A Constraint Programming Approach for Solving Multiple Traveling Salesman Problem

Masoumeh Vali[1], Khodakaram Salimifard[2]

[1] Department of Industrial Management, Persian Gulf University, Bushehr 75168, Iran
m.vali@mehr.pgu.ac.ir
[2] Department of Industrial Management, Persian Gulf University, Bushehr 75168, Iran
salimifard@pgu.ac.ir

Abstract. The multiple traveling salesman problem (mTSP) is a NP-hard combinatorial optimization problem. It has many real-world applications, for example, the School Bus Routing Problem, and the Pickup and Delivery Problem. In the mTSP, a set of routes is assigned to m salesmen who all start from and return to a home city(depot). In this problem, each other node is located in exactly one tour, the number of nodes visited by a salesman lies within a predetermined interval, and the overall cost of visiting all nodes is minimized. In this study, we discuss how to use constraint programming (CP) to formulate and solve mTSP by applying interval variables, global constraints and domain filtering algorithms. We propose a CP model for the mTSP. The CP-mTSP was tested on a set of benchmark instances from the TSPLIB. Solutions of the CP- mTSP are compared to the ILP-CPLEX of mTSP model and other algorithms (ACO, SW+AS_{elite}, GELS-GA and Enhanced GA) in the literature. The computational results indicate that CP- mTSP performs, on average, quite well compared to mentioned algorithms in terms of the number of identified best-known solutions. CP is well known for its ability to find good quality feasible solutions for complex structured problems within reasonable time.

Keywords: Multiple traveling salesman problem, Constraint programming, interval variable

1 Introduction

One of the most famous Np-hard combinatorial optimization problems is the multiple traveling salesman problem(mTSP). From one side, the mTSP can be considered as a generalization of the Travelling Salesman Problem (TSP) [1], where a set of routes is assigned to m salesmen who all start from and return to a home city. On the other, the mTSP can be considered as a special case of the vehicle routing problem (VRP), in which customers are considered unitary demands and every travelling salesman only visits a predetermined number of cities. Thus, the mTSP can also be utilized for solving several types of VRPs and all formulations and solution approaches for the VRP are valid for the mTSP. Several methodologies have been raised to solve the mTSP, such as heuristic and metaheuristic algorithms, neural network-based methods, ant systems and exact techniques. exact algorithms are based on lagrangean relaxation algorithm

[2], branch-and-cut method [3], etc. Some of the well-known heuristic algorithms are gravitational emulation search [4], local search [5], and lin-kernighan [6]. In recent years, some well-known methods used to solve this problem are the evolutionary algorithms including the genetic algorithm (GA) [7-9], simulated annealing (SA) [9], ant colony optimization (ACO) [10, 11], artificial neural networks (ANN) [12, 13] and particle swarm optimization (PSO) [14].

The mTSP is very time consuming due to its NP-hard nature. The mTSP has a multiplicity of applications mostly in the areas of routing and scheduling such as the School Bus Routing Problem [15, 16], and the Pickup and Delivery Problem [17, 18].

Many metaheuristic solution techniques have been developed for mTSP in recent years. Zhou and Li [19] introduced a modified GA to solve the mTSP problem by a modified GA. They utilized a greedy strategy to create the initial population, and the mutation operator to combine with the local search strategy 2−Opt, which allows one to quickly determine quality neighboring solutions and accelerates the convergence of the algorithm. Sedighpour et al. [20] presented an effective GA for solving the mTSP, in which the 2-Opt local search algorithm is used for improving solutions. Wacholder et. al. [12] proposed an effective neural network algorithm to solve the mTSP based on transforming the mTSP to the TSP. Y. Wang et al. [21] proposed a novel memetic algorithm for solving mTSP, which integrates with a sequential variable neighborhood descent that is a powerful local search procedure to exhaustively search the areas near the high-quality solutions. They also investigated the total distance traveled by all the salesmen when optimizing the minmax objective, and the results showed that in comparison with the six existing algorithms, the proposed algorithm had a better or at least competitive capacity to maintain the total distance as short as possible. C. Doppstadt et al. [22] proposed a heuristic solution approach, based mainly on a Tabu Search, to solve the Hybrid Electric Vehicle - Traveling Salesman Problem. The aim of this approach was The reduction in carbon dioxide levels by using hybrid electric vehicles. J. Kinable et al. [23] presented two integer programming models to solve the Equitable Traveling Salesman Problem (ETSP) problem and compare the strength of these formulations. They solved the first model through branch-and-cut, and the other model is solved through a branch-and-price framework. They used branch-and bound and branch-and-price for small and medium sized instances. However, for larger instances branch-and bound outperforms branch-and-price.

In this paper, a new exact solution technique for the mTSP is proposed. Also, mTSP is formulated using a constraint programming (CP) model and refer to this model as CP- mTSP. We use CP Optimizer for solving mTSP. CP has been shown to be an efficient solution technique for numerous combinatorial optimization problems.

The contributions of this paper are threefold. First, a CP model is introduced for mTSP and CP Optimizer is used for solving mTSP. Due to the strengths of CP in expressing complex relationships, very difficult constraints such as selective node visits, subtour elimination, etc. are represented. Compared with ILP formulations for mTSP, CP- mTSP does not require a large number of decision variables and constraints. Thus, we are able to run benchmark instances without experiencing any memory problems. When compared with metaheuristic approaches in the literature such as ACO, SW+

AS_{elite}, GELS-GA and Enhanced GA, our CP model does not require extensive parameter tuning as those methods do. And while the metaheuristic methods are quite efficient in finding good quality solutions, they are not able to prove the optimality of those solutions, as we are able for some instances using CP. Second, CP- mTSP and its components, such as global constraints, provide a strong base for other solution techniques for OP variants and related routing problems, potentially fostering new methodological developments. Third, the results we obtain using CP Optimizer with CP- mTSP advance current knowledge regarding TSPLIB benchmark instances in a number of ways.

The remainder of this paper is organized as follows. Section 2 provides the definition of the multiple Travelling Salesman Problem (mTSP). Section 3 provides the CP formulation for mTSP and provides an illustrative example. Section 4 provides results for CPLEX- mTSP and CP- mTSP on a set of benchmark instances from the TSPPLIB. Section 5 provides results for CP- mTSP and a comparison to existing algorithms from the literature. Finally, conclusions and future research directions are discussed in Section 6.

2 Problem Definition

A generalization of the well-known Travelling Salesman Problem is the standard multiple Travelling Salesman Problem (mTSP). The problem can be defined simply as the determination of a set of routes for m salesmen who all start from and return to a single home city.

Consider a complete directed graph $G = (V, A)$, where V is the set of nodes (vertices), A is the set of arcs and $C = (c_{ij})$ is the cost (distance) matrix associated with each arc $(i, j) \in A$. The cost matrix C can be symmetric, asymmetric or Euclidean. Let there be m salesmen located at the depot city 1. Then, the single depot mTSP consists of finding tours for m salesmen such that all start and end at the depot, each other node is located in exactly one tour, the number of nodes visited by a salesman lies within a predetermined interval, and the overall cost of visiting all nodes is minimized.

Let us define x_{ij} as a binary variable equal to 1 if arc (i, j) is in the optimal solution and 0 otherwise. For any salesman, u_i is the number of nodes visited on that salesman 's path from the origin up to node i (i.e., the visit number of the i^{th} node). L is the maximum number of nodes a salesman may visit; thus, $1 \le u_i \le L$; for all $i \ge 2$. In addition, let K be the minimum number of nodes a salesman must visit, i.e., if $x_{ij} = 1$, then $K \le u_i$ must be satisfied. The position variables u_i make it possible to avoid the classical subtour elimination constraints.

In [24] the following integer linear programming formulation for the mTSP proposed as follows:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\sum_{j=2}^{n} x_{1j} = m \tag{1}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{2}$$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \qquad j = 2,...,n \tag{3}$$

$$\sum_{j=1, i\neq j}^{n} x_{ij} = 1 \qquad i = 2,...,n \tag{4}$$

$$u_i + (L-2)x_{1i} - x_{i1} \leq L-1 \qquad i = 2,..,n \tag{5}$$

$$u_i + x_{1i} + (2-K)x_{i1} \geq 2 \qquad i = 2,..,n \tag{6}$$

$$x_{1i} + x_{i1} \leq 1 \qquad i = 2,..,n \tag{7}$$

$$u_i - u_j + Lx_{ij} + (L-2)x_{ji} \leq L-1 \qquad 2 \leq \mathrm{i} \neq \mathrm{j} \leq \mathrm{n} \tag{8}$$

$$x_{ij} \in \{0,1\}, \qquad \forall(\mathrm{i},\mathrm{j}) \in A$$

In this formulation, constraints (1) and (2) ensure that exactly m salesmen leave from and return to the depot. Constraints (3) and (4) are the degree constraints. The inequalities given in (5) and (6) serve as upper and lower bound constraints on the number of nodes visited by each salesman, and initialize the value of to 1 if and only if is the first node on the tour for any salesman. Inequality (7) forbids a vehicle from visiting only a single node. The inequalities given in (8) ensure that $u_j = u_i + 1$ if and only if $x_{ij} = 1.$

3 Constraint Programming

Constraint programming is a powerful paradigm for solving combinatorial search problems that draws on a wide range of techniques from artificial intelligence, computer science and operation research. A fundamental challenge in constraint programming is to understand the computational complexity of problems involving constraints. There are three differences between CP models and MILP models: firstly, CP models mostly concentrate on the constraints and feasibility. Secondly, In CP models, constraints can be logic constraints. And thirdly, CP models, for eliminating infeasible configurations

and decreasing the search space utilize heuristics, such as consistency techniques, constraint propagation, and branch and prune.

In general, CP models can be divided into two groups: first one is constraint satisfaction problems, and second one is constraint optimization problems that used in present research. An assignment of values from the variables domain which satisfies all the constraints is a solution to a constraint satisfaction problem. Otherwise, search procedures are used to find the best solution for constraint optimization problems. The readers are referred to [25-30] for reviewing of the solving techniques of the CP problems.

The multiple traveling salesman problem (mTSP) is a NP-hard combinatorial optimization problem. To address these computational challenges, we aim to test the effectiveness of CP, which is well known for its abilities to express complex relationships using global constraints and to obtain good quality solutions within reasonable times. A CP implementation contains a search strategy and a constraint propagation mechanism designed to filter out the values in (integer) variable domains that cause infeasible solutions[27, 31].

In the constraint model, algorithms are triggered every time a change occurs in the domain of a variable. A feasible solution is obtained when all domains are reduced to a single value. A variable can be used to model more than one constraint so whenever a change occurs in the domain of a shared variable, propagation algorithms of all global constraints are run to search for the possible domain reductions of other variables [29, 32, 33]. Search strategies may include both look back and look ahead procedures. As the search proceeds, filtering algorithms are re-run with the updated information to identify a feasible solution.

Milano and Wallace [34] presented Constraint Programming as a natural formalism for modelling problems, and as a flexible platform for solving them. They combined linear programming with propagation and novel and varied search techniques which can be easily expressed in CP.

In the case of CP Optimizer, the optimality proofs are provided by the Failure-Directed search[35] whereas good quality solutions are produced by the Large Neighborhood Search [36], these 2 techniques are combined in the CP Optimizer automatic search.

Failure-directed Search (FDS) assumes that there is no (better) solution or that such a solution is very hard to find. Therefore, instead of looking for solution(s), it focuses on a systematic exploration of the search space, first eliminating assignments that are most likely to fail. Self-Adapting Large Neighborhood Search approach [36] combines Large Neighborhood Search with a portfolio of neighborhoods and completion strategies together with Machine Learning techniques to converge on the most efficient neighborhoods and completion strategies for the problem being solved. Large Neighborhood Search (LNS) and FDS form the basis of the automatic search for scheduling problems in CP Optimizer, part of IBM ILOG CPLEX Optimization Studio.

Pesant et.al [37] presented a constraint logic programming model for the traveling salesman problem with time windows which yields an exact branch-and-bound optimization algorithm without any restrictive assumption on the time window. This algorithm unlike dynamic programming approaches does not rely on the degree of discretization applied to the data. The data-driven mechanism at its core more fully exploits pruning

rules developed in operations research by using them not only a priori but also dynamically during the search.
Caseau et. al. [38] presented a set of techniques which include a propagation scheme to avoid intermediate cycles (a global constraint), a branching scheme and a redundant constraint that can be used as bounding method. Also, they show that these techniques can solve problems twice larger than those solved previously with constraint programming tools.

Our CP model is denoted by using the Optimization Programming Language (OPL) and formulated using the IBM ILOG CP optimizer. The remainder of this part indicates how to make the CP model step-by-step.

3.1 Variables

In this section, we introduce a CP model for mTSP. In this CP model, two types of variables are needed. One is an interval variable. We utilize (time) interval variables that are capable of expressing several critical decisions such as start time, end time, duration and usage rate under one variable [39, 40]. Interval variables are useful in order to represent complex scheduling and routing activities especially when they are optional (i.e. a task may or may not be processed, a customer may or may not be visited, etc.). An important additional feature of interval variables is the fact that they can be optional, that is, one can decide not to consider them in the solution schedule Laborie and Rogerie [45] mention several advantages of interval variables. One is that the optionality is already modeled in the definition of the interval variable and there is no need for additional constraints in order to enforce this binary relationship, as the traditional integer decision variables require in scheduling problem formulations. Also function presenceOf (a) is a boolean function which indicates (a) is absent or present [41]. In this study, a city can be selected by a set of salesmen, therefore optional interval variables are used to model this case. We define x_{ij} as an interval variable which represents the salesman j to visit city i. The other type of variable needed to complete the model is the interval sequence variable. In our model, a sequence variable represents an assortment on the set of cities visited by a salesman. Since we have several salesmen, so we have several sequences. The size of the sequence depends on the number of cities visited by any salesman. We define l_j as an interval sequence variable which represents the j^{th} salesman.

OPL gives us specific facilities to express decision variables in a more compact way. Dependent variables are implemented as OPL variable expressions (dexpr) defined through equality constraints. We use two dependent variables in our model, described just below:

$$NR_j = \sum_{i=2}^{n} presenceOf\left(x_{ij}\right),\ j = 1,...,m$$

NR_i represents the total number of cities visited by a salesman.

3.2 Constraints

By considering the function of *presenceOf(a)*, we define following constraint to ensure that a city is visited only by one salesman:

$$\sum_{j=1}^{m} presenceOf\left(x_{ij}\right)=1, \quad i=2,...,n$$

The distance between two cities is calculated as:

```
tuple triplet { int p1; int p2; int d; };
 {triplet}  dist  =  {<p1,p2,ftoi(round(sqrt(((coord[1][p1])-
(coord[1][p2]))^2+(coord[2][p1]-coord[2][p2])^2)))> | p1,p2 in
city };
```

CP has some predefined constraints which show the flexibility of the CP model. One of them is *noOverlap* which is used to guarantee that there is no overlap between any two cities. We use this on the interval sequence variables as:

$$noOverlap\left(l_j, dist\right), \quad j=1,2,...,m$$

This constraint states that the sequence defines a chain of non-overlapping intervals (distances), where any intervals in the chain are constrained to end before the start of the next interval in the chain.

We use $K \le NR_i \le U$ to enforce the upper and lower bounds for the number of cities visited by one salesman. To ensure that all salesmen start from depot 0 and turn back to the depot 1(, add two constraints as:

$$first\left(l_j, x_{0j}\right), \quad j=1,2,...,m$$

$$last\left(l_j, x_{1j}\right), \quad j=1,2,...,m$$

3.3. Objective

The objective function, minimizing total cost (distance) of visiting all cities, formulated as below:

$$Minimize \quad \sum_{j=1}^{m} endOf\left(x_{1j}\right)=\sum_{j=1}^{m}\max\left(endOf\left(x_{ij}\right)\right)$$

Where $endOf\left(x_{1j}\right)$ represents the distance travelled by each salesman to the depot city 1. In other words, *endOf* is an integer expression used to access the end time of an interval. The objective function minimizes the sum of the end time of the routes. The CP code of the proposed method shown in figure 1 is as follows:

```
using CP;
int cities=76;
range city=0..cities;
int salesmen=5;
range sale=1..salesmen;
int U=20;
int K=1;
float coord[i in 1..2][j in city]=...;

tuple triplet { int p1; int p2; int d; };
 {triplet} dist = {<p1,p2,ftoi(round(sqrt(((coord[1][p1])-
(coord[1][p2]))^2+(coord[2][p1]-coord[2][p2])^2)))> | p1,p2 in
city } ;

dvar interval task[t in city][l in sale] optional(t>1);
dvar sequence lane[l in sale] in all (t in city) task [t][l];
dexpr int num[l in sale]=sum(t in city:t>1)(presen-
ceOf(task[t][l]));
dexpr float y= sum(l in sale) endOf(task[1][l]);
minimize y;
subject to{
c1:
forall (t in city: t>1)
    sum(l in sale)presenceOf(task [t][l])==1;
c2:
forall (l in sale){
    noOverlap(lane[l], dist);
    first(lane[l],task[0][l]);
    last (lane[l],task[1][l]); }
c3:
forall (l in sale)
  {
  num[l]<=U;
  num[l]>=K;
    }
```

Figure1. CP Code of the mTSP

4 Computational Results

This section presents the computational study and its results. At first, the heuristic algorithms compared mTSP are described. Next, solving the standard issues of the mTSP from the TSPLIB library by two methods (MILP and CP) has been introduced in the subsections 4.2 and 4.3, respectively.

4.1 Test Algorithms for Comparison

In this subsection, we introduced the heuristic algorithms compared with mTSP as follows.

M. Yousefikhoshbakht et al. [42] proposed a hybrid two-phase algorithm called SW+ AS_{elite}, for solving the MTSP which can be explained as the problem of designing collection of routes from one depot to a number of customers subject to side constraints. At the first stage, the MTSP is solved by the sweep algorithm, and at the second stage, the Elite Ant Colony Optimization (AS_{elite}) and 3-opt local search are used for improving solutions.

LTang et al. [13] presented the model, solution method, and system developed and implemented for hot rolling production scheduling. This project is part of a large-scale effort to upgrade production and operations management systems of major iron and steel companies in China. They propose a parallel strategy to model the scheduling problem and solve it using a new modified genetic algorithm (MGA).

P. Junjie and W. Dingwei[10] presented ant colony optimization(ACO) to solve multiple travelling salesman problem (MTSP). They showed that how the ant colony optimization (ACO) can be applied to the MTSP with ability constraint.

A. R. Hosseinabadi et al.[43] presented a new hybrid algorithm, called GELS-GA for solving the mTSP. The objective of presenting the proposed algorithm was to combine the public search capabilities of GA with local search of GELS algorithm and to create a stable algorithm, which can make reaching the global optimum largely possible.

F. Zhao et al. [19] introduced an improved genetic algorithm for the multiple traveling salesman problem. In this algorithm, a pheromone-based crossover operator is designed, and a local search procedure is used to act as the mutation operator the pheromone-based crossover can utilize both the heuristic information including edge lengths and adjacency relations, and pheromone to construct offspring.

4.2 MILP Results

The proposed model in section 2 was solved by using IBM-ILOG-CPLEX-Optimization Studio software on a HP laptop with the processor Intel ® Core i7-2630QM CPU @ 2GHz. Standard issues of the mTSP are also included in the implementation of the proposed model, which are derived from the TSPLIB library. If the variable $x_{ij} = 1$, it means the j^{th} city is placed immediately after the i^{th} city by the same salesman. The instance of the TSPLIB called burma14 is solved in the following. The variables x_{ij} that are equal to 1 and the corresponding salesman in the mTSP -14 example with 14 cities , 2 salesmen and $1 \le u_i \le 9$ are as follows:

$$Salesman\,1 = \{1 \to 11 \to 9 \to 10 \to 2 \to 1\}$$

$$Salesman\,2 = \{1 \to 14 \to 3 \to 4 \to 12 \to 5 \to 6 \to 7 \to 13 \to 8 \to 1\}$$

The value of the objective function (optimal solution) is obtained 32 within 1 second. Also, this model is applied and tested on several instances from TSP problems of TSPLIB including Pr76, Pr152, Pr226, Pr299, Pr439, Pr1002, mTSP-51, mTSP-100-II

and mTSP-150-II. For each instance, the number of customers, n, the number of salesman, m, and the max number of customers that a salesman can visit, L, is presented. The numerical results of the proposed method on these instances are presented in Tables 1.

Table 1. The Result of the proposed method in order to minimize the total travelled distance

problem			MILP Result			
Name	n	m	L	MILP objective	Iterations	Times
Pr76	76	5	20	170700	1879075	00:10:01:22
Pr152	152	5	40	201770	356626	00:10:10:05
Pr226	226	5	50	1736500	84580	00:10:03:11
Pr229	299	5	70	671510	56925	00:10:04:40
Pr439	439	5	100	1999900	29446	00:10:11:03
Pr1002	1002	5	220	--	--	--
mTSP-51	51	3	51	**454-opt**	1131	00:00:06:40
	51	5	51	**496-opt**	34754	00:00:10:10
	51	10	51	**643-opt**	22990	00:00:04:32
mTSP-100-II	100	3	100	28419	1573953	00:20:05:12
	100	5	100	25135	6580111	00:20:15:08
	100	10	100	29048	5738885	00:20:03:21
	100	20	100	43537	3501321	00:20:10:00
mTSP-150-II	150	3	150	27964	538442	00:20:01:08
	150	5	150	29244	654921	00:20:00:49
	150	10	150	36795	547833	00:20:01:55
	150	20	150	53621-opt	412523	00:20:03:16
	150	30	150	77875-opt	357803	00:20:08:07

4.3 Constraint Programming Results

The mTSP-51 example of TSPLIB with 51 cities and 3 salesmen was solved by using IBM-ILOG-CP-Optimization Studio software on an aforesaid laptop and the best solution after 120 seconds is obtained 456. The cities travelled with each salesman are presented in the Table 2.

Table 2. The Result of the MTSP-51 in order to minimize the total travelled distance

Salesman 1			**Salesman 2**		
Visited cities	Arrival time	Leave time	Visited cities	Arrival time	Leave time
0	0	0	0	0	0
2	12	12	27	8	8
16	21	21	51	16	16
50	27	27	46	18	18
9	33	33	12	25	25
30	41	41	47	31	31
34	48	48	17	40	40
21	57	57	37	45	45
29	64	64	44	52	52
20	74	74	15	58	58
35	81	81	45	65	65
36	87	87	33	72	72
3	99	99	39	86	86
28	108	108	10	96	96
31	115	115	49	104	104
8	124	124	5	112	112
26	131	131	38	119	119
7	142	142	11	126	126
23	148	148	32	132	132
43	161	161	1	138	138
24	173	173			
14	184	184	**Salesman 3**		
25	190	190	Visited cities	Arrival time	Leave time
13	203	203	0	0	0
41	212	212	22	7	7
40	224	224	1	14	14
19	235	235			
42	244	244			
4	260	260			
18	268	268			
6	283	283			
48	292	292			
1	304	304			

The computational results of the proposed method on several instances from TSP problems of TSPLIB are presented in Tables 3 and 4.

Table 3. The Result of the proposed method in order to minimize the total travelled distance

Problem				**CP Result**	
Name	**n**	**m**	**L**	**CP objective**	**Time(s)**
Pr76	**76**	**5**	**20**	156388	00:02:00:52
Pr152	**152**	**5**	**40**	155595	00:02:00:50
Pr226	**226**	**5**	**50**	165804	00:02:00:58
Pr299	**299**	**5**	**70**	82834	00:02:00:71
Pr439	**439**	**5**	**100**	198990	00:02:01:05
Pr1002	**1002**	**5**	**220**	241468	00:02:03:71

Table 4. The Result of the proposed method in order to minimize the total travelled distance

Problem	**mTSP-51**			**mTSP-100-I**			
m	**3**	**5**	**10**	**3**	**5**	**10**	**20**
CP Objective	456	485	608	23509	24714	32440	44611
Time(s)	00:02:00:11	00:02:00:41	00:02:00:52	00:02:00:42	00:02:00:46	00:02:00:44	00:02:00:53
Problem	**mTSP-100-II**				**mTSP-150-I**		
m	**3**	**5**	**10**	**20**	**3**	**5**	**10**
CP Objective	23023	23910	31556	40652	29687	31045	37605
Time(s)	00:02:00:42	00:02:00:33	00:02:00:04	00:02:00:51	00:02:00:46	00:02:00:49	00:02:00:52
Problem	**mTSP-150-I**		**mTSP-150-II**				
m	**20**	**30**	**3**	**5**	**10**	**20**	**30**
CP Objective	49917	63795	27327	29050	39675	56306	80018
Time(s)	00:02:00:66	00:02:00:83	00:02:00:47	00:02:00:44	00:02:00:49	00:02:00:63	00:02:00:82

5 Comparison of Computational Results

The implementation results of the proposed model are compared to the other existing optimization algorithm in the literature and articles [10, 13, 19, 42, 43].

A very important advantage of CP-mTSP compared to ILP model in Vansteenwegen et al. [44] and Labadie et al. [45] is that the number of constraints is no longer exponentially growing with the size of the input such as number of customers, vehicles and tour duration.

Table 5. Comparison of the MILP model and CP model with the result of papers **[10, 13, 42, 43]**

Problem	**Name**	**Pr76**	**Pr152**	**Pr226**	**Pr299**	**Pr439**	**Pr1002**
	n	**76**	**152**	**226**	**299**	**439**	**1002**
	m	**5**	**5**	**5**	**5**	**5**	**5**
	L	**20**	**40**	**50**	**70**	**100**	**220**
ACO[10]	**Avg.**	180690	136341	170877	83845	165035	387205
	Time(s)	51	128	143	288	563	2620
SW+ AS_{elite} [42]	**Avg.**	157562	128004	168156	82195	162657	381654
	Time(s)	19	41	62	65	95	186
GELS-GA[43]	**Avg.**	132784	105205	152135	76554	146523	354341
	Time(s)	5	8	9	11	16	27
MGA [13]	**Avg.**	160574	133337	178501	85796	183698	459179
	Time(s)	43	91	165	363	623	2892
MILP Result	**Obj. func.**	170700	201770	1736500	671510	1999900	--
	Time(s)	601	610	603	604	611	--
CP Result	Obj. func.	156388	155595	165804	82834	198990	241468
	Time(s)	120	120	120	120	120	120

According to Table 5, the MILP model lost its computational efficiency in large scale problem. In problems with 76 and 152 cities, MILP could not find optimal solution, as well as, in problems with 226, 299, 439 and 1002 cities, MILP could not find feasible solution in 600 seconds. The computational efficiency of the CP model is better than MILP model for Pr76, Pr152, Pr226, Pr299, Pr439, Pr1002. Also, the CP model is better than ACO and SW+ AS_{elite} algorithms for Pr76, Pr152 and Pr226. In addition, the CP model is better than ACO for Pr299.

Table 6. Comparison of the MILP model and CP model with the result of papers**[19, 43]**

Problem	**Name**	**mTSP-51**			**mTSP-100-II**			
	m	**3**	**5**	**10**	**3**	**5**	**10**	**20**
Enhanced GA [19]	**Best**	447.42	446.11	583.57	22366.57	23895.38	27675.42	39993.83
	Avg.	448.5	478.41	587.39	22466.41	24040.57	28033.53	40274.58
	Worst	449.62	482.41	589.86	22611.24	24095.96	28216.64	40582.55
	Time (s)	7.10	8.78	11.20	17.27	20.18	26.52	34.89
GELS-GA [43]	**Best**	252	259	324	17800	20532	24354	35142
	Avg.	254.5	263.5	328	18035	20589	24803.5	35969
	Worst	257	268	332	18270	20646	25253	36796
	Time(s)	5	4	9	12	14	17	23
MILP Result	**Obj. func.**	454-opt	496	643-opt	28419	25135	29048	43537
	Time (s)	6	10	4	1205	1215	1203	1210
CP Result	**Obj. func.**	456	485	608	23023	23910	31556	40652
	Time (s)	120	120	120	120	120	120	120

	Name	**mTSP-150-II**				
Problem	**m**	**3**	**5**	**10**	**20**	**30**
Enhanced GA [19]	**Best**	39179.41	40437.18	40437.18	55959.70	71605.25
	Avg.	39361.04	40663.31	40663.31	56417.86	71808.99
	Worst	39557.43	40803.15	40803.15	56572.87	71923.98
	Time (s)	28.04	34.02	34.02	57.13	67.47
GELS-GA [43]	**Best**	37600	38132	38132	51393	66474
	Avg.	37933.5	38336.5	38336.5	51443	66824.5
	Worst	38267	38541	38541	51493	67175
	Time (s)	25	28	28	36	44
MILP	Obj. func.	27964	29244	36795	53621-opt	77875-opt
	Time	1201	1200	1201	1203	1208

Result	(s)						
CP Result	Obj. func.	27327	29050	29050	56306	80018	
	Time (s)	120	120	120	120	120	

According to Table 6, the MILP is better than CP model only for mTSP-51 with 3 salesmen and lost its computational efficiency in large scale with 100 and 150 cities. The CP model is better than Enhanced GA and GELS-GA algorithms for mTSP-150-II with 3, 5 and 10 salesmen.

6 Conclusion

In this paper, we study the mTSP, formulate it as a constraint programming model and report solutions for benchmark instances of the TSPLIB. Constraint programming is a powerful paradigm for solving combinatorial search problems so we use the CP model for solving multiple Travelling Salesman Problem (mTSP). The mTSP is very time consuming due to its NP-hard nature. The new model, CP- mTSP, has been shown to be quite competitive with the heuristic algorithms and can be a reference method for solving variants of OP. The computational results indicate that CP- mTSP performs, on average, quite well compared to heuristic algorithms and outperforms most of the existing approaches in the literature in terms of the number of identified best-known solutions. TSP problems of TSPLIB including Pr76, Pr152, Pr226, Pr299, Pr439, Pr1002, mTSP-51, mTSP-100-II and mTSP-150-II show that the efficiency of the CP model for solving problem in large scale problem. Also, these results show that the CP model is more efficient than other algorithms such as ACO, SW+ AS_{elite}, GELS-GA and Enhanced GA in some instances of TSPLIB. Future work for consideration is related to the development of an efficient and effective decomposition algorithm.

ILP provides a global perspective through reporting upper and lower bounds that guide the search effectively. Also, CP is well known for its ability to find good quality feasible solutions for complex structured problems within reasonable time. Therefore, a hybrid decomposition algorithm that exploits these benefits of ILP and CP may obtain new best-known solutions and/or prove the optimality of the current ones for the benchmark problem instances.

References

1. Carter, A.E. and C.T. Ragsdale, *A new approach to solving the multiple traveling salesperson problem using genetic algorithms.* European journal of operational research, 2006. **175**(1): p. 246-257.
2. Yadlapalli, S., et al., *A Lagrangian-based algorithm for a multiple depot, multiple traveling salesmen problem.* Nonlinear Analysis: Real World Applications, 2009. **10**(4): p. 1990-1999.

3. Cordeau, J.-F., M. Dell'Amico, and M. Iori, *Branch-and-cut for the pickup and delivery traveling salesman problem with FIFO loading.* Computers & Operations Research, 2010. **37**(5): p. 970-980.
4. Balachandar, S.R. and K. Kannan, *Randomized gravitational emulation search algorithm for symmetric traveling salesman problem.* Applied Mathematics and Computation, 2007. **192**(2): p. 41.٤٢١-٣
5. Bianchi, L., J. Knowles, and N. Bowler, *Local search for the probabilistic traveling salesman problem: Correction to the 2-p-opt and 1-shift algorithms.* European Journal of Operational Research, 2005. **162**(1): p. 206-219.
6. Karapetyan, D. and G. Gutin, *Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem.* European Journal of Operational Research, 2011. **208**(3): p. 221-232.
7. Király, A. and J. Abonyi, *Optimization of multiple traveling salesmen problem by a novel representation based genetic algorithm*, in *Intelligent Computational Optimization in Engineering*. 2011, Springer. p. 241-269.
8. Yuan, S., et al., *A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms.* European Journal of Operational Research, 2013. **228**(1): p. 72-82.
9. Rostami, A.S., et al., *Solving multiple traveling salesman problem using the gravitational emulation local search algorithm.* Appl. Math, 2015. **9**(2): p. 699-709.
10. Junjie, P. and W. Dingwei. *An ant colony optimization algorithm for multiple travelling salesman problem*. in *Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on*. 2006. IEEE.
11. Liu, W., et al. *An ant colony optimization algorithm for the multiple traveling salesmen problem*. in *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*. 2009. IEEE.
12. Wacholder, E., J. Han, and R. Mann, *An extension of the Hopfield-Tank model for solution of the multiple Traveling Salesmen Problem*. 1988: Engineering Physics and Mathematics Division, Oak Ridge National Laboratory.
13. Tang, L., et al., *A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex.* European Journal of Operational Research, 2000. **124**(2): p. 267-282.
14. Trelea, I.C., *The particle swarm optimization algorithm: convergence analysis and parameter selection.* Information processing letters, 2003. **85**(6): p. 317-325.
15. Angel, R., et al., *Computer-assisted school bus scheduling.* Management Science, 1972. **18**(6): p. B-279-B-288.
16. Orloff, C., *Routing a fleet of M vehicles to/from a central facility.* Networks, 1974. **4**(2): p. 147-162.
17. Christofides, N. and S. Eilon, *An algorithm for the vehicle-dispatching problem.* Journal of the Operational Research Society, 1969. **20**(3): p. 309-318.
18. Wilhelm, W.E., *A technical review of column generation in integer programming.* Optimization and Engineering, 2001. **2**(2): p. 159-200.
19. Zhao, F., et al. *An improved genetic algorithm for the multiple traveling salesman problem*. in *Control and Decision Conference, 2008. CCDC 2008. Chinese*. 2008. IEEE.
20. Sedighpour, M., M. Yousefikhoshbakht, and N. Mahmoodi Darani, *An effective genetic algorithm for solving the multiple traveling salesman problem.* Journal of Optimization in Industrial Engineering, 2012(8): p. 73-79.

21. Wang, Y., Y. Chen, and Y. Lin, *Memetic algorithm based on sequential variable neighborhood descent for the minmax multiple traveling salesman problem.* Computers & Industrial Engineering, 2017. **106**: p. 105-122.
22. Doppstadt, C., A. Koberstein, and D. Vigo, *The Hybrid Electric Vehicle–Traveling Salesman Problem.* European Journal of Operational Research, 2016. **253**(3): p. 825-842.
23. Kinable, J., et al., *Exact algorithms for the equitable traveling salesman problem.* European Journal of Operational Research, 2017.
24. Kara, I. and T. Bektas, *Integer linear programming formulations of multiple salesman problems and its variations.* European Journal of Operational Research, 2006. **174**(3): p. 1449-1458.
25. Kumar, V., *Algorithms for constraint-satisfaction problems: A survey.* AI magazine, 1992. **13**(1): p. 32.
26. Barták, R., *On-line guide to constraint programming*. 1998.
27. Hooker, J.N., *Integrated methods for optimization*. Vol. 100. 2007: Springer Science & Business Media.
28. Heipcke, S., *Comparing constraint programming and mathematical programming approaches to discrete optimisation—the change problem.* Journal of the Operational Research Society, 1999. **50**(6): p. 581-595.
29. van Hoeve, W. and I. Katriel, *Handbook of Constraint Programming, ser. Foundations of Artificial Intelligence*. 2006, Elsevier Science.
30. Wallace, M. *Principles and Practice of Constraint Programming-CP 2004*. in *10th International Conference, CP*. 2004. Springer.
31. Rossi, F., P. Van Beek, and T. Walsh, *Handbook of constraint programming*. 2006: Elsevier.
32. Lombardi, M. and M. Milano, *Optimal methods for resource allocation and scheduling: a cross-disciplinary survey.* Constraints, 2012. :(١)١٧p. 51-85.
33. Harjunkoski, I. and I.E. Grossmann, *Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods.* Computers & Chemical Engineering, 2002. **26**(11): p. 1533-1552.
34. Milano, M. and M .Wallace, *Integrating operations research in constraint programming.* Annals of Operations Research, 2010. **175**(1): p. 37-76.
35. Vilím, P., P. Laborie, and P. Shaw. *Failure-directed search for constraint-based scheduling*. in *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*. 2015. Springer.
36. Laborie, P. and D. Godard, *Self-adapting large neighborhood search: Application to single-mode scheduling problems.* Proceedings MISTA-07, Paris, 2007 :p. 276-284.
37. Pesant, G., et al., *An exact constraint logic programming algorithm for the traveling salesman problem with time windows.* Transportation Science, 1998. **32**(1): p. 12-29.
38. Caseau, Y. and F. Laburthe. *Solving Small TSPs with Constraints*. in *ICLP*. 1997.
39. Studio, I.I.C.O., *V12. 3, Inc.,"Using the CPLEXR Callable Library and CPLEX Barrier and Mixed Integer Solver Options," 2011*.
40. Laborie, P. and J. Rogerie. *Reasoning with Conditional Time-Intervals*. in *FLAIRS conference*. 2008.
41. ILOG, I., *IBM ILOG CPLEX Optimization Studio, V12. 5*. 2013.
42. Yousefikhoshbakht, M. and M. Sedighpour, *A combination of sweep algorithm and elite ant colony optimization for solving the multiple traveling salesman problem.* Proceedings of the Romanian academy A, 2012. **13**(4): p. 295-302.
43. Hosseinabadi, A.A., et al. *Gels-ga: hybrid metaheuristic algorithm for solving multiple travelling salesman problem*. in *Intelligent Systems Design and Applications (ISDA), 2014 14th International Conference on*. 2014. IEEE.

44. Vansteenwegen, P., et al., *Iterated local search for the team orienteering problem with time windows.* Computers & Operations Research, 2009. **36**(12): p. 3281-3290.
45. Labadie, N., et al., *The team orienteering problem with time windows: An lp-based granular variable neighborhood search.* European Journal of Operational Research, 2012. **220**(1): p. 15-27.