A Dynamic Programming Approach for a Travelling Purchaser Problem With Additional Constraints

Luis Gouveia*, Ana Paias*, Stefan Voß◇

* *CIO and DEIO, Faculdade de Ciências da Universidade de Lisboa*
BLOCO C6, Piso 4, 1749-016 Lisboa, Portugal

◇ *Institute of Information Systems,University of Hamburg*
Von-Melle-Park 5, 20146 Hamburg, Germany

Abstract

The traveling purchaser problem (TPP) is the problem of determining a tour of a purchaser that needs to buy several items in different shops such that the total amount of travel and purchase costs is minimized. We study a version of the problem with additional constraints, namely, a limit on the maximum number of markets to be visited, a limit on the number of items bought per market and where only one copy per item needs to be bought. We present a method based on a dynamic programming formulation for the problem incorporating market and item information in the state definition. Due to the expected exponentially sized state space of the proposed dynamic program, we develop a state space relaxation (SSR) that provides a lower bound on the cost of the optimal solution. The limited number of markets and purchases per market permit us to incorporate, in a practical way, this relaxation within a subgradient optimization procedure for improving the original SSR lower bound. We also propose a heuristic that is used together with the subgradient optimization procedure, in order to attempt to transform relaxed solutions into feasible solutions for the problem. We present computational results taken from instances with up to 300 markets. The results show that the dynamic programming based method appears to be a good method for giving good intervals on the optimal value for this restricted version of the TPP.

Keywords: *Traveling purchaser problem, dynamic programming,state space relaxation,*

1 Introduction

We are given a set $V = \{0, 1, 2, ..., m-1\}$ of m markets or nodes, a depot $s \in V$ (we assume s = 0), and a set $I = \{1, 2..., n\}$ of n items. For all feasible connections between two markets $i, j \in V$, let c_{ij} denote the cost of traveling from i to j. If item k is available at market i, d_{ik} denotes the cost of purchasing item k at market i. The (traveling purchaser problem) TPP is to find a tour starting from and returning to the depot s while visiting a subset of the m markets to purchase each one of the n items at one of these markets, such that the total of travel and purchase costs is minimized. We consider a variation of the TPP with asymmetric costs, a limit on the maximum number of markets to be visited and a limit on the number of items bought per market. Furthermore, only one copy per item needs to be bought and the number of items is small compared to the number of potential markets.

The TPP can be seen as a mathematical model for several scheduling applications where items correspond to the jobs that have to be processed by a m-state machine and each market corresponds to a different state of the machine. The variation studied here is also motivated by a special scheduling application corresponding to a production site with an oven where jobs have to be treated at different

temperatures (the states). This case has asymmetric set-up times, i.e., c_{ij} need not be the same as c_{ji} (since heating up the oven is more time consuming and more costly than cooling it down) having a maximum allowed number of states until completion (since the machine needs to be cooled down and cleaned after a certain number of jobs have been performed) and having a very limited number of processed jobs per state (due to capacity limitations of the oven).

The TPP is known to be NP-hard because it resumes the well-known traveling salesman problem (TSP) as a special case. The version studied here is also NP-hard since it contains the unit-demand capacitated location problem as a special case (which is obtained after ignoring the travel costs).

Exact methods for the TPP are described in [4] and [6]. In the first work, a branch-and-cut algorithm for the symmetric TPP has been developed and tested for instances with up to 200 markets and 200 items. The results show that the proposed method outperforms previously best methods. In [6] the branch and cut concept is applied to the asymmetric TPP. The reported results indicate that the authors solve instances with up to 200 markets and 200 items and that their method clearly outperforms previously best known methods.

The variation of the TPP that we study here incorporates several constraints, motivated by the previously described scheduling application. As far as we know, none of the previous studies on the TPP considered these additional constraints. The approach that we propose in this paper is specifically tailored for this variant. Thus, we propose a method based on a dynamic programming formulation for the problem that incorporates market and item information in the state definition (see Section 2). Due to the expected exponentially sized state space of the proposed *dynamic program* (DP), we propose a *state space relaxation* (SSR) that provides a lower bound on the cost of the optimal solution. The limited number of markets and purchases per market permits us to incorporate, in a practical way, this relaxation within a subgradient optimization procedure for improving the original SSR lower bound. We also propose a heuristic that is used together with the subgradient optimization procedure, in order to attempt to transform relaxed solutions into feasible solutions for the problem. We present computational results taken from instances with up to 300 markets.

2 The Dynamic Programming Approach

Let $F_k(S, j, B)$ indicate the cost of the least cost solution where the purchaser has already bought all items in the set B and has followed a path with k arcs including all nodes in S and ending at node j and let $d_j(C)$ indicate the cost of purchasing the items in set C at market j. For each market j, we also denote by M(j) the set of items that can be purchased at j. Then,

$$F_k(S, j, B) = \min_{\substack{i : i \in S \text{ and } (i,j) \in A, \\ C \text{ satisfies several conditions}}} \{F_{k-1}(S - \{j\}, i, B - C) + c_{ij} + d_j(C)\}$$

$$S \subseteq V - \{0\}, j \in S, |S| = k, k = 1, ..., H, B \subseteq I \quad (1)$$

To prevent the examination of unfeasible states, the set C in the above recursion must satisfy the following conditions:

1. $C \subseteq (B \cap M(j))$ and $|C| \leq D$
   This condition guarantees that in location j we can only buy the items that are existing in that location and that no more than D objects are bought in any given market.

2. There exists a subset of at most $k-1$ different nodes, $i_1, ..., i_{k-1}, i_p \neq j$ $(p = 1, ..., k-1)$, in $V - \{0\}$ such that $S \subseteq (\cup_{p=1,...,k-1} M(i_p)) \cup M(j)$. This condition states that the items in the set S can be purchased after visiting at most $k-1$ different nodes as well as node j.

3. There exists a subset of at most $m-k$ different nodes, $i_1, ..., i_{k-1}, i_p \neq j$ $(p = 1, ..., m-k)$, in $V - \{s\}$ such that $I \subseteq (\cup_{j=1,...,k-1} M(i_j)) \cup S$. This condition states that the set S has to include enough items to guarantee that the whole set I can be completed by visiting the potential remaining markets.

The recursion is initialized as follows:

$$F_1(S, j, B) = c_{j0} + d(B) \qquad j \in V - \{0\} : (0, j) \in A, S \subseteq M(j), |B| \leq D \tag{2}$$

The optimal solution for the problem is obtained by connecting the last node in a path ending in a state of the form (S, j, I) (meaning that the purchaser is at location j and has already bought all the items) to node 0

$$\min_{S \subseteq V-\{0\}, j \in S:(j,0) \in A,\ |S|=k, k=1,\ldots,H} \{F_1(j, S) = c_{sj} + d(S)\} \tag{3}$$

This DP has an extremely large state space (the size of the state space is $O(|V|^2 2^{|I|})$) making the DP hard to use for instances with as many as 15 markets, even having a reasonably small number of nodes (20, for instance). Note also that if we ignore the item parameter in the state definition, if we set $H = n - 1$ and change (3) to only allow circuits with n arcs (i.e., set $k = n - 1$), we obtain a well known (exponential sized) formulation for the TSP (see, e.g., [5, 3]).

State space relaxation (SSR) is a technique (see, e.g.,[1]) that overcomes the large state space of the original DP. The SSR uses a "relaxed" DP that is defined in a smaller state space permitting us to obtain a lower bound on the value of the DP originally developed. To define a state space relaxation for a given DP we need to define an appropriate mapping function (MF) that maps several states of the original state space into a single state on the relaxed state space. In this way, we can obtain a DP recursion with a (hopefully much) smaller state space. In the relaxed DP, a transition from a state $s1$ to a state $s2$ corresponds to all the transitions in the original DP from all states $S1$ with $MF(S1) = s1$ to all states $S2$ with $MF(S2) = s2$. The cost of the transition from $s1$ to $s2$ in the relaxation will be equal to the minimum of all the costs of the original transitions. Thus, in the relaxed DP, only a lower bound on the value of the optimal solution of the original problem is guaranteed. One way of defining a SSR for this problem is to use a mapping function that maps an original state (S, j, B) into a state $(|S|, j, |B|)$ (more general mapping functions, could be defined as well - see [1]). Here, the relaxation is performed into the two parameter sets S and B and thus, the markets (items) will not be distinguished from each other since we are only keeping track of the number of markets visited (items purchased).

Using the suggested mapping function, we denote by $f_k(j, b)$ (note that we can eliminate the first parameter, s, from the relaxed state definition since it will always be equal to the stage parameter k) the cost of the least cost solution where the purchaser has already bought b items and followed a path with k arcs ending at node j. This allows us to obtain the following relaxed recursion

$$f_k(j, b) = \min_{i:(i,j) \in A,\ c \text{ satisfies several conditions}} \{f_{k-1}(i, b - c) + c_{ij} + d_j(c)\}$$
$$j \in V - \{0\}, k = 1, ..., H, b = 1, ..., |I| \tag{4}$$

where $d_j(c)$ indicates the minimum cost of purchasing c different objects available at node j. Again and similarly to the original recursion, in every stage k, we should impose some conditions on the value c to prevent the examination of unfeasible states.

1. $c \leq min(b, |M(j)|)$ and $c \leq D$
2. There exists a subset of at most $k-1$ different nodes, $i_1, ..., i_{k-1}, i_p \neq j$ $(p = 1, ..., k-1)$, in $V - \{0\}$ such that $c \leq |\cup_{p=1,\ldots,k-1} M(i_p)) \cup M(j)|$

   This condition states that the c items can be purchased after visiting at most $k-1$ different nodes as well as node j.
3. There exists a subset of at most $m - k$ different nodes, $i_1, ..., i_{m-k}, i_p \neq j$ $(p = 1, ..., m - k)$, in $V - \{0\}$ such that $|I| \leq b + \sum_{p=i_1,\ldots,i_{k-1}} |M(p)|$.
   This condition states that b is large enough to guarantee that $|I|$ items can be purchased by visiting the potential remaining markets.

The recursion is initialized as follows:

$$f_1(j,b) = c_{0j} + d_j(b) \qquad j \in V - \{0\} : (0,j) \in A, b = 1, ..., |M(j)| \tag{5}$$

As with the DP in the original space, the optimal solution for the relaxed problem is obtained by connecting the last node in a path ending in a state of the form $(j, |I|)$ (meaning that the purchaser is at location j and has already bought $|I|$ items) to node 0

$$\min_{j \in V-\{0\}:(j,0)\in A, k=1,...,H} \{f_k(j,|I|) + c_{j0}\} \tag{6}$$

The relaxed DP has a much more reduced state space (the size of the state space is $O(|V|^2|I|)$) making this DP much easier to use than the original one.

As before, we note that if we ignore the item parameter b in the state definition, if we set $H = n - 1$ and change (6) to only allow circuits with n arcs (i.e., set $k = n - 1$), we obtain the well known n-path relaxation for the TSP (see, e.g., [5, 3]).

As pointed out before, the relaxed solution may not be feasible for the original problem. It is possible to combine the SSR with a lagrangean relaxation scheme in order to penalize unfeasible solutions. This can be done by describing a generic ILP (integer linear programming) formulation for the problem which permits us to show which constraints should be dropped in order to obtain a relaxed problem that is equivalent to the state space relaxation. A subgradient optimization method is then used to obtain an approximation of the optimal multipliers. A simple lagrangean heuristic that transforms unfeasible solutions into feasible solutions was also developed. This heuristic might not always provide a feasible solution. However, as it is tried in every iteration of the subgradient method, almost every test has terminated with a (reasonably good) solution. Details of this procedure are given in [2]. Here we present results from three approaches for solving the TPP based on the dynamic programming method combined with the subgradient optimization procedure. The first method, named *Simple*, uses the original recursion equation (4), while in the second method, named *Double*, we forbid loops with only two markets by using a double recursion as suggested in [1]. Finally, we have also considered another method, named $R-Double$ which is a restricted version of *Double*. In $R-Double$ we do not penalize the constraints stating that the circuits do not repeat nodes. This method is motivated by the following observations: i) most of the loop infeasibilities of the relaxed solutions of the method *Simple* correspond to 2-loops (which are prevented by the double recursion and ii) removing a set of multipliers from the subgradient helps convergence.

Our programs were coded in C. The computations were done on a PC Pentium IV 3.2 GHz and the CPU times were measured in seconds. For our computational experiments we have used the Class6 instances described in [6] and using the generator described by [7]. We have included the additional parameters for the version of the problem under study. Below, we include the results for instances with up to 300 markets and we have considered 150 iterations for the subgradient method. For each set of parameters, the reported results are average results based on five instances tested. Table 1 gives results for 100 market instances and shows the number of items, the values of H and D, average gaps of the methods *Simple*, *Double* and $R - Double$. In the last column, an (a) means that in one out of the five instances the heuristic was not able to find a feasible solution. In some situations the instances become infeasible for the chosen set of parameters H and D. We indicate this in the last column by giving the number of infeasible instances. Tables 2 and 3 have similar information for instances with 200 and 300 markets respectively. Let $v(\ .\ )$ indicate the objective function obtained with a respective method. The gaps are given as 100*(v(optimal) - v(lower bound))/v(optimal) for the instances with 100 markets and given as 100*(v(upper bound) - v(lower bound))/v(lower bound) for the 200 and 300 market instances. The optimal solutions for the 100 instances were obtained by using the time-dependent ILP model described in [2].

The results for the reported DP methods show that the *Simple* DP approach is the method producing smallest CPU times. However, it is the restricted DP method, $R - Double$, that shows a better performance when we consider gaps as well as CPU times together. In particular, when comparing this method with *Double* one concludes that the extra number of penalized constraints do not appear to help the convergence of the method. To conclude, the reported gaps (note that the largest average reported

Table 1: Gaps and CPU for Class6 with 100 markets

			Gaps			CPU			
Items	H	D	S	RD	D	cpu_ S	cpu_ RD	cpu_ D	
	10	2	0.26	0.26	0.26	5.38	6.40	6.43	
	8	3	0.47	0.34	0.34	7.11	9.81	9.88	
10	8	2	0.26	0.26	0.26	4.20	4.98	5.13	
	5	4	0.14	0.14	0.14	4.68	6.28	6.88	
	5	3	0.47	0.34	0.20	4.10	5.71	5.76	
	10	8	0.60	0.58	0.63		106.31	118.09	118.36
	8	10	0.61	0.52	0.52	93.14	103.53	103.71	
50	8	8	0.60	0.58	0.63	81.36	92.08	92.22	
	5	15	0.63	0.57	0.59	68.57	73.90	74.01	
	5	12	0.59	0.54	0.56	60.02	65.71	65.81	
	10	16	0.94	0.89	0.92	378.95	406.13	406.15	
	8	20	1.46	1.34	1.31	362.31	371.25	371.93	(a)
100	8	16	1.49	1.32	1.42	296.16	316.35	316.97	(a)
	5	30	-	-	-	-	-	-	(5)
	5	24	-	-	-	-	-	-	(5)

Table 2: Gaps and CPU for Class6 with 200 markets

			Gaps			CPU			
Items	H	D	S	RD	D	cpu_ S	cpu_ RD	cpu_ D	
	10	2	1.80	1.79	1.72	35.93	51.48	51.56	
	8	3	1.85	1.85	1.85	31.39	44.61	44.57	
10	8	2	1.73	1.73	1.66	28.04	40.22	40.31	
	5	4	2.44	2.37	2.37	21.30	31.44	31.50	
	5	3	1.85	1.85	1.85	18.15	25.90	25.90	
	10	8	3.09	3.07	3.07	461.49	524.65	524.96	
	8	10	2.73	2.73	2.73	418.63	464.96	465.44	
50	8	8	2.95	2.93	2.93	358.65	409.04	409.26	
	5	15	3.20	3.16	3.16	314.65	340.77	341.07	(1)
	5	12	2.79	3.19	3.17	269.67	297.94	298.27	(1) (a)
	10	16	2.46	2.46	2.53	1599.63	1641.68	1366.82	
	8	20	3.48	3.43	3.44	1389.48	1480.72	1481.19	
100	8	16	2.65	2,.62	2.64	1246.67	1279.61	1280.57	(a)
	5	30	2.18	2.13	2.13	951.27	987.81	988.25	(4)
	5	24	2.27	2.17	2.17	805.28	871.27	871.69	(4)

Table 3: Gaps and CPU for Class6 with 300 markets

			Gaps			CPU			
Items	H	D	S	RD	D	cpu_ S	cpu_ RD	cpu_ D	
	10	2	2.26	2.19	2.19	64.74	97.32	97.57	
	8	3	3.28	3.28	3.28	71.91	101.52	101.53	
10	8	2	2.26	2.19	2.19	50.54	76.08	76.27	
	5	4	2.49	2.49	2.49	37.21	58.24	58.37	
	5	3	3.28	3.28	3.28	41.07	58.87	58.89	
	10	8	2.57	2.54	2.54	1042.98	1188.17	1189.92	
	8	10	2.29	2.26	2.28	949.74	1052.29	1053.52	
50	8	8	2.58	2.54	2.54	811.05	925.57	926.97	
	5	15	2.73	2.70	2.70	711.29	770.32	771.43	
	5	12	1.99	1.98	1.98	604.12	674.15	674.87	(a)
	10	16	2.45	2.44	2.47	3486.42	3741.94	3705.80	
	8	20	2.42	2.42	2.42	3160.51	3322.55	3333.90	
100	8	16	2.94	2.91	2.91	2678.55	2879.60	2887.07	(1)
	5	30	4.60	4.54	4.55	2279.94	2475.44	2483.40	(2)
	5	24	3.41	3.38	3.38	2060.04	2149.56	2155.39	(2)

gap is equal to 4.54) show that the DP approach appears to be a good option for giving good intervals on the optimal value for this restricted version of the TPP.

Future research can elaborate on extended or modified versions of our problem. One of the settings related to our problem that we have not considered in this paper refers to using time and cost values together. Moreover, we might investigate the occurrence of fixed costs for including a market into a solution (related to setup times and costs when entering a market) as well as the consideration of more elaborate processing time and cost functions accounting for economies of scale as well as rebates for specific markets. Note that these variants are easily modeled by the DP and SSR approach.

References

[1] N. Christofides, A. Mingozzi and P. Toth, *State space relaxation procedures for the computation of bounds to routing problems*, Networks, **11** (1981) pp. 145–164.

[2] L. Gouveia, A. Paias and S. Voß, *Models for a Travelling Purchaser Problem With Additional Side-Constraints*, Technical Report 10/2008, Centro de Investigação Operacional, Universidade de Lisboa.

[3] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*, Kluwer, Boston, 2002.

[4] G. Laporte, J. Riera-Ledesma and J.-J. Salazar-Gonzalez, *A Branch-and-Cut Algorithm for the Undirected Travelling Purchaser Problem*, Operations Research, **51** (2003) pp. 940–951.

[5] E.L. Lawler, J.K. Lenstra, A.H.G.R. Kan and D.B. Shmoys, *The Traveling Salesman Problem*, Wiley, Chichester, 1985.

[6] J. Riera-Ledesma and J.-J. Salazar-Gonzalez, *Solving the asymmetric traveling purchaser problem*, Annals of Operations Research, **144** (2006) pp. 83–97.

[7] K.N. Singh and D.L. van Oudheusden, *A branch and bound algorithm for the traveling purchaser problem*, European Journal of Operational Research, **97** (1997) pp. 571–579.