A new generalization of the Traveling salesman problem

Article *in* Applied and computational mathematics · January 2010

CITATIONS
6

READS
98

2 authors:

Ali Fuat Alkaya
Marmara University
19 PUBLICATIONS **111** CITATIONS
SEE PROFILE

Ekrem Duman
Ozyegin University
47 PUBLICATIONS **462** CITATIONS
SEE PROFILE

Appl. Comput. Math., V.9, N.2, 2010, pp.162-175

A NEW GENERALIZATION OF THE TRAVELING SALESMAN PROBLEM*

ALI FUAT ALKAYA[1], EKREM DUMAN[2]

Abstract. Traveling Salesman Problem (TSP) is one of the well-known NP-Complete combinatorial optimization problems. Adding new constraints yields different generalizations of the problem, and each new generalization forms the basis of a new research area. The main contribution of this study is to define and formulate a new generalization of the TSP, which we call the Sequence Dependent TSP (SDTSP). In SDTSP, the cost of traveling between two vertices depends not only on the distance between these vertices, but also on the characteristics of a number of vertices to be visited next. The problem is formulated as a nonlinear integer programming. Then a real life problem environment where this problem appears is described. Some discussions on previous solution attempts to this problem and on closely related problems are also given. We believe that with the definition of the SDTSP, a basis for new research area will be established.

Keywords: Traveling Salesman Problem, Integer Programming, Combinatorial Optimization.

AMS Subject Classification: 90C10, 90C27, 90C30, 68Q17.

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most well-known NP-Hard combinatorial optimization problems. After a small survey on TSP, one can easily argue that there are numerous studies towards solving it since it has been introduced to the literature, ranging back to at least the late 1940's. The TSP is the focus of interest for many research disciplines (mostly industrial engineers, computer scientists, and mathematicians), becaus, even after about half a century of research, the problem has not been completely solved. This is because of the fact that the TSP, its variants, and problems alike, posed such computational complexity, that any effort to solve such problems would grow super-polynomially with the problem size. These categories of problems became known as NP-Complete [13].

We observe the spirit of the TSP within many practical applications in real life. For example, a mail delivery person tries to figure out the optimal route that will cover all of his/her daily stops, a network architect tries to design the most efficient ring topology that will connect hundreds of computers, and a manufacturing engineer tries to design the shortest sequence for assembling components on a printed circuit board (PCB). In all of these instances, the cost or distance between locations, whether they be cities, buildings, or nodes in a network, is known (we use the terms "vertex", "node", "city", and "point" interchangeably in this study). With this information, the goal is to find the optimal tour. That is, to determine a sequence in which each location should be visited only once, and the total distance traveled, or cost incurred, is minimal. In the general TSP, there are no restrictions on the distance/cost values.

*This study is a part of a project supported by the Scientific and Technological Research Council of Turkey under the project ID 108M198

[1]Computer Engineering Department, Marmara University, 34722, Istanbul, Turkey, e-mail: falkaya@marmara.edu.tr

[2] Industrial Engineering Department, Dogus University, 34722, Istanbul, Turkey, e-mail: eduman@dogus.edu.tr

Manuscript received 5 January 2009.

In order to obtain a solution to the TSP, several types of solution methodologies are developed. Of these, exact algorithms should be explained first. Exact algorithms are guaranteed to find an optimal solution to every instance of a class of combinatorial optimization problems. The run-time, however, often increases dramatically with the instance size, and often only small or moderately sized instances can be practically solved to provable optimality. Several exact algorithms are developed for the TSP like problems. We can categorize the exact algorithms as branch-and-bound, cutting planes, and branch-and-cut [6, 14, 18, 19].

Another approach for solving combinatorial optimization problems like TSP or its generalizations, is developing heuristic methods. In heuristic methods we sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time. Thus, the use of heuristic methods to solve combinatorial optimization problems has received more and more attention in the last 30 years [4, 23].

Among the basic heuristic algorithms, we usually distinguish between constructive algorithms and local search algorithms. Constructive algorithms generate solutions from scratch by adding-to an initially empty partial solution-vertices, until a solution is complete. They are typically the fastest heuristic methods, yet they often return solutions of inferior quality when compared to local search algorithms. Local search algorithms start from some initial solution, and iteratively try to replace the current solution by a better solution in an appropriately defined neighborhood of the current solution.

Among the constructive algorithms developed for the TSP, we can list the Convex-Hull and the Nearest Neighbor algorithms [15, 20, 22]. Several local search algorithms are developed for the TSP. Croes developed the first 2-opt algorithm, and not after a decade later, Lin generalized the 2-opt concept to r-opt [8, 17]. Or developed the or-opt algorithm, where he relocates chains of length three, two, and one [20]. Recently, Babin et al. showed that improved Or-opt + 2-opt is an excellent combination, and is easy to implement [3]. All of these local search algorithms put promising results for reaching optimality in a reasonable time. For a complete survey of the TSP, we direct the reader to [15].

Adding new constraints to the problem yields different generalizations of the problem, and each new generalization forms the basis of a new research area. Some of the well known generalizations are the Asymmetric TSP (ASTP), and the TSP with time windows (TSPTW). In ASTP, cost of travel between two cities depends on the direction of travel. So, for example, cost of travel from city A to city B is not necessarily equal to, cost of travel from city B to city A. Hence, ATSP is defined in directed graphs [15]. On the other hand, in TSPTW, for each node i, a triplet $\{a_i, b_i, s_i\}$ is given, and the quantity s_i represents the service time at node i, and the job at node i must be completed in the time interval $[a_i, b_i]$. If the salesman arrives at node i before a_i, he will have to wait until a_i [15].

The Time Dependent TSP (TDTSP) is another generalization of the TSP, in which, cost of travel between node i and node j depends not only on the respective locations involved, but also on their positions in the sequence that defines the tour [21]. So, the cost of travel between two nodes is represented by c_{ijp}, where p index states the position (order) of the trip from location i to location j in the sequence.

As we pointed out above, we observe the spirit of the TSP within many practical applications in real life. One of these practical applications in real life is the PCB manufacturing industry. Most electronic products such as computers, calculators, robots, remote controllers, business telephones, and cellular phones contain PCBs as important components. A PCB is formed by populating (assembling) electronic components by means of automated placement machines, and the use of automated placement machines has aroused some optimization issues, which have attracted the interest of researchers for several decades.

Nowadays, there are many types of placement machines available, such as sequential pick-and-place, dual-delivery, turret type (chip shooter), concurrent pick-and-place, etc. [2]. Various types of placement machines have different characteristics and restrictions [24]. Thus, the PCB

production scheduling process is highly influenced by the type of placement machine being used. For a complete survey on PCB assembly, we direct the reader to [7, 16].

Generally speaking, the operations of these machines yield at least one major problem, which is the determination of component placement sequence [9]. Generally, the determination of component placement sequence is akin to the TSP or its generalizations, such as the Precedence Constrained TSP [10] or multiple TSP [11]. In determination of component placement sequence, the aim is to find the optimal placement order of components, and returning back to the starting point (to get ready for the next PCB) in a minimum time.

In this study, we introduce a new generalization of the TSP which we call the Sequence Dependent TSP (SDTSP). In SDTSP, the cost of traveling between two vertices does not only depend on the distance between these vertices, but also on the characteristics of a number of vertices to be visited next. The SDTSP can be observed in the optimization of a specific electronic component placement machine (as we devoted a section to in this study), or in newest technology placement machines called chip shooters. In the next section, we define the problem and formulate it by means of nonlinear integer programming. In section three, operations of a specific placement machine are described, and thus the SDTSP arising from them is given in detail. Previous work on the analyzed placement machine is also summarized in section three. In the last section, main conclusions are given together with some possible further research areas.

2. Problem Definition

In the classical TSP, the cost of travel between two cities depends only on the distance between them. If Chebyshev measure is used in the Cartesian coordinate system, then the distance between two points i and j, *d(i, j)*, is defined as $\max\{\,|i_x - j_x|, |i_y - j_y|\,\}$ where i_x and i_y represent the x and y-coordinates of point i, respectively. In this study, we calculate cost in time units, hence we define the cost function as follows.

$$CF1(i,j) = \frac{d(i,j)}{v}, \tag{1}$$

where v is a predefined speed value.

If we define a new cost function $CF2$ for each point j, then the problem gets a bit more complicated.

$$CF2(ft_j, i, j) = \max\{\, ft_j,\ CF1(i,j)\,\}, \tag{2}$$

where ft_j is a predefined free time value for each point j. It is called free time, since during this time interval the salesman is free to move to any vertex j without increasing the cost. So, the cost function depends not only on the distance between them, but also the free time value of the destination point. Actually, this problem turns out to be an Asymmetric TSP because $CF2(ft_i, j, i) \neq CF2(ft_j, i, j)$, unless $ft_j = ft_i$. Furthermore, if the free time value for all points is constant for all points, then the problem turns out to be a classical TSP with Chebyshev distance measure.

To generalize the problem that we introduce, the cost of travel between two cities depends on the distance between these cities, on the free time value of the destination city, and, the free time value of a number of following cities (say d). So, the position of each city becomes important because it can affect the cost calculation of travel between former cities. Hence, the problem turns out to be more complex than both the ATSP and the TDTSP. We will call this problem the Sequence Dependent TSP (SDTSP), because the cost calculations are dependent on the travel sequence.

So, for a given travel sequence, cost of travel between points i and j when point j is visited in p^{th} position is calculated by

$$\begin{aligned} CF3(p,i,j) &= \max\{\, CF1(i,j), \max\{ft_p, ft_{p+1}, ft_{p+2}, \ldots, ft_{p+d}\,\}\} = \\ &= \max\{CF1(i,j),\ \max_{m=p}^{p+d}\{ft_m\}\}, \end{aligned} \tag{3}$$

where ft_{p+1} is the free time of the point visited in $(p+1)^{th}$ position, ft_{p+2} is the free time of the point visited in $(p+2)^{th}$ position, and so on. d represents the number of points in the travel sequence whose free time characteristic will be taken into account in the cost calculation. If d is zero, then $CF3$ turns out to be max $\{\, ft_j,\ CF1(i,j)\,\}$ which is equal to $CF2$ and thus the problem becomes an ATSP as shown above. Besides the former condition, if cost of travel from point i to j does not depend on free time, but on its position in the travel sequence, then the problem turns out to be a TDTSP.

Below, we give the mathematical representation of the SDTSP. We start by introducing the notation used:
N: total number of points to be visited
p: the visiting order or visiting position
d: number of cities whose free time must be taken into account
We also need to define, decision variable x_{ip};

$$x_{ip} = \begin{cases} 1, & \text{if node } i \text{ is visited in } p^{th} \text{ position;} \\ 0, & \text{otherwise.} \end{cases}$$

It is important to note a detail about indices used in the formulations below. We assume that traveling along the path restarts after a tour ends. Hence, in the formulations, if an index exceeds its upper limit, it should be considered as if the counting of indices restart from one. For example, if we are talking about, $(p+d)^{th}$ visit when p is 98, d is five, and N is 100, then the $(p+d)^{th}$ visit means the third point of the next tour.

Then, the cost of travel from point i to point j when point j is visited in p^{th} position, C_{ijp} is

$$C_{ijp} = x_{i,p-1}\ x_{jp}\ \max\{CF1(i,j),\ \max_{m=p}^{p+d}\{\sum_{l=1}^{N} ft_l x_{lm}\}\}. \tag{4}$$

Since determining the travel sequence is the objective of the problem, we defined C_{ijp} by incorporating the decision variables x_{ip} into $CF3$. In this formulation, the first term in the outer max function is the time for traveling from i to j. The second term finds the maximum of free time values of the following $d+1$ points to be visited, including point j. The multiplication term calculates the free time for the point visited in m^{th} position, where m states the positions from p to $p+d$. Hence, cost of travel from point i to point j is nonzero if point i is visited in the $(p-1)^{th}$ position and point j is visited in the p^{th} position.

Then the mathematical model for the problem can be stated as follows.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{p=1}^{N} C_{ijp} \tag{5}$$

s.t.

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1, \ldots, N, \tag{6}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1, \ldots, N, \tag{7}$$

$$x_{ip} \in \{0,1\}, \qquad i, p = 1, \ldots, N. \tag{8}$$

Constraint (6) guarantees that each node is assigned to a position, and constraint (7) guarantees that each position is occupied by only one node. There is no need for the constraints of the classical TSP that are used to prevent subtours elimination, because placing each node in exactly one position is achieved by these constraints. The problem may seem easy with this formulation, but it is nonlinear because of maximum functions used in the C_{ijp} definition, and the quadratic term in the objective function. In order to remove the nonlinearity in the objective function, we first introduce D_{ijp} as the dominating factor in the C_{ijp} definition.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \tag{9}$$

s.t.

$$D_{ijp} - x_{i,p-1}\, x_{jp}\, CF1(i,j) \geq 0, \qquad i,j,p = 1,\ldots,N, \tag{10}$$

$$D_{ijp} - x_{i,p-1}\, x_{jp} \sum_{l=1}^{N} ft_l x_{lp} \geq 0, \qquad i,j,p = 1,\ldots,N, \tag{11}$$

$$D_{ijp} - x_{i,p-1}\, x_{jp} \sum_{l=1}^{N} ft_l x_{l,p+1} \geq 0, \qquad i,j,p = 1,\ldots,N, \tag{12}$$

$\vdots$

$$D_{ijp} - x_{i,p-1}\, x_{jp} \sum_{l=1}^{N} ft_l x_{l,p+d} \geq 0, \qquad i,j,p = 1,\ldots,N, \tag{13}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\ldots,N, \tag{14}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\ldots,N, \tag{15}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\ldots,N, \tag{16}$$

$$D_{ijp} \geq 0 \ \text{ and } \ \text{integer}. \tag{17}$$

This formulation now, a linear objective function, but it still has nonlinear terms in constraints. The quadratic term $x_{i,p-1}x_{jp}$ in (10)-(13) can be eliminated by applying the following rules [27]:

- Replace the product term xy by a binary variable w.
- Impose the logical condition w=1 if and only if x=1 and y=1 by means of the extra constraints $w \leq x$, $w \leq y$ and $w \geq x + y - 1$.

Hence, we introduce the variable w_{ijp} with the following interpretation.

$$w_{ijp} = \begin{cases} 1, & \text{if node } j \text{ is visited in } p^{th} \text{ position after node } i \text{ was visited in } (p-1)^{th} \text{ position;} \\ 0, & \text{otherwise.} \end{cases}$$

Then, the problem can be stated as

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \tag{18}$$

s.t.

$$D_{ijp} - x_{i,p-1}\, x_{jp}\, CF1(i,j) \geq 0, \qquad i,j,p = 1,\ldots,N \tag{19}$$

$$D_{ijp} - x_{i,p-1}\, x_{jp} \sum_{l=1}^{N} ft_l x_{lp} \geq 0, \qquad i,j,p = 1,\ldots,N \tag{20}$$

$$D_{ijp} - x_{i,p-1}\, x_{jp} \sum_{l=1}^{N} ft_l x_{l,p+1} \geq 0, \qquad i,j,p = 1,\ldots,N \tag{21}$$

$\vdots$

$$D_{ijp} - x_{i,p-1}\, x_{jp} \sum_{l=1}^{N} ft_l x_{l,p+d} \geq 0, \qquad i,j,p = 1,\ldots,N \tag{22}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1, \ldots, N \tag{23}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1, \ldots, N \tag{24}$$

$$w_{ijp} \leq x_{i,p-1}, \qquad i, j, p = 1, \ldots, N,\ i \neq j \tag{25}$$

$$w_{ijp} \leq x_{j,p}, \qquad i, j, p = 1, \ldots, N,\ i \neq j \tag{26}$$

$$w_{ijp} \geq x_{i,p-1} + x_{jp} - 1, \qquad i, j, p = 1, \ldots, N,\ i \neq j \tag{27}$$

$$x_{ip} \in \{0, 1\}, \qquad i, p = 1, \ldots, N \tag{28}$$

$$w_{ijp} \in \{0, 1\}, \qquad i, j, p = 1, \ldots, N \tag{29}$$

$$D_{ijp} \geq 0 \ \text{ and } \ \text{integer}. \tag{30}$$

Constraints (25)-(27) can be stated more compactly after the following mathematical discussion. If we sum up the terms in (25)-(27) across the i index, we obtain

$$\sum_{i=1}^{N} w_{ijp} \leq \sum_{i=1}^{N} x_{i,p-1}, \qquad j, p = 1, \ldots, N,\ i \neq j \tag{31}$$

$$\sum_{i=1}^{N} w_{ijp} \leq \sum_{i=1}^{N} x_{j,p}, \qquad j, p = 1, \ldots, N,\ i \neq j \tag{32}$$

$$\sum_{i=1}^{N} w_{ijp} \geq \sum_{i=1}^{N} x_{i,p-1} + \sum_{i=1}^{N} x_{jp} - N, \qquad j, p = 1, \ldots, N,\ i \neq j. \tag{33}$$

Equation (31) can be rewritten by using equation (24) as follows

$$\sum_{i=1}^{N} w_{ijp} \leq 1, \qquad j, p = 1, \ldots, N,\ i \neq j. \tag{34}$$

If $x_{jp} = 1$, then by using Equations (33) and (34) we can say that

$$1 \leq \sum_{i=1}^{N} w_{ijp} \leq 1, \qquad j, p = 1, \ldots, N,\ i \neq j \ \Leftrightarrow \ \sum_{i=1}^{N} w_{ijp} = 1, \qquad j, p = 1, \ldots, N,\ i \neq j. \tag{35}$$

If $x_{jp} = 0$, then by using Equations (29) and (32) we can conclude that

$$\sum_{i=1}^{N} w_{ijp} \leq 0, \qquad j, p = 1, \ldots, N,\ i \neq j \ \Leftrightarrow \ \sum_{i=1}^{N} w_{ijp} = 0, \qquad j, p = 1, \ldots, N,\ i \neq j. \tag{36}$$

Hence, we can conclude that

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \qquad j, p = 1, \ldots, N, \tag{37}$$

After summing up the terms in (25)-(27) across the j index, and carrying out a similar discussion as above, we can easily conclude that

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \qquad i, p = 1, \ldots, N, \tag{38}$$

So, constraints (25)-(27) can be stated by the equations (37) and (38).

Why we transformed the equations into a different formulation, is because we want to interpret these constraints more explicitly according to our context. Constraint (37) states that if point j is assigned to p^{th} position, then exactly one point precedes j in the travel sequence. Similarly

(38) states that if point i is assigned to $(p-1)^{th}$ position, then exactly one point proceeds it in the travel sequence.

The complete formulation of the SDTSP is now given below:

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \tag{39}$$

s.t.

$$D_{ijp} - x_{i,p-1}\, x_{jp}\, CF1(i,j) \geq 0, \qquad i,j,p = 1,\ldots,N, \tag{40}$$

$$D_{ijp} - x_{i,p-1}\, x_{jp} \sum_{l=1}^{N} ft_l x_{lp} \geq 0, \qquad i,j,p = 1,\ldots,N, \tag{41}$$

$$D_{ijp} - x_{i,p-1}\, x_{jp} \sum_{l=1}^{N} ft_l x_{l,p+1} \geq 0, \qquad i,j,p = 1,\ldots,N, \tag{42}$$

$\vdots$

$$D_{ijp} - x_{i,p-1}\, x_{jp} \sum_{l=1}^{N} ft_l x_{l,p+d} \geq 0, \qquad i,j,p = 1,\ldots,N, \tag{43}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\ldots,N, \tag{44}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\ldots,N, \tag{45}$$

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \qquad j,p = 1,\ldots,N, \tag{46}$$

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \qquad i,p = 1,\ldots,N, \tag{47}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\ldots,N, \tag{48}$$

$$w_{ijp} \in \{0,1\}, \qquad i,j,p = 1,\ldots,N, \tag{49}$$

$$D_{ijp} \geq 0 \ \text{ and } \ \text{integer}. \tag{50}$$

As a case study of the problem, we studied on a chip placement machine. In the next section, we will explain the properties and working principles of the machine, and describe the problems that arise from using it.

3. Case Study - Improving the Operations of a Chip Mounter Machine

One of the examples that SDTSP can be observed is the optimization process of placement machines. The machine type considered in this study is the TDK brand, model RX-5A placement machine. This machine is usually encountered in assembly environments deploying surface mount technology (SMT). It is referred to as "chip mounter" by the manufacturer, and we will adopt this name in the study. In order to formulate the arising problem, we should first describe the operations of the machine. The machine is explained in detail in [12], but we will describe the basic features of the machine without a need for directing the reader to it.

3.1. **Description of Operations.** Chip mounter has a rotational turret consisting of multiple heads, which are responsible for picingkup the components from the component magazine, and placing them on the PCB. The component magazine (also called the feeder mechanism)is where the component tapes are installed it is a circular, stationary structure behind the machine. The rotational turret, has 72 heads, and collects the components to its heads (one to each head) in the placement order, from the component tapes placed in the component magazine, while rotating in a clockwise direction (Figure 1). Even though the machine actually has 72 heads, in the figure we depicted a few of them for the sake of clarity. The head reaching the "placement location" over the board moves down and makes the placement to the precise location on the PCB, which is pre-aligned with the placement position by the board carrier actions. Thus, each head makes the placement exactly at the same Cartesian coordinate, and the alignment of the placement point is made by the board carrier through its simultaneous and independent movements (leading to a Chebyshev distance measure) in the $x - y$ plane.

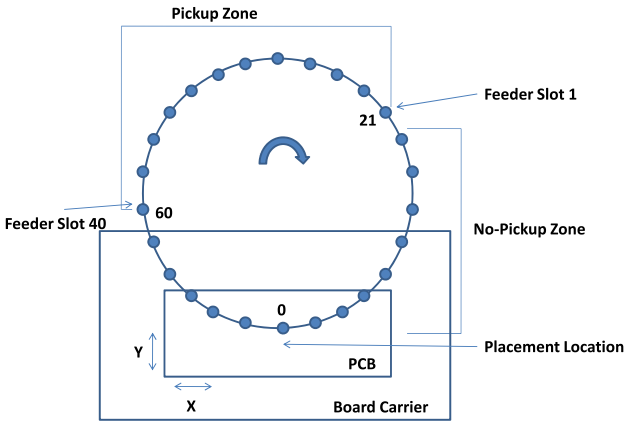


Figure 1. The Chip Mounter Machine.

These machines may handle component types of different weights (each head is equipped with three suction nozzles compatible with different weight categories). When any of the heads picks up a heavier component type, the rotation speed is reduced. For the particular machine considered, there are four discrete speed values corresponding to different component weight categories. These are 0.20, 0.23, 0.33, and 0.40 s per five degrees (or per rotational movement of a distance of one head) ordered from lightest to heaviest components. On the other hand, the board carrier movement speed in both x and y directions is 120 mm per second, which could practically be taken as constant (i.e. no acceleration or deceleration).

The component tapes are mounted to a component magazine behind the machine, and this whole system is stationary, that is, it does not move in any direction. If the placement heads are numbered with 0 to 71 in a counter clockwise manner, with number 0 being the placement head, then up to 120 component tapes can be installed to the feeder slots along heads 21 and 60, which are in the pickup zone. There are 40 slots, and depending on their width, one to three component types can be installed on a slot. The pickup times of the components from the magazine is completely determined, by the order of their placement. Once the placement sequence is determined and the position of each component type in the magazine is entered to the computer, the component type that each head arriving at the placement location carries is known and fixed a priori. In other words, what component type should each and every head picks up is known. Then, during the rotational movement of the turret, the head(s) situating over the correct component tape move down, and pick a component.

To summarize, a placement cycle of such a machine can be itemized as follows:

Item 1:

- turret rotates, and the next placement head comes over the PCB
- board carrier aligns the new placement point under the placement head
- placement heads rotate if necessary to align the suction nozzle carrying the component to be placed

Item 2:

- the placement head moves down, makes the placement, and moves up
- heads in the pickup zone that are above the appropriate component tapes move down and pick up components.

In the above cycle, the actions under each item are performed concurrently, and there is a sequential order between items one and two. The definition of the PCB assembly time minimization problem arising from the use of this kind of a machine is discussed in the next section.

There are two problems that must be formulated and solved for this machine. The first one is the placement sequencing problem, and the other one is the feeder configuration problem. For a given placement sequence, the feeder configuration problem is effectively solved in [12]. In this study, we will focus on the placement sequencing problem, and show how it turns out to be a SDTSP by assuming that the feeder configuration is given.

3.2. **Placement Sequencing Problem.** Given a feeder configuration, the objective of the placement sequencing problem is to find a placement sequence of the components so that the assembly process is completed in the shortest time possible. In order to express placement sequencing problem, we need to define the time between the completion of consecutive placements at points x and y. The time between consecutive placements at points x and y is defined by $CF2$.

Recall that $CF2$ takes the maximum of two values, and the analyzed machine has two simultaneously operating parts, and in order to complete a placement operation, the two parts must finish their operations. The first part is the rotational turret, and tt_k denotes the time it needs to complete its operation. Formally, we define turret time, tt_k, as the turret rotation time required for the next placement head to arrive over the PCB. Turret time is also known as the free time, since during this time interval the board carrier is free to move to any placement point without increasing the cost. In other words, the points at a Chebyshev distance of free time or less, can be taken as equidistant points. When the component types to be placed are categorized according to their weight, we observe that there are four types of component types to be placed on the PCB. The speed of the turret is determined according to the heaviest component type carried. Hence, tt_k takes four values such that tt_1=0.20 s, tt_2=0.23 s, tt_3=0.33 s and tt_4=0.40 s. The second part of the machine is board carrier, and the time for board carrier to align the PCB under the placement head of the turret is calculated by $CF1$ as given in equation (1).

If the rotational turret rotates in a unique speed value, then the problem turns out to be a classical TSP with Chebyshev measure. Thus, what makes the problem complicated is the varying rotational speed of the turret, and this property yields to the SDTSP.

In order to further clarify how a complex problem arises, we need to make the following discussion. In our case, turret time can take four possible values where its value is determined by the heaviest component type being carried by one of the heads of the turret. In other words, for the placement of component j on head number 0,then the rotational speed value corresponding to the heaviest component being carried by the heads numbered 0-59, will be the current turret time value for this rotational step of the turret. Cost of travel between two vertices i and j depends on vertice j's, and following up to 59 components' free time value in the travel sequence. In such a case, cost calculation can be stated by $CF3$, and thus, the problem turns out to be an SDTSP, as we explain below. Before that, we need the following notation and definitions in order to formulate the problem.

N: total number of components to be placed,

n: number of component types,
K: number of weight categories (K=4 in our case),
ct_{it}: component type matrix ($N \times n$) indicating the component type for each component

$$ct_{it} = \begin{cases} 1, & \text{if component } i \text{ is of type } t; \\ 0, & \text{otherwise.} \end{cases}$$

g_{tk}: group (weight category) matrix ($n \times K$) indicating the group for each component type

$$g_{tk} = \begin{cases} 1, & \text{if component type } t \text{ is in group } k; \\ 0, & \text{otherwise.} \end{cases}$$

tt_k : turret rotation time for each weight category k, (when k=1, the turret rotation speed is the maximum, that is, turret rotation time is minimum).

Components (points) can have four different free time values, and they are specified by the weight category they belong to. Each component is of a component type, each component type belongs to a weight category or group, and each group has turret time tt_k value. Hence, in this case study, free time value for each point is not given readily, but rather will be specified upon its component type and its component type's group. Then, free time value for a component j is given by equation (51).

$$ft_j = \sum_{t=1}^{n} \sum_{k=1}^{K} tt_k g_{tk} ct_{jt}. \tag{51}$$

After the placement of component i is completed, the placement time of component j is dependent on, distance between points i and j, and the free time value for succeeding up to 60 components the including j. So, the placement sequencing problem turns out to be the SDTSP defined in the previous section, with the free time values calculated using equation (51).

3.3. **Related Work.** Even though it is stated that the placement sequencing problem is a hard one even to formulate, a solution methodology is developed in [12]. The developed methodology is based on the observation that mixing heavy and light components in the placement sequence is quite inefficient. So, given the feeder configuration, it seems advantageous to formulate separate TSPs for the groups of components in different weight categories. Accordingly, it seems to be a good idea to place all of the lighter components first, and then the heavier ones (or, vice versa). This way, the tt_k values corresponding to each weight category would be constant and it would be possible to use the TSP formulation to find the placement sequences within each weight category. This is the idea behind the Assembly Time Minimization Algorithm (ATMA) that is suggested in [12] for the solution of the placement sequencing problem. In ATMA, after finding the TSP routes for each weight category, the routes are connected in a cost-efficient way. Specifically, among all group 2 components, the closest one to the last component of the group 1 tour is chosen as the starting point for the tour of group 2 components, so the tours for group 1 and group 2 are connected. This process is repeated for the group 2 and group 3 components, and then for the group 3 and group 4 components.

To summarize, in his solution approach, Duman transformed the SDTSP problem into a number of (four) classical TSP problems. He achieved this by grouping components according to their weight categories. Solutions of these TSPs are found by using Convex-Hull and Or-Opt algorithms [20, 22], and they are connected by a cost efficient way.

Another study towards solving the implicitly defined SDTSP is [1]. In this study, it is shown that reversing the placement process may bring improvements in terms of assembly time. They called it inverse ATMA (iATMA), and placed the components by starting from heaviest to lightest. By applying other local search improvement techniques, more than 4.50% improvement is obtained in assembly time against ATMA.

As discussed above, the free time calculation is dependent on what the incoming 60 heads are carrying at that moment. Although the type of component that these heads would carry is determined by the placement sequence, the timing of loading these heads is determined by the feeder configuration (a head will pick up a component just when it arrives on the appropriate

feeder slot). So, we need to know how the feeder is arranged to calculate the time between two placements. In other words, the placement sequencing problem is dependent on the feeder configuration problem. In this study, and in the above formulation, we assume that the feeder configuration problem is solved, and its result is given.

3.4. **Auxiliary Problem.** Solution approach ATMA (or iATMA) gives rise to a new combinatorial optimization problem. We will define the problem by an example, and call it the auxiliary problem. Consider the following example.

Suppose that in an example with 40 points, ATMA is applied, four groups are specified, and separate routes for each group of points are built and connected through the shortest connection. Let N_k denote the number of points in group k, and assume that d=15, N_1=25, N_2=5, N_3=5, and N_4=5. While calculating the travel time for the first point, points between 2 and 16 will be taken into account. So, travel time will be evaluated by $CF2(ft_1, i, j)$, because points between 2 and 16 are in group 1. Similarly, while calculating the travel time to tenth point, points between 11 and 25 will be taken into account. Thus, for the first 10 points in group 1, the travel cost is calculated with $CF2(ft_1, i, j)$. While calculating the travel time for the 11^{th} point, points between 12 and 26 will be taken into account, but point 26 belongs to group 2, which means that cost function turns out to be $CF2(ft_2, i, j)$. The same applies for the next 4 points. Similarly, $CF2(ft_3, i, j)$ is used for the next 5 points, and $CF2(ft_4, i, j)$ for the last 5 points in group 1. This variability in the ft_k values raises the question: Of the 25 points in group 1 which 10, 5, 5, and 5 should be selected to be placed when the free time value is ft_1, ft_2, ft_3 and ft_4, respectively?

The auxiliary problem is focused on determining the placement sequence of group 1 components. For a generic formulation, we introduce M_k to denote the number of points in group 1 to be visited when the cost function is $CF2(ft_k, x, y)$. The M_k values can be easily calculated according to the N_k and d values, as done in the above example. Also, observe that $\sum_{k=1}^{K} M_k = N_1$.

$$\min \sum_{i=1}^{N_1} \sum_{j=1}^{N_1} \sum_{p=1}^{N_1} C_{ijp} w_{ijp} \tag{52}$$

s.t.

$$\sum_{p=1}^{N_1} x_{ip} = 1, \qquad i = 1, \ldots, N_1, \tag{53}$$

$$\sum_{i=1}^{N_1} x_{ip} = 1, \qquad p = 1, \ldots, N_1, \tag{54}$$

$$\sum_{i=1}^{N_1} w_{ijp} = x_{jp}, \qquad j, p = 1, \ldots, N_1, \tag{55}$$

$$\sum_{j=1}^{N_1} w_{ijp} = x_{i,p-1}, \qquad i, p = 1, \ldots, N_1, \tag{56}$$

$$C_{ijp} = CF2(ft_1, i, j), \qquad p = 1, \ldots, M_1, \tag{57}$$

$$C_{ijp} = CF2(ft_2, i, j), \qquad p = M_1 + 1, \ldots, M_1 + M_2, \tag{58}$$

$$C_{ijp} = CF2(ft_3, i, j), \qquad p = M_1 + M_2 + 1, \ldots, M_1 + M_2 + M_3, \tag{59}$$

$\vdots$

$$C_{ijp} = CF2(ft_K, i, j), \qquad p = \sum_{k=1}^{K-1} M_k + 1, \ldots, N_1, \tag{60}$$

$$x_{ip} \in \{0, 1\}, \qquad i, p = 1, \ldots, N, \tag{61}$$

$$w_{ijp} \in \{0,1\}, \qquad i,j,p = 1,\ldots,N. \tag{62}$$

In this formulation, x_{ip} decision variables and w_{ijp} are used in the same way they were defined previously. This problem is actually a special case of the TDTSP. Constraints (53) and (54) guarantee that each node is assigned to only one position. Hence, subtours are not a problem in the formulation. Constraint (55) states that if point j is assigned to p^{th} position, then exactly one point precedes j in the travel sequence. Similarly, (56) states that if point i is assigned to $(p-1)^{th}$ position, then exactly one point proceeds it in the travel sequence. Cost of travel between two points i and j is calculated by considering its position in the travel sequence, and it is predetermined by the constraints (57)-(62). Thus, the above formulation is the same as the TDTSP formulation given by [26]. As we pointed out in the first section, TDTSP is a generalization of TSP, and some of the basic and recent studies on it can be summarized as follows.

Wiel and Sahinidis applied a Benders decomposition to this formulation in order to speed up lower bound calculation, and then by embedding their bound in a branch and bound algorithm, they solved pure TDTSP instances having up to 18 nodes [26].

An improvement heuristic for the TDTSP, and a method for generating TDTSP instances with known optimal solutions is presented in [25].

Recently, Bigras, et. al consider single machine scheduling problems in sequence dependent setup environments, and show how these problems can be seen as a special case of the TDTSP. They observed that the above formulation gave poor lower bounds, and using such weak lower bounds in a branch and bound algorithm generally leads to large search trees. Stronger lower bounds for the TDTSP formulation are obtained by using cutting plane techniques of integer programming, theory such as subtour cuts in the Dantzig-Fulkerson-Johnson formulation of TSP, 2-matching cuts and clique inequality in the node packing problem. Thus, they illustrate the benefits of using stronger formulations, even though the computation of their Linear Programming (LP) relaxation is more time consuming. Incorporating these improved LP relaxation bounds, an exact branch-and-bound algorithm capable of solving instances having 50 nodes is presented [5].

4. Conclusion

In this study, we introduced a new generalization of the TSP to the literature, and called it the SDTSP. In this generalization of the problem, cost of travel between two cities depends on the distance between these cities, on the free time value of the destination city, and also the free time value of a number of following cities. In the study, it is first shown that the problem is harder than other generalizations of TSP, such as the TDTSP. Then, the problem is formulated using nonlinear integer programming.

Then, we analyzed the operations of a placement machine as a case study where the SDTSP arises from its operation. It is shown that the placement sequencing problem of the analyzed machine turns out to be a SDTSP. Two former studies on the analyzed machine are summarized and an auxiliary problem is formulated, which arises as a consequence of applying the developed heuristics. Moreover, it is shown that the auxiliary problem is actually a TDTSP.

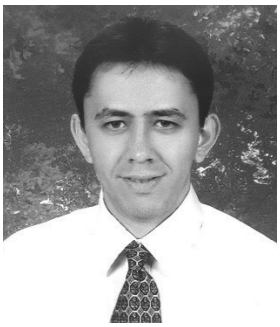
As a main contribution of this study, we can say that a new nonlinear integer programming problem is defined. We believe that for the case described, further improvements can be obtained by developing effective heuristics for both the SDTSP and the auxiliary problem.

References

[1] Alkaya, A. F., Duman E., Eyler, A. Assembly time minimization for an electronic component placement machine, WSEAS Transactions on Computers, V.7, N.4, 2008, pp.326-340.

[2] Ayob, M., Kendall, G. A survey of surface mount device placement machine optimisation: Machine classification, European Journal of Operational Research, V.186, 2008, pp.893-914.

[3] Babin, G., Deneault, S., Laporte, G. Improvements to the Or-opt heuristic for the symmetric travelling salesman problem, Journal of the Operational Research Society, V.58, 2007, pp.402-407.
[4] Berberler, M.E., Nuriyev, U.G. A New Heuristic Algorithm for the One-Dimensional Cutting Stock Problem, Applied and Computational Mathematics, V.9, N.1, 2010, pp.19-30.
[5] Bigras, L.-P., Gamache, M., Savard, G. The Time-Dependent Traveling Salesman Problem and Single Machine Scheduling Problems with Sequence Dependent Setup Times, Discrete Optimization, V.5, N.4, 2008, pp.685-699.
[6] Clausen, J. Branch and Bound Algorithms - Principles and Examples, Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark, 1999.
[7] Crama, Y., Klundert, J.V.D., Spieksma, F.C.R. Production Planning Problems in Printed Circuit Board Assembly, Discrete Applied Mathematics, V.123, 2002, pp.339-361.
[8] Croes, G. A Method for Solving Traveling-Salesman Problems, Operations Research, V.6, 1958, pp.791-812.
[9] Duman, E. Optimization Issues in Automated Assembly of Printed Circuit Boards, Bogazici University, Unpublished PhD Thesis, 1998.
[10] Duman, E., Or I. Precedence Constrained TSP Arising in Printed Circuit Board Assembly, International Journal of Production Research, V.42, N.1, 2004, pp.67-78.
[11] Duman, E. An application of the multiple TSP in printed circuit board assembly, Journal of Operations and Logistics, V.2, N.3, 2009, pp.III.1-III.9.
[12] Duman E. Modelling the operations of a component placement machine with rotational turret and stationary component magazine, Journal of the Operational Research Society, V.58, 2007, pp.317-325.
[13] Garey, M.R., Johnson, D.S. Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Co., 1979.
[14] Gomory, R.E. Outline of an algorithm for integer solutions to linear programs, Bulletin of the American Mathematical Society, V.64, 1958, pp.275-278.
[15] Gutin, G., Punnen, A. The Traveling Salesman Problem and Its Variants, Kluwer Academic Publishers, 2002.
[16] Ji, P., Wan, Y.F. Planning for printed circuit board assembly: The state-of-the-art review, International Journal of Computer Applications in Technology, V.14, 2001, pp.136-144.
[17] Lin, S. Computer solutions of the traveling salesman problem, Bell System Technical Journal, Vol.44, 1965, pp.2245-2269.
[18] Marchand, H., Martin, A., Weismantel, R., Wolsey, L. Cutting planes in integer and mixed integer programming, Discrete Applied Mathematics, V.123, 2002, pp.397-446.
[19] Mitchell, J.E. Branch-and-Cut Algorithms for Combinatorial Optimization Problems, Handbook of Applied Optimization, Oxford University Press, 2002, pp.65-77.
[20] Or, I. Traveling Salesman Type Combinatorial Problems and Their Relation to the Logistics of Blood Banking, Northwestern University, Unpublished PhD Thesis, 1976.
[21] Picard, J.C., Queyranne, M. The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling, Operations Research, V.26, N.1, 1978, pp.86-110.
[22] Stewart, W.R. A computationally efficient heuristic for the traveling salesman problem, Proceedings of the 13th Annual Meeting of Southeastern TIMS, Myrtle Beach, SC, USA, 1977, pp.75-83.
[23] Uysal, M. Using Heuristic Search Algorithms for Predicting the Effort of Software Projects, Applied and Computational Mathematics, V.8, N.2, 2009, pp.251-262.
[24] Wang, W., Nelson, P.C., Tirpak, T.M. Optimization of High-Speed Multistation SMT Placement Machines Using Evolutionary Algorithms, IEEE Transactions on Electronics Packaging Manufacturing, V.22, N.2, 1999, pp.137-146.
[25] Wiel, R.J.V., Sahinidis, N.V. Heuristic Bounds and Test Problem Generation for the Time Dependent Traveling Salesman Problem, Transportation Science, V.29, N.2, 1995, pp.167-183.
[26] Wiel, R.J.V., Sahinidis, N.V. An exact solution approach for the time-dependent traveling-salesman problem, Naval Research Logistics, V.43, N.6, 1996, pp.797-820.
[27] Williams, H.P. Model Building in Mathematical Programming, John Wiley & Sons, New York, 1999.



Ali Fuat Alkaya - received the B.S. degree in Mathematics from Koc University, the M.S. degree in Computer Engineering and Ph.D. degree in Engineering Management from Marmara University. He is currently an Assistant Professor in the Computer Engineering Department and Vice Dean of the Engineering Faculty in Marmara University. His research interests include scheduling, analysis of algorithms and combinatorial optimization.

Ekrem Duman - was born in Afyon, Turkey, in 1967. He received the BS degree in electrical and electronical engineering from Bogazici University. He then received his MS and PhD degrees in industrial engineering from the same university. Since 2001 he has been working for the Industrial Engineering Department of Dogus University. His areas of interests include industrial applications of operations research, scheduling and data mining.