AN ALGORITHMIC FRAMEWORK FOR IMPROVING HEURISTIC SOLUTIONS APPLIED TO NEW VERSIONS OF THE TRAVELING SALESMAN PROBLEM

Jaein Choi [*] **Matthew J. Realff** [*,1] **Jay H. Lee** [*]

[*] *Center for Product and Process Systems Engineering, School of Chemical Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0100*

Abstract: Many practical scheduling and planning problems involve optional tasks or conditional tasks such as cleaning process units or taking additional measurements. Proper consideration of these tasks increases the complexity of scheduling problems. The task network structure becomes more complicated and the decisions often require information that is available only as the schedule is executed. To exemplify the role of such optional tasks in scheduling problems, new deterministic and stochastic traveling salesman problems(TSPs) variants are introduced. Based on the premise that the use of heuristics would be the only recourse in handling such problems, we propose a novel way to improve upon a solution obtained from heuristics by applying dynamic programming to the subset of states visited by the heuristics. The method represents a way to take a family of solutions and patch them up as an improved solution. However, the 'patching up' is accomplished in state space rather than in solution space. We develop and apply the approach to the TSP variants to examine the degree of improvement that can be obtained by the method. For small problems, we compare the quality of these solutions with the global optimal ones. The proposed method can be generalized to other planning and scheduling problems.

Keywords: Scheduling, Planning, Combinatorial Optimization, Conditional Tasks, Heuristics, Traveling Salesman Problem

1. INTRODUCTION

Many scheduling and planning applications involve *optional* or *conditional* tasks. Here optional tasks refer to those that the scheduler has the option of determining whether and when to perform. Conditional tasks represent those tasks that must be performed if certain conditional requirements are met. For example, cleaning process units to remediate fouling may be related to an observation of the state of the unit or a measurement of batch quality. Additional measurements of current process conditions or batch properties could be made to enable better downstream processing and batch-to-batch control. These measurements may trigger new processing tasks to be performed on batches that do not meet specifications.

Scheduling and planning problems with these types of tasks have additional complexities and some novel features with respect to traditional problem definitions and formulations. First, the task network structure is no longer fixed; it can be changed by additional tasks. Second, the batch task parameters may assume values that depend on the performance of the tasks; for example cleaning a reactor may increase the product yield. Third, the decision of whether to perform these types of tasks is often based on information about the process state directly gathered from the process at the time of decision. Fourth, often only partial information to support the decision is available, thus what information is known must be ex-

[1] Email: matthew.realff@che.gatech.edu

plicitly represented. The evolution of the *information state* is usually coupled with the performance of certain optional tasks. For example, optional tasks such as measurements, may not change the state of the process itself, but improve the future information available for decision-making.

Scheduling or planning problems involving these types of tasks require the solution of decision problems that have significant combinatorial complexity, layering the decisions about when and if to perform the optional tasks on top of other decisions. Furthermore, the information tasks make the problem multi-stage in nature, as the new information state can be used to revise the existing scheduling or planning decisions.

To begin to develop an understanding of this class of task planning and scheduling problems, we introduce a new deterministic Traveling Salesman Problem variant as a representative of a deterministic scheduling problem with optional tasks. For the deterministic TSP with an optional task, conventional optimization problem formulations such as a dynamic program (DP) and mixed integer linear program (MILP) can be used to find global optimum solution. These methods are limited to small problems due to the rapid growth the solution space that has to be searched with the size of the problem. We propose a new solution method based on combining suboptimal solutions from heuristics for the problem through an exhaustive dynamic programming search of the subset of states "visited" by the heuristics. This enables us to find a solution that outperforms the heuristic solutions without significant additional computational cost.

2. DETERMINISTIC VERSION OF TRAVELING SALESMAN PROBLEM WITH AN OPTIONAL TASK

The prototypical traveling salesman problem requires the construction of the lowest cost tour of a set of cities, where each city must be visited exactly once on a tour(Lawler and Eugene, 1985). The cost is representative of the travel between the cities, and can be the same or different in both directions. The problem is deceptively simple, but belongs to the class of NP-hard optimization problems, (Garey and Johnson, 1979), for which no algorithm with computation time that scales as a polynomial in the size of the problem is known. Several scheduling problems of interest to chemical engineers have been formulated as TSP's or close variants. For example, the No-Wait Flowshop problem can be transformed to TSP (Penky and Miller, 1991), resource constrained sequencing problem can be reduced to a resource constrained TSP (Pekny *et al.*, 1993) and the parallel flowshop problem can be transformed to a constrained TSP (Gooding *et al.*, 1994).

2.1 Problem Description

Our variant of TSP maintains the same basic structure of the problem, each city being visited exactly once per tour, but modifies the cost of travel. It is assumed that when a salesman reaches a city, a coupon may be purchased that will lower the "distance" or, more abstractly, the cost of traveling between the cities he has not yet visited. The discount is not applied uniformly to the costs of travel and hence the salesman may bias his tour to reach certain cities early to take advantage of discounts on other potential legs of the journey. The coupon is to be purchased exactly once or not to be purchased at all during the tour, and its cost decreases with the number of cities that remain to be visited. The decisions that the salesman has to make are both the order in which he visits the cities and the location at which he buys the coupon.

The introduction of the coupon adds a new dimension to the classic TSP, marginally increasing the size of the solution space ($(N-1)!$ to $N!$), but more importantly, it disrupts the original problem structure. For example, to represent the problem as an integer program requires not just capturing the binary decisions of the connectivity between cities, but also the relative location of the city within the tour with respect to the coupon purchase. We have developed this model to explore the idea of additional tasks in scheduling that are not directly involved in the schedule but which modify the environment. The proposed formulation is only exemplary and can be modified, for example, to accommodate multiple coupon buying opportunities, which decrease the cost of travel in a successive manner. In this case, the costs might represent the transitions between batches of different products. The transition costs can be lowered by executing a task, such as a set of lab tests, after each batch.

2.2 Illustrative Example : Deterministic TSP with a discount coupon

Consider a small size(10 cities) TSP with the option of buying a coupon. In this example, the coupon prices were chosen by drawing 10 random numbers from a uniform distribution from 0 to 120 and assigning them to stages 1 through 10 in order of decreasing value. The discount factor for each cost element was also drawn from a uniform random distribution between 0 to 0.8. The details of the particular cost matrices and coupon prices can be found at *(http://dot.che.gatech.edu/ Information/research/issicl/FORCAPO2003CRL.pdf).* For the original TSP, without the optional task, a large number of algorithms and heuristics have been developed (Lawler and Eugene, 1985). But with the optional task these methods may not apply, at least not directly. In this paper, four different solution methods will be introduced, including our novel approach based on combining heuristics with dynamic programming. Each approach represents a different level of compromise between the accuracy of solution and computational complexity.

2.3 Dynamic Programming

Dynamic Programming is a technique that can be used to solve optimization problems with a certain multi-stage structure. Dynamic programming obtains solutions by working stage by stage, usually backward from the last stage to the first stage, thus breaking up a large, unwieldy problem into a series of smaller, more tractable, problems. The original TSP has been formulated as a dynamic programming problem (Winston, 1993) (Bertsekas, 1995), and we modify this for our particular variant.

2.3.1. Definition of State The *state*, denoted by X_t, consists of three pieces of information: the first two are the current city, i, and the set of cities already visited before the current stage t, which is denoted by S_t. These two are the states used for the original TSP. The additional state information is a binary variable,γ_t, indicating whether or not the coupon has been purchased. It takes the value of 1 if the coupon has already been purchased before stage t, and 0 if not, this will be termed the coupon status. Hence, the state for our problem is:

$$X_t = (i, S_t, \gamma_t) \tag{1}$$

Once the state is defined, the DP recursion can be formulated using the following equations.

$$\begin{aligned} f_t(X_t) = & \min_{j \notin S_t, j \neq 1, \delta_t \in \{0,1\}} \{g_t(X_t, j, \delta_t) \\ & + f_{t+1}(X_{t+1})\} \text{ for } t = \{1, 2, ..., N\} && (2) \\ f_N(X_N) = & \, 0 \ \ \forall X_N && (3) \\ X_{t+1} = & \, (j, S_t \cup j, \gamma_t + \delta_t), \text{s.t. } \gamma_t + \delta_t \leq 1 && (4) \end{aligned}$$

where $t = 1, 2, ..., N$ for N-cities TSP and $f_t(X_t)$ represents the minimum cost that must be incurred to complete a tour if the $t-1$ cities in the set S_t have been visited, city i was the last city visited, and the coupon has been purchased already if $\gamma_t = 1$ (or not purchased if $\gamma_t = 0$). δ_t is introduced to represent the decision of purchasing the coupon at stage t. According to the problem definition in 2.1, the salesman can buy the coupon only once throughout the tour; therefore δ_t is constrained to be 0 if γ_t is 1. γ_{t+1} can be expressed as $\gamma_t + \delta_t$. In the equation (2), the current stage cost, g_t, is calculated by following equations:

$$\begin{gathered} g_t = c_{ij}^O; \text{ if } \gamma_t = 0 \text{ and } \delta_t = 0, \\ g_t = c_{ij}^D + CP_t; \text{ if } \gamma_t = 0 \text{ and } \delta_t = 1, \\ g_t = c_{ij}^D; \text{ if } \gamma_t = 1 \text{ and } \delta_t = 0 \end{gathered}$$

where c_{ij}^O is the cost of traveling from the city i to j before buying the coupon, c_{ij}^D is the cost for doing the same once the coupon has been purchased, and CP_t is the coupon price at stage t.
The DP approach keeps tracking the feasible state transitions between stages while finding minimum cost-to-go at each stage.

2.3.2. Computational Load of Dynamic Programming The computational load of this DP algorithm is directly dependent on the size of state space. Since one must always start from city 1, there is only 1 state for the first stage. Also, since one must end at city 1 and one has the option of not buying the ticket at all, there are two possible states for the last stage.

From stage 2 to N, the number of possible states for stage t,(NP_t), can be calculated by following equation (5)

$$NP_t = \frac{2(N-1)!}{(N-t)!(t-2)!} \tag{5}$$

where, $N =$ the number of cities , $t = \{2, 3, ..., N\}$ stage number. Because the DP solves the problem though stage-wise recursion, the number of comparisons at each stage t is given by the multiplication of the number of states at stage t and the number of states at stage $t+1$. Considering the state transition rules (for example, one cannot "unbuy" the previously bought coupon) and using the equations in (5), the total number of comparisons required to solve this problem by DP can be calculated. There are NP_t and NP_{t-1} states at stage t and $t-1$ respectively, therefore without considering state transition rule, there can be $NP_t \cdot NP_{t-1}$ cost-go-values incurred by connecting states at t stage and states at $t-1$ stage. But the state transitions from "not bought" states at stage t to "bought" states at stage $t-1$ are not allowed. Therefore, at stage t, $\frac{3}{4} NP_t \cdot NP_{t-1}$ comparisons of cost-to-go are required.

For this example(10 cities), the number of possible states is 4612 and the number of comparisons is 2779974 calculated by $\sum_{t=1}^{N} (\frac{3}{4} NP_t \cdot NP_{t-1})$. On the other hand, to solve this problem with explicit enumeration requires $(N!) \ln(N!) =$

54810892 comparisons. Despite the superiority of DP to explicit enumeration, it is limited to fairly small TSPs. For example for a 50-city TSP, the total number of states goes up to 2.76×10^{16}. The computational load scales exponentially with the number of cities and the approach quicly becomes intractable. The solution obtained by using the DP approach for the given example is listed and compared with the solutions from the other methods in subsection 2.7.

2.4 MILP Formulation

The MILP formulations for the original TSP have been developed by adding constraints for eliminating subtours in the assignment problem (Lawler and Eugene, 1985). Unfortunately, these classical MILP formulations are not directly applicable to our variant of the TSP. The solution of our variant of the TSP is an incomplete tour if we consider it as a TSP with $2N$ cities (N cities with original cost matrix and N cities with discounted cost matrix). The compact subtour elimination constraints cannot be used to develop an MILP model for the given problem.

An MILP model for the given problem can be derived by modifying the assignment problem and shown in the full version of this paper posted on the web.*(http://dot.che.gatech.edu/ Information/research/issicl/FORCAPO2003CRL.pdf)*.

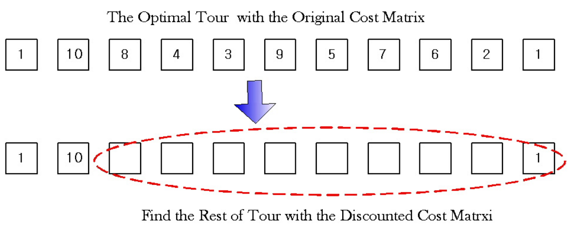
2.5 Heuristics

For the original TSP, many heuristics have been developed to find suboptimal but "good" solutions in reasonable amount of time for large N ($> 10^6$) TSPs. For our TSP example, we consider two heuristics to find suboptimal solutions. The main idea behind the heuristics is to solve a TSP and then modify the solution using a shortest path problem to reflect the change in the cost matrix that occurs after buying the coupon.

2.5.1. Heuristic 1 This heuristic can be described by the following procedure and Figure 1.

(1) Solve the TSP with the original cost matrix and obtain the optimal tour, set $i = 1$
(2) For the option of purchasing the coupon at the ith city in the optimal tour, follow the obtained optimal tour until the ith city
(3) Solve the shortest path problem with the discounted cost matrix for the rest of the tour after the ith city
(4) $i = i + 1$, while $i \leq N$, repeat from 2.

Heuristic 1 determines the first part of the tour from the optimal tour obtained from the original cost matrix. The tour after purchasing the coupon is found by solving a shortest path problem through the rest of the cities. *Heuristic 1* generates N different suboptimal solutions with N different coupon buying locations for the N city problem. Another N suboptimal solutions can be obtained by inverting the optimal tour with the original cost matrix and proceeding as before.



The Optimal Tour with the Original Cost Matrix
1
10
8
4
3
9
5
7
6
2
1
1
10
1
Find the Rest of Tour with the Discounted Cost Matrxi

Fig. 1. Heuristic 1

2.5.2. Heuristic 2 This heuristic follows the same idea as in *Heuristic 1* but we reverse the sequence. That is, we first determine the optimal tour for the regular TSP with the discounted cost matrix. In this case, the tour obtained with the discounted cost matrix gives the second part of the suboptimal tour because the discounted cost matrix is in effect after buying the coupon.

2.6 DP in the Sub-Set of the States

DP is shown in 2.3 as a solution method that can guarantee the global optimum for this type of problem. But DP is often not applicable to practical problems due to the exponential growth of the state space. In theory, the portion of the state space that has to be visited by an "intelligent" algorithm consists of just N states, those on the optimal path, a vanishingly small fraction of the overall state space. Identifying this subset of states, without searching the state space, must clearly be as intractable as solving the original problem. However, finding some small regions of the state space that might contain the optimal (or very good suboptimal) subset and then searching them rigorously, using DP, could prove to be a tractable approach for large problems. The idea is to use heuristics to identify the relevant regions of the states and use the DP to "patch" these states together. The proposed method in this section describes how to obtain this *relevant subset of the states* and find an optimal path of states within the subset.

In subsection 2.5, two heuristics for the given problem are developed based on the idea of first fixing the stage at which the coupon is bought. For many well-known types of combinatorial optimization problems, a large number of heuristics can be and have been developed. Oftentimes, certain heuristics can be modified by changing a parametric description of them. For example, the 'Nearest Neighborhood' search (Winston, 1993), a well known greedy algorithm for TSP, can be parameterized by an explicit description of the neighborhood operator.

The N city problem has N stages, thus from one suboptimal solution, N visited states are obtained. Since $4N$ suboptimal solutions can be obtained from the two heuristics we introduced earlier, at most $4N$ states get visited by these heuristics at each stage. The same state can be visited by different suboptimal solutions and this will happen more frequently when heuristics that exploit the problem structure in a similar manner are used. The key idea of our method comes from hypothesizing that the important states visited by several reasonable heuristics can be put together as a 'good' subset of the state space, within which search for a 'good' solution can be conducted. The reduction of the search space is often dramatic, thus making the approach feasible for even very large problems. For the given 10 city illustrative example, the size of this subset of the states(101 states) is much smaller than the entire state space (4612 states), which is used for the DP in subsection 2.3. The DP in the subset of states follows the same algorithms as the full DP, except some of the states in adjacent stages cannot be connected. We can state several features of this approach.

(1) The original 4N paths guarantee that a feasible solution exists.
(2) If no states can be connected from different heuristics, the procedure will be no better than the original heuristics.
(3) There is a chance of finding an improved solution by connecting states in the subset of states found by different heuristics.
(4) If the state space includes all of the states of the global optimum solution, eventually DP will lead to it – often in dramatically reduced computational time.

The reduction of the state space to the subset visited by the two heuristics enables a considerable reduction in the computational load compared to the original DP.

2.7 Comparison of Solutions

For the illustrative example, 4 different solution methods are proposed. Among these 4 solution methods, the *DP* and *MILP* approaches can guarantee the global optimum solution. On the other hand, the other two methods, the *heuristics* and *DP in the subset of the states* are computationally more tractable, even though they cannot guarantee the global optimum solution. The comparison of solutions by the four methods must be based on two points, optimality of the solutions and the computational time used to obtain the solutions. However, comparing the MILP method with the other methods is not appropriate because different languages were used to pose and solve the proposed MILP from the other solution methods. The proposed MILP is solved by using *CPLEX 7.0* in *GAMS* (Brooke *et al.*, 1998) and *MATLAB* is used for the other solution methods. Generally, the speed of computing with *MATLAB* is much slower than that with GAMS, which uses Fortran as a base computational language. For this reason, in table 1, we compare the computational times for the three solutions obtained by *MATLAB* only. However, it should be noted that the computational time of solving the proposed MILP with 0.01 error bound on same machine is 1015.0 seconds.

As a part of the two heuristics, one must solve the original TSPs without the coupon buying option. For this purpose, a TSP solver is coded in *MATLAB* with the *simulated annealing algorithm* (Aarts and Korst., 1989). With a good initial solution from the simulated annealing, a greedy heuristic for the original symmetric TSP, the Nearest Neighborhood Search (Winston, 1993), can find the globally optimal or nearly optimal solution for relatively small size(less than 100 cities) instances of TSP. The *simulated annealing algorithm* TSP solver starts its stochastic cooling from a temperature of 60 until it cools down to 5 with a reduction rate of 0.99. For most TSPs with less than 50 cities, the TSP solver can find the global optimal solution owing to the good initial solution and the high temperature reduction rate(0.99).

	DP Solution	The Best Heuristic	DP in the SSS[+]
Total Traveling Cost	416.34	432.54	422.78
Calculation Time (Sec.)[*]	30546.8	8.80	8.8+1.8=10.6

[+] Sub-Set of the States
[*] On a Pentium III at 800 MHz: 512MB RAM

Table 1. Solution Comparison

As we expected, the DP method finds the global optimal solution and the other methods result in suboptimal solutions. Table 1 highlights the efficiency of the proposed method of performing DP within the subset of the states visited by the heuristics. The additional computational time for performing DP within the visited set is trivial because of the dramatically reduced state space. At the expense of small additional calculation time on top of the heuristics, we can obtain a significantly improved solution.

	Solution Method	Traveling Cost	Route
The Global Opt.	DP or MILP	416.34	1-10-8-4-***2-6-7-5-9-3-1***
2nd Ranked Soln	*DP in the SSS*	422.78	1-10-8-2-***6-7-5-9-3-4-1***
3rd Ranked Soln	Enum+	427.84	1-10-8-4-6-***7-5-9-3-4-1***
4th Ranked Soln	Enum+	431.23	1-8-2-6-***10-7-5-9-3-4-1***
5th Ranked Soln	Heu*	432.54	1-10-6-2-***8-5-7-9-3-4-1***

Italic Bold represents discounted tour
$^{+}$ Solution method: Enumeration
* Solution Method: Heuristics

Table 2. Feasible Solutions between the Best Heuristic Solution and the Global Optimum

Furthermore, we can see that the solution from the *DP in the subset of the states* approach is not that far from the global optimum in this particular example. To measure the quality of each solution by different solution method, exhaustive enumerations are performed. The best 5 feasible solutions obtained by exhaustive enumerations of all feasible solutions are shown in table 2 and it turns out the solution by the method ranks second.

3. CONCLUSIONS AND FUTURE STUDY

This paper has presented a new framework for improving heuristics of deterministic optimization problems. The key idea of the proposed method is to perform DP in a subset of the states visited by reasonable heuristics. To test the proposed mathematical framework a new variants of the deterministic and stochastic TSP has been introduced. This variant includes an optional task that change the problem cost structure. Four different solution methods,DP, MILP, heuristics and DP in the subset of the states were applied to this problem. The performance of these 4 solution method was tested for a 10-city illustrative example. Among the 4 solution methods, DP in the sub-set of the states shows significant advantages in computational time and solution quality. The performance of this method was also tested for larger examples of the variant TSP that are computationally intractable with other conventional methods. The proposed method showed good performance in these problems. An important extension to this approach, and a subject of future work, is to develop reasonable systematic methods for expanding the subset of states that is searched via DP. Finally, the basic idea of the proposed method, solving optimization problem through the rigorous search of a solution space that is comprised of the states found by suitable heuristics is quite general. We expect it can be applied to many types of optimization problems, multi-stage, stochastic, or multi-objective, as long as some initial heuristics exist for their solution.

Note : A larger version of this paper with the extension of the proposed method to a stochastic problem can be found on following URL.
The paper also includes larger size(50 cities) deterministic TSPs with random parameter generation. ***(http://dot.che.gatech.edu/Information /research/issicl/FORCAPO2003CRL.pdf)***

REFERENCES

Aarts, E. and J. Korst. (1989). *Simulated annealing and Boltzmann machines : a stochastic approach to combinatorial optimization and neural computing.* 1st ed.. Wiley Press.

Bertsekas, DP. (1995). *Dynamic Programming and Optimal Control.* Vol. 1,2. 2nd ed.. Athena Scientific.

Brooke, AN., D. Kendrick, A. Meeraus and R. Raman (1998). *GAMS : A User's Guide.* 1st ed.. GAMS Development Corporation.

Garey, M.R. and D.S. Johnson (1979). *Computers and interactability : a guide to the theroy of NP-completeness.* 1st ed.. W.H. Freeman.

Gooding, WB, JF Pekny and PS Mccroskey (1994). Eunymerative approaches to parallel flowshop scheduling via problem transformation. *Computers and Chemical Engineering* **18**(10), 909–927.

Lawler, E.L. and L Eugene (1985). *The Traveling Salesman Problem : A guided Tour of Combinatorial Optimization.* 2nd ed.. Wiley Press.

Pekny, JF, DL Miller and GK Kudva (1993). An exact algorithm for resource constrained sequencing with application to production scheduling under an aggregate deadline. *Computers and Chemical Engineering* **17**(7), 671–682.

Penky, JF and DL Miller (1991). Exact solution of the no-wait flowshop scheduling problem with a comparison to heuristisc methods. *Computers and Chemical Engineering* **15**(11), 741–748.

Winston, WL. (1993). *Operations Research : Applications and Algorithms.* 3rd ed.. Duxbury Press.