien Lin, *A hybrid heuristic to solve a task allocation problem*, Computers & Operations Research, 27, 2000.