Optimizing the Route of an Assembly Arm

Hajieh Jabbari K.

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Industrial Engineering

Eastern Mediterranean University
January 2011
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director (a)

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Industrial Engineering.

Asst. Prof. Dr. Gokhan Izbirak
Chair, Department of Industrial Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Industrial Engineering.

Prof. Dr. Bela Vizvari
Supervisor

ExaminingCommittee

1. Prof. Dr. Bela Vizvari
2. Prof. Dr. Alagar Rangan
3. Asst. Prof. Dr. Gokhan Izbirak

ABSTRACT

Optimizing the route of an assembly arm is a procedure of finding placement tours of pick and place robot's arm for the equipping of Printed Circuit Board (PCB). The problem of finding placement tours is a production planning problem with *n* positions on a board as the assembly points and *n* bins containing *n* components as *n* locations for the bins, called cell points. PCB manufacturing requires a good route for the robot that makes production, so that time savings can be achieved. In the robots considered, working time of the robot is proportional to the distance travelled, and the problem appears as a combination of the Traveling Salesman Problem (TSP) and the matching problem. Such a problem is a special type of the TSP, known as the bipartite TSP. Given the complete graph $G = (V, E)$ on $2n$ vertices, a weight function $W: E \rightarrow \mathbb{R} \geq 0$, and a partition of V into 2 subsets of size n, bipartite TSP is to find a Hamiltonian cycle of minimum weight that visits the subsets in a fixed alternating order. The problem has simulated many efforts to find an efficient algorithm but no algorithm is presently available that can solve for the optimal solution of this problem in polynomial time. As its complexity is NP-Complete the general opinion of scientists is that a fast polynomial algorithm does not exist.

The aim of this thesis is to introduce an efficient approximation algorithm for medium-sized (up to 500 assembly points) problems and to derive bounds for the typical length of optimal tours. We present an iterative algorithm which applies a cutting model to get a shorter lower bound by adding cuts to the Linear Programming

(LP) relaxations and a combined heuristic algorithm for finding an acceptable upper bound when the optimal integer solution is not found. The method is applied for both Dantzig-Fulkerson-Johnson and Miller-Tucker-Zemlin models. As the problem is NP-Complete, it is often unnecessary to have an exact solution. Thus a special heuristic algorithm is developed to obtain near-optimal solution in a reasonable time, suitable for practical purposes. The developed heuristic method is applied a constructive scheme combining two famous efficient heuristics: Nearest Neighbor and Insertion algorithms.

Keywords: TSP, Bipartite Graph, Pick and Place Robot, Heuristic Algorithms, Minimal Cut.

ÖZ

Bir montaj kolu rota optimizasyonu seçme ve yerleştirme robot kolunun yerleştirme turlarını bulma işlemidir ki Baskılı Devrenin onatılması için kullanılır. Yerleştirme turları bulma sorunu bir üretim planlama sorunudur. Bu problemlerde montaj noktaları için n tane pozisyon vardır ve her birisinin içerisinde bir tane birleşen parçası yerleştirilmiş n tane kutu vardır ki bunlara hücre denilir. Baskılı devre üretimi zaman tasarufu elde edebilir böylece üretim yapan robot için iyi bir rota bulunması gerekir. Ele alınan robotlarda çalışma süresi gezilen mesafe ile orantılıdır. Bu tip problemler Gezgin Satıcı Problemi ve Eşleştirme Probleminin birleşimi olarak görülür. Böyle bir problem özel gezgin satıcı problemidir ki ikili gezgin satıcı problemi olarak bilinir. Verilen tamamlanmış grafikde G=(V,E), 2n köşe noktası ve ağırlık fonksiyonu $W:E \rightarrow R \geq 0$ ve V iki n taneli alt kümeye bölünen ikili gezgin satici problemi minumum agırlıklı Hamilton çevrimi bulmakdır ki bir alt kümeyi sabit bir alternatif sırayla ziyaret eder. Etkin bir algorithma bulmak için çok çaba harcanmış ama halen bu sorunun optimal çözümü bulmak için polinom zamanda bir algoritma bulunmamıştır. Bilim adamlarının genel görüşüne göre hızlı bir polinom algoritma yoktur çünkü bu problem bir polinom zamanlı olmayan-tam problemidir.

Bu tezin amacı orta büyüklükteki soruları (500 montaj noktasına kadar) için etkin bir yaklaşım algoritması tanıtmaktır. En uygun turların uzunluğunun sınırlarını türetir. Biz bir yenilenen algoritma sunuyoruz ki bir kesim modeli ve birleştirilmiş sezgisel algoritmadan oluşur. Optimal tam sayı çözüm bulunmadığında kesim modelini daha yakın bir alt sınırı ve sezgisel algoritmayı kabul edilebilir bir üst sınırı bulmak için kullanıyoruz. Bu yöntem Dantzig-Fulkerson-Johnson ve Miller-Tucker-Zemlin modelleri için uygulanmıştır. Problem NP-Tam olduğu için çoğu kez kesin bir çözüm olması gereksizdir. Bu nedenle özel bir sezgisel algoritma geliştirilmiştir ki neredeyse optimal çözümü makul bir süre içerisinde ve pratik amaçlar için uygun olan cevabı elde ediyor. Geliştirilmiş sezgisel yöntem iki tane ünlü etkin yapısal sezgiselin birleşimidir. Bahsedilen iki sezgisel algoritma en yakın komşu ve yerleştirme algoritmalarıdır.

Anahtar Kelimeler: Gezgin Satici Problemi, Ikili Grafik, Ceçme ve Yerleştirme Robotu, Sezgisel Algoritmalar, Minimal Kesme

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF PROGRAMS

To My Love:

Ehsan

ACKNOWLEDGMENT

It is a pleasure to thank the many people who made this thesis possible.

In the first place I would like to record my gratitude to my supervisor, Prof. Dr. Bela Vizvari. With his enthusiasm, his inspiration, and his great efforts to explain things clearly and simply, he helped to make mathematics especially graph theory fun for me. Above all and the most needed, he provided me unflinching encouragement and support in various ways. His truly scientist intuition has made him as a constant oasis of ideas and passions in science, which exceptionally inspire and enrich my growth as a student, a researcher and an engineer want to be. Throughout two years that I was studying my master degree, he provided good teaching and lots of good ideas. I am indebted to him more than he knows.

I gratefully acknowledge Moosa Moghimi Hadji for his advice, and crucial contribution, which made him a backbone of this research and so to this thesis. Moosa, I am grateful in every possible way.

I would like to place on record, my sincere thanks to Prof. Dr. Alagar Rangan for his constructive comments and careful evaluation of the thesis.

I am deeply indebted to Ass. Prof. Dr. Gokhan Izbirak the chairman of the industrial engineering department for providing a loving environment to learn and grow.

It is impossible to overstate the influence of the Industrial Engineering department members in EMU University. I wish to thank them for assisting me in many different ways. Professors, instructors, research assistants, secretor and librarians deserve special mention.

I owe quit a lot to my family who encouraged me all throughout my studies. I would like to thank them because they raised me, supported me, taught me, and loved me.

Words fail me to express my appreciation to my husband Ehsan whose dedication, love and persistent confidence in me, has taken the load off my shoulders. I owe him for being unselfishly let his intelligence, passions, and ambitions collide with mine. Therefore, I would also thank Moghimi Hadji family for letting me take his hand in marriage, and accepting me as a member of the family, warmly.

Finally, I would like to thank everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

APPENDICES

Appendix A: Collected Data

Table A_1: Some Famous Pick and Place Machines and Their Specifications

MFCTR	**Product Name**	**Specifications**
APS Novastar	APS Novastar L60	Max board size: 343*813 mm; Max placement rate: 4800cph; Dispense option: 10,000 dots per hr.
Fuji	CP642 ME	Max board size: 457*356 mm; Max placement rate: 40,000cph; Placement accuracy: 0.1 mm; two feeder carriages
Fuji	QP 242E	Max board size: 457*356 mm; Max placement rate: 14,000cph; Placement accuracy: 0.1 mm; Modular multi-purpose machine
Fuji	IP 1	Max placing points: 999 sequences/program; Max placement rate: 1.5 sec/part; Placement accuracy: 0.1 mm
Hitachi	GXH-1	Max component size: 44*44 mm; Max placement rate: 60,000cph; 200 feeder positions 8 mm
Mydata	Mydata TP11 UFP	Max component size: 51.9*51.9*15 mm; Max picking rate: 6,000cph; 128 feeder positions 8 mm; Pick up nozzles 7
PMJ	HiSAC 1000	Odd form placement system; Pick & place travel: 450*870*150mm
Siemens	Siplace 80 F5 HM	With 12 nozzle collect and place plus pick and place head or 6 nozzle; Max placement rate: pick & place (1,800 cph); Placement accuracy: 38 micron 3 Sigma (p & p head)
Siemens	Siplace CF	With Compact 6 nozzle collect and place plus pick & place head; Max placement rate: pick & place (1,800 cph); Placement accuracy: 40 micron 4 Sigma (p & p head)
Siemens	Siplace 80 S27 HM	With 12 nozzle collect and place plus pick and place head or optional 6 nozzle; Max placement rate: 12 nozzle (26,500cph); Placement accuracy: 90 micron 4 Sigma (12 nozzle head)
Universal	Universal GSM II	X, Y Accuracy: 0.0381 mm; Rotational accuracy: 0.06 degree; Pick & place travel: 727.46*720.73*762.00 mm; Max. board: 508*457 mm

- cph is the abbreviation of chip per hour

Table A_2: Distance Matrix of Problem 6-City

		1	2	3	4	5	6
		ABADAN	ASTARA	ARAK	ARDABIL	URMIA	ISFAHAN
1	ABADAN	0	1351	704	1401	1192	868
2	ASTARA	1351	0	766	77	604	953
3	ARAK	704	766	0	843	786	288
4	ARDABIL	1401	77	834	0	527	1030
5	URMIA	1192	604	786	527	0	1074
6	ISFAHAN	868	953	288	1030	1074	0

Table A_3: Distance Matrix of Problem 10-City

	ABADAN	ASTARA	ARAK	ARDABIL	URMIA	ISFAHAN	AHVAZ	BABOL	BIRJAND	TABRIZ
ABADAN	0	1351	704	1401	1192	868	123	1226	1889	1198
ASTARA	1351	0	766	77	604	953	1228	515	1737	296
ARAK	704	766	0	843	786	288	581	522	1606	785
ARDABIL	1401	77	834	0	527	1030	1305	592	1814	219
URMIA	1192	604	786	527	0	1074	1064	1136	2220	308
ISFAHAN	868	953	288	1030	1074	0	745	668	1173	1038
AHVAZ	123	1228	581	1305	1064	745	0	1103	1918	1075
BABOL	1226	515	522	592	1136	668	1103	0	1222	828
BIRJAND	1889	1737	1606	1814	2220	1173	1918	1222	0	1912
TABRIZ	1198	296	785	219	308	1038	1075	828	1912	0

Table A_6: Input Data for Problem 80-1

Point	X	Y	Point	X	Y
1	100	100	41	388	378
2	100	200	42	328	1018
3	100	300	43	1518	1281
4	100	400	44	299	251
5	100	500	45	1390	1833
6	100	600	46	1052	1510
7	100	700	47	246	1878
8	100	800	48	1548	851
9	100	900	49	671	1441
10	100	1000	50	1218	1666
11	100	1100	51	442	1728
12	100	1200	52	1758	453
13	100	1300	53	1673	1064
14	100	1400	54	634	1390
15	100	1500	55	769	1640
16	100	1600	56	1189	1173
17	100	1700	57	592	1865
18	100	1800	58	1922	154
19	100	1900	59	1017	1084
20	100	2000	60	1897	979
21	2200	100	61	939	1654
22	2200	200	62	409	1779
23	2200	300	63	1559	1981
24	2200	400	64	412	324
25	2200	500	65	772	1654
26	2200	600	66	185	269
27	2200	700	67	166	1301
28	2200	800	68	1022	241
29	2200	900	69	532	243
30	2200	1000	70	1	14
31	2200	1100	71	772	655
32	2200	1200	72	73	107
33	2200	1300	73	1580	103
34	2200	1400	74	255	303
35	2200	1500	75	1910	949
36	2200	1600	76	543	1976
37	2200	1700	77	1654	816
38	2200	1800	78	1911	673
39	2200	1900	79	1239	482
40	2200	2000	80	1702	1559

Table A_7: Input Data for Problem 80-2

Point	X	Y	Point	X	Y
1	100	100	41	276	895
2	100	200	42	1303	1754
3	100	300	43	1005	1992
4	100	400	44	1247	186
5	100	500	45	24	839
6	100	600	46	1559	1970
7	100	700	47	72	781
8	100	800	48	608	466
9	100	900	49	1478	1766
10	100	1000	50	330	834
11	100	1100	51	1249	1675
12	100	1200	52	1647	1358
13	100	1300	53	931	250
14	100	1400	54	608	1074
15	100	1500	55	246	1398
16	100	1600	56	529	323
17	100	1700	57	1876	177
18	100	1800	58	1035	390
19	100	1900	59	1660	1568
20	100	2000	60	1698	1715
21	2200	100	61	1544	1510
22	2200	200	62	1409	615
23	2200	300	63	602	1137
24	2200	400	64	701	1982
25	2200	500	65	730	595
26	2200	600	66	1633	668
27	2200	700	67	1453	1092
28	2200	800	68	713	1287
29	2200	900	69	691	958
30	2200	1000	70	844	1205
31	2200	1100	71	185	1477
32	2200	1200	72	1031	1136
33	2200	1300	73	280	150
34	2200	1400	74	131	1056
35	2200	1500	75	948	1919
36	2200	1600	76	79	774
37	2200	1700	77	1016	26
38	2200	1800	78	1713	78
39	2200	1900	79	1984	1289
40	2200	2000	80	759	1753

Table A_8: Input Data for Problem 100-1

Point	X	Y	Point	X	Y
1	100	100	51	530	2297
2	100	200	52	2157	2019
3	100	300	53	1926	2416
4	100	400	54	231	1387
5	100	500	55	640	1773
6	100	600	56	1105	2358
7	100	700	57	2304	1764
8	100	800	58	1943	150
9	100	900	59	461	592
10	100	1000	60	1645	1702
11	100	1100	61	462	1863
12	100	1200	62	1186	1049
13	100	1300	63	104	2238
14	100	1400	64	304	1929
15	100	1500	65	293	569
16	100	1600	66	1925	1585
17	100	1700	67	2427	687
18	100	1800	68	1555	2302
19	100	1900	69	440	2396
20	100	2000	70	2298	1002
21	100	2100	71	1203	585
22	100	2200	72	479	1131
23	100	2300	73	1441	2509
24	100	2400	74	255	1666
25	100	2500	75	616	2109
26	2700	100	76	1113	1855
27	2700	200	77	2416	1767
28	2700	300	78	419	895
29	2700	400	79	382	2516
30	2700	500	80	621	749
31	2700	600	81	136	1219
32	2700	700	82	721	1907
33	2700	800	83	521	2097
34	2700	900	84	1152	353
35	2700	1000	85	1744	528
36	2700	1100	86	2050	2367
37	2700	1200	87	728	2091
38	2700	1300	88	2270	2545
39	2700	1400	89	443	1143
40	2700	1500	90	2589	703
41	2700	1600	91	641	494
42	2700	1700	92	1885	451
43	2700	1800	93	967	2543
44	2700	1900	94	2067	1545
45	2700	2000	95	2016	1379
46	2700	2100	96	516	1330
47	2700	2200	97	103	1187
48	2700	2300	98	1896	399

49	2700	2400	99	2290	832
50	2700	2500	100	2040	633

Table A_9: Input Data for Problem 100-2

Point	X	Y	Point	X	Y
1	100	100	51	648	558
2	100	200	52	2112	1091
3	100	300	53	901	804
4	100	400	54	961	1395
5	100	500	55	1114	128
6	100	600	56	437	1357
7	100	700	57	1304	1859
8	100	800	58	2228	526
9	100	900	59	1547	1455
10	100	1000	60	1071	616
11	100	1100	61	1881	842
12	100	1200	62	471	1202
13	100	1300	63	1533	2381
14	100	1400	64	140	854
15	100	1500	65	296	883
16	100	1600	66	522	440
17	100	1700	67	902	1039
18	100	1800	68	274	2485
19	100	1900	69	263	2390
20	100	2000	70	906	1938
21	100	2100	71	329	420
22	100	2200	72	1977	2003
23	100	2300	73	1435	136
24	100	2400	74	2386	1087
25	100	2500	75	1186	301
26	2700	100	76	1295	515
27	2700	200	77	2044	2537
28	2700	300	78	1738	542
29	2700	400	79	2020	330
30	2700	500	80	439	422
31	2700	600	81	210	1217
32	2700	700	82	2077	1446
33	2700	800	83	1459	1158
34	2700	900	84	1839	1403
35	2700	1000	85	2065	1823
36	2700	1100	86	2102	829
37	2700	1200	87	2137	2210
38	2700	1300	88	1092	827
39	2700	1400	89	1723	935
40	2700	1500	90	2215	948
41	2700	1600	91	1604	2529
42	2700	1700	92	1297	190
43	2700	1800	93	2414	1651
44	2700	1900	94	963	1535
45	2700	2000	95	1333	413
46	2700	2100	96	2596	528
47	2700	2200	97	528	1649
48	2700	2300	98	2360	1466
49	2700	2400	99	1411	929
50	2700	2500	100	698	2393

Point	X	Y	Point	X	Y	Point	X	Y	Point	X	Y
1	100	200	41	1700	200	81	134	1837	121	330	1111
2	100	300	42	1700	300	82	1512	867	122	499	386
3	100	400	43	1700	400	83	832	1883	123	1422	1219
4	100	500	44	1700	500	84	530	2534	124	980	1922
5	100	600	45	1700	600	85	1186	2540	125	753	1003
6	100	700	46	1700	700	86	1557	523	126	1030	1366
7	100	800	47	1700	800	87	1050	1783	127	293	1290
8	100	900	48	1700	900	88	749	848	128	1281	1722
9	100	1000	49	1700	1000	89	970	385	129	1269	2315
10	100	1100	50	1700	1100	90	249	324	130	1298	2362
11	100	1200	51	1700	1200	91	1563	869	131	1450	2517
12	100	1300	52	1700	1300	92	559	1925	132	1596	1423
13	100	1400	53	1700	1400	93	166	272	133	881	778
14	100	1500	54	1700	1500	94	1551	1071	134	1453	2238
15	100	1600	55	1700	1600	95	868	993	135	559	1489
16	100	1700	56	1700	1700	96	924	734	136	140	1896
17	100	1800	57	1700	1800	97	1218	566	137	533	2328
18	100	1900	58	1700	1900	98	416	1835	138	1161	432
19	100	2000	59	1700	2000	99	1275	830	139	804	530
20	100	2100	60	1700	2100	100	292	836	140	1510	786
21	100	2200	61	1700	2200	101	340	1191	141	1067	1271
22	100	2300	62	1700	2300	102	1243	1863	142	1199	1722
23	100	2400	63	1700	2400	103	353	548	143	1141	770
24	100	2500	64	1700	2500	104	304	517	144	322	1329
25	100	2600	65	1700	2600	105	143	1520	145	664	2551
26	200	100	66	200	2700	106	1241	630	146	1367	699
27	300	100	67	300	2700	107	1254	1849	147	1434	2293
28	400	100	68	400	2700	108	133	445	148	169	679
29	500	100	69	500	2700	109	841	1094	149	948	1120
30	600	100	70	600	2700	110	1540	198	150	1367	265
31	700	100	71	700	2700	111	115	1382	151	1598	616
32	800	100	72	800	2700	112	1252	835	152	1304	609
33	900	100	73	900	2700	113	933	1850	153	154	2140
34	1000	100	74	1000	2700	114	1116	1685	154	827	252
35	1100	100	75	1100	2700	115	1432	2117	155	1138	1154
36	1200	100	76	1200	2700	116	467	1814	156	136	1991
37	1300	100	77	1300	2700	117	434	1177	157	1045	1904

Point	X	Y	Point	X	Y	Point	X	Y	Point	X	Y
38	1400	100	78	1400	2700	118	1289	2131	158	1568	1569
39	1500	100	79	1500	2700	119	1506	2339	159	1396	2209
40	1[illegible]00	100	80	1600	2700	120	1107	1846	160	1381	1800

Table A_{10}: Input Data for Problem 25-15-1

Table A_{11}: Input Data for Problem 25-15-2

1	100	200	41	1700	200	81	811	1353	121	625	1500
2	100	300	42	1700	300	82	682	507	122	423	714
3	100	400	43	1700	400	83	696	1141	123	719	653
4	100	500	44	1700	500	84	427	2534	124	1179	653
5	100	600	45	1700	600	85	572	655	125	313	594
6	100	700	46	1700	700	86	746	2331	126	963	562
7	100	800	47	1700	800	87	1170	1845	127	1066	1556
8	100	900	48	1700	900	88	497	297	128	741	2104
9	100	1000	49	1700	1000	89	748	1018	129	133	228
10	100	1100	50	1700	1100	90	1110	1036	130	1485	1154
11	100	1200	51	1700	1200	91	602	2204	131	1245	926
12	100	1300	52	1700	1300	92	834	1719	132	492	401
13	100	1400	53	1700	1400	93	1130	1083	133	1256	2587
14	100	1500	54	1700	1500	94	1514	212	134	687	1673
15	100	1600	55	1700	1600	95	936	1349	135	915	2345
16	100	1700	56	1700	1700	96	805	1081	136	763	174
17	100	1800	57	1700	1800	97	1322	1230	137	374	861
18	100	1900	58	1700	1900	98	545	1796	138	621	1196
19	100	2000	59	1700	2000	99	929	2057	139	999	1198
20	100	2100	60	1700	2100	100	419	2495	140	1263	1238
21	100	2200	61	1700	2200	101	498	911	141	1376	2463
22	100	2300	62	1700	2300	102	162	2150	142	274	140
23	100	2400	63	1700	2400	103	909	2196	143	1051	1420
24	100	2500	64	1700	2500	104	876	1998	144	1170	226
25	100	2600	65	1700	2600	105	687	2479	145	1396	956
26	200	100	66	200	2700	106	275	886	146	1259	257
27	300	100	67	300	2700	107	338	157	147	722	1994
28	400	100	68	400	2700	108	1143	1084	148	823	727
29	500	100	69	500	2700	109	961	1468	149	320	830
30	600	100	70	600	2700	110	1455	2203	150	1186	860
31	700	100	71	700	2700	111	254	674	151	1547	1959
32	800	100	72	800	2700	112	1060	387	152	569	1630
33	900	100	73	900	2700	113	656	1203	153	402	201
34	1000	100	74	1000	2700	114	1309	1261	154	1557	1087
35	1100	100	75	1100	2700	115	648	1995	155	458	422
36	1200	100	76	1200	2700	116	1270	2032	156	474	2252
37	1300	100	77	1300	2700	117	423	1082	157	418	1613
38	1400	100	78	1400	2700	118	283	958	158	1365	2510
39	1500	100	79	1500	2700	119	556	1700	159	570	2156
40	1600	100	80	1600	2700	120	1012	1514	160	711	1819

Table A_{12}: Input Data for Problem 25-15-3

Point	X	Y	Point	X	Y	Point	X	Y	Point	X	Y
1	100	200	41	1700	200	81	1319	633	121	567	1297

2	100	300	42	1700	300	82	1566	2308	122	520	588
3	100	400	43	1700	400	83	761	444	123	1272	942
4	100	500	44	1700	500	84	1349	675	124	1003	231
5	100	600	45	1700	600	85	1564	1654	125	321	428
6	100	700	46	1700	700	86	184	1154	126	464	2365
7	100	800	47	1700	800	87	775	496	127	949	1486
8	100	900	48	1700	900	88	1529	1612	128	888	218
9	100	1000	49	1700	1000	89	836	834	129	696	1008
10	100	1100	50	1700	1100	90	1038	607	130	438	820
11	100	1200	51	1700	1200	91	571	1728	131	834	143
12	100	1300	52	1700	1300	92	1189	565	132	1165	2086
13	100	1400	53	1700	1400	93	1552	1752	133	797	288
14	100	1500	54	1700	1500	94	1029	1543	134	884	659
15	100	1600	55	1700	1600	95	142	278	135	791	2213
16	100	1700	56	1700	1700	96	1297	2213	136	826	1325
17	100	1800	57	1700	1800	97	1163	2247	137	158	878
18	100	1900	58	1700	1900	98	1255	1696	138	912	1434
19	100	2000	59	1700	2000	99	1360	1663	139	655	1565
20	100	2100	60	1700	2100	100	1086	711	140	965	2559
21	100	2200	61	1700	2200	101	584	698	141	250	2376
22	100	2300	62	1700	2300	102	525	389	142	1225	369
23	100	2400	63	1700	2400	103	706	873	143	1250	867
24	100	2500	64	1700	2500	104	202	1216	144	304	658
25	100	2600	65	1700	2600	105	1130	396	145	280	2492
26	200	100	66	200	2700	106	877	1971	146	1002	1903
27	300	100	67	300	2700	107	765	1390	147	118	183
28	400	100	68	400	2700	108	780	1597	148	827	1688
29	500	100	69	500	2700	109	1528	313	149	1285	1861
30	600	100	70	600	2700	110	640	1585	150	359	228
31	700	100	71	700	2700	111	1270	215	151	352	684
32	800	100	72	800	2700	112	735	726	152	581	1250
33	900	100	73	900	2700	113	1028	2522	153	377	2317
34	1000	100	74	1000	2700	114	1321	1524	154	1321	2057
35	1100	100	75	1100	2700	115	974	2279	155	1574	1313
36	1200	100	76	1200	2700	116	261	395	156	220	1568
37	1300	100	77	1300	2700	117	651	2151	157	1153	334
38	1400	100	78	1400	2700	118	1111	1444	158	900	2232
39	1500	100	79	1500	2700	119	1169	1933	159	437	1335
40	1600	100	80	1600	2700	120	503	2193	160	246	167

Table A_{13}: Input Data for Problem 200-1

Point	X	Y	Point	X	Y	Point	X	Y	Point	X	Y
1	100	100	51	200	100	101	1298	2196	151	1032	231

2	100	200	52	300	100	102	662	2443	152	1862	884
3	100	300	53	400	100	103	842	184	153	2529	1112
4	100	400	54	500	100	104	2025	1195	154	1011	1306
5	100	500	55	600	100	105	1198	989	155	455	463
6	100	600	56	700	100	106	320	1266	156	302	1414
7	100	700	57	800	100	107	634	189	157	152	1620
8	100	800	58	900	100	108	126	2576	158	1035	1091
9	100	900	59	1000	100	109	689	442	159	2236	706
10	100	1000	60	1100	100	110	1960	2396	160	1451	2545
11	100	1100	61	1200	100	111	2114	2339	161	1453	315
12	100	1200	62	1300	100	112	2531	2124	162	2080	1075
13	100	1300	63	1400	100	113	814	2197	163	340	1081
14	100	1400	64	1500	100	114	373	794	164	2230	2093
15	100	1500	65	1600	100	115	319	1792	165	917	1928
16	100	1600	66	1700	100	116	1878	1974	166	351	1763
17	100	1700	67	1800	100	117	1740	654	167	1749	1232
18	100	1800	68	1900	100	118	1434	523	168	1510	2358
19	100	1900	69	2000	100	119	548	243	169	501	2021
20	100	2000	70	2100	100	120	1216	443	170	580	823
21	100	2100	71	2200	100	121	1933	983	171	2180	1431
22	100	2200	72	2300	100	122	853	785	172	692	410
23	100	2300	73	2400	100	123	2405	629	173	233	595
24	100	2400	74	2500	100	124	2555	1596	174	788	1497
25	100	2500	75	2600	100	125	1120	476	175	1651	321
26	2700	100	76	200	2700	126	1138	779	176	1997	2335
27	2700	200	77	300	2700	127	1794	1590	177	1505	2255
28	2700	300	78	400	2700	128	902	1494	178	1382	478
29	2700	400	79	500	2700	129	208	522	179	1941	2120
30	2700	500	80	600	2700	130	1421	2335	180	950	811
31	2700	600	81	700	2700	131	523	2439	181	1920	484
32	2700	700	82	800	2700	132	1437	535	182	664	2011
33	2700	800	83	900	2700	133	2460	2598	183	1990	531
34	2700	900	84	1000	2700	134	1028	2188	184	1375	1463
35	2700	1000	85	1100	2700	135	2042	1188	185	1278	1721
36	2700	1100	86	1200	2700	136	2441	1359	186	478	1716
37	2700	1200	87	1300	2700	137	146	261	187	1399	1075
38	2700	1300	88	1400	2700	138	1350	2476	188	812	1255
39	2700	1400	89	1500	2700	139	2300	1771	189	1776	1414
40	2700	1500	90	1600	2700	140	2271	974	190	1585	134
41	2700	1600	91	1700	2700	141	2147	1265	191	847	597
42	2700	1700	92	1800	2700	142	1305	2518	192	2455	1404
43	2700	1800	93	1900	2700	143	1160	2359	193	2479	2183
44	2700	1900	94	2000	2700	144	238	204	194	1060	1497
45	2700	2000	95	2100	2700	145	880	874	195	1114	1936
46	2700	2100	96	2200	2700	146	1744	756	196	1514	916
47	2700	2200	97	2300	2700	147	2506	159	197	303	1367
48	2700	2300	98	2400	2700	148	2275	306	198	2217	2039
49	2700	2400	99	2500	2700	149	2509	1560	199	2585	1589
50	2700	2500	100	2600	2700	150	1982	1528	200	536	2207

Table A_{14}: Input Data for Problem 200-2

Point	X	Y	Point	X	Y	Point	X	Y	Point	X	Y
1	100	100	51	200	100	101	2068	1257	151	1707	2243

2	100	200	52	300	100	102	1742	1570	152	2134	1548
3	100	300	53	400	100	103	1963	832	153	1981	2164
4	100	400	54	500	100	104	938	2488	154	1548	1520
5	100	500	55	600	100	105	1422	1794	155	1047	514
6	100	600	56	700	100	106	415	1225	156	169	912
7	100	700	57	800	100	107	2279	1875	157	1010	826
8	100	800	58	900	100	108	2066	2171	158	2492	1952
9	100	900	59	1000	100	109	997	1083	159	242	643
10	100	1000	60	1100	100	110	1142	247	160	260	113
11	100	1100	61	1200	100	111	428	1291	161	2408	1329
12	100	1200	62	1300	100	112	795	2245	162	1132	1876
13	100	1300	63	1400	100	113	2392	173	163	522	1076
14	100	1400	64	1500	100	114	1051	176	164	1629	1795
15	100	1500	65	1600	100	115	309	2070	165	1999	805
16	100	1600	66	1700	100	116	565	687	166	2525	1368
17	100	1700	67	1800	100	117	201	1630	167	820	342
18	100	1800	68	1900	100	118	948	267	168	2571	2482
19	100	1900	69	2000	100	119	2402	1663	169	2368	851
20	100	2000	70	2100	100	120	162	1204	170	1533	1032
21	100	2100	71	2200	100	121	618	2213	171	601	2583
22	100	2200	72	2300	100	122	1478	1310	172	101	1967
23	100	2300	73	2400	100	123	2325	534	173	882	600
24	100	2400	74	2500	100	124	1989	1597	174	1433	1668
25	100	2500	75	2600	100	125	1981	1273	175	2093	1874
26	2700	100	76	200	2700	126	1921	2466	176	2044	2502
27	2700	200	77	300	2700	127	145	1524	177	765	1605
28	2700	300	78	400	2700	128	2502	2494	178	763	1096
29	2700	400	79	500	2700	129	1298	803	179	1016	1411
30	2700	500	80	600	2700	130	942	1576	180	1695	2060
31	2700	600	81	700	2700	131	1777	2084	181	473	559
32	2700	700	82	800	2700	132	835	1030	182	1315	794
33	2700	800	83	900	2700	133	2584	982	183	1228	1084
34	2700	900	84	1000	2700	134	2063	1499	184	691	1271
35	2700	1000	85	1100	2700	135	2046	230	185	1482	210
36	2700	1100	86	1200	2700	136	2137	1973	186	2017	1395
37	2700	1200	87	1300	2700	137	1615	2329	187	107	2492
38	2700	1300	88	1400	2700	138	2388	766	188	908	2040
39	2700	1400	89	1500	2700	139	1784	1196	189	1374	2073
40	2700	1500	90	1600	2700	140	1522	1385	190	1785	1792
41	2700	1600	91	1700	2700	141	204	1223	191	2423	1169
42	2700	1700	92	1800	2700	142	2455	2542	192	439	2094
43	2700	1800	93	1900	2700	143	1770	873	193	2372	2321
44	2700	1900	94	2000	2700	144	1113	1889	194	312	1832
45	2700	2000	95	2100	2700	145	2556	1733	195	2004	118
46	2700	2100	96	2200	2700	146	1763	2430	196	1607	2368
47	2700	2200	97	2300	2700	147	1595	2243	197	769	1535
48	2700	2300	98	2400	2700	148	1113	2024	198	2518	1160
49	2700	2400	99	2500	2700	149	2285	560	199	2055	1817
50	2700	2500	100	2600	2700	150	437	910	200	1654	1892

Table A_{15}: Input Data for Problem 240-1

P	X	Y	Z	P	X	Y	Z	P	X	Y	Z
1	1	0	1	41	1	0	3	81	1	21	2

2	2	0	1	42	2	0	3	82	2	21	2
3	3	0	1	43	3	0	3	83	3	21	2
4	4	0	1	44	4	0	3	84	4	21	2
5	5	0	1	45	5	0	3	85	5	21	2
6	6	0	1	46	6	0	3	86	6	21	2
7	7	0	1	47	7	0	3	87	7	21	2
8	8	0	1	48	8	0	3	88	8	21	2
9	9	0	1	49	9	0	3	89	9	21	2
10	10	0	1	50	10	0	3	90	10	21	2
11	11	0	1	51	11	0	3	91	11	21	2
12	12	0	1	52	12	0	3	92	12	21	2
13	13	0	1	53	13	0	3	93	13	21	2
14	14	0	1	54	14	0	3	94	14	21	2
15	15	0	1	55	15	0	3	95	15	21	2
16	16	0	1	56	16	0	3	96	16	21	2
17	17	0	1	57	17	0	3	97	17	21	2
18	18	0	1	58	18	0	3	98	18	21	2
19	19	0	1	59	19	0	3	99	19	21	2
20	20	0	1	60	20	0	3	100	20	21	2
21	1	0	2	61	1	21	1	101	1	21	3
22	2	0	2	62	2	21	1	102	2	21	3
23	3	0	2	63	3	21	1	103	3	21	3
24	4	0	2	64	4	21	1	104	4	21	3
25	5	0	2	65	5	21	1	105	5	21	3
26	6	0	2	66	6	21	1	106	6	21	3
27	7	0	2	67	7	21	1	107	7	21	3
28	8	0	2	68	8	21	1	108	8	21	3
29	9	0	2	69	9	21	1	109	9	21	3
30	10	0	2	70	10	21	1	110	10	21	3
31	11	0	2	71	11	21	1	111	11	21	3
32	12	0	2	72	12	21	1	112	12	21	3
33	13	0	2	73	13	21	1	113	13	21	3
34	14	0	2	74	14	21	1	114	14	21	3
35	15	0	2	75	15	21	1	115	15	21	3
36	16	0	2	76	16	21	1	116	16	21	3
37	17	0	2	77	17	21	1	117	17	21	3
38	18	0	2	78	18	21	1	118	18	21	3
39	19	0	2	79	19	21	1	119	19	21	3
40	20	0	2	80	20	21	1	120	20	21	3

Table A_{15}: Input Data for Problem 240-1(continue)

P	X	Y	Z	P	X	Y	Z	P	X	Y	Z
121	1	2	0	161	8	5	0	201	15	3	0

122	1	3	0	162	8	8	0	202	15	5	0
123	1	10	0	163	8	11	0	203	15	8	0
124	1	11	0	164	8	17	0	204	15	10	0
125	1	13	0	165	8	18	0	205	15	11	0
126	1	16	0	166	8	19	0	206	15	15	0
127	1	19	0	167	8	20	0	207	15	15	0
128	2	1	0	168	9	7	0	208	15	17	0
129	2	12	0	169	9	9	0	209	16	1	0
130	2	16	0	170	9	17	0	210	16	7	0
131	2	16	0	171	9	19	0	211	16	8	0
132	3	6	0	172	10	8	0	212	16	9	0
133	3	9	0	173	10	9	0	213	16	11	0
134	3	10	0	174	10	12	0	214	16	15	0
135	3	13	0	175	10	13	0	215	16	18	0
136	3	15	0	176	10	20	0	216	16	19	0
137	3	16	0	177	11	3	0	217	17	1	0
138	3	17	0	178	11	4	0	218	17	6	0
139	3	18	0	179	11	12	0	219	17	12	0
140	3	20	0	180	11	14	0	220	17	14	0
141	4	7	0	181	11	16	0	221	17	15	0
142	4	12	0	182	11	19	0	222	17	17	0
143	4	15	0	183	12	2	0	223	18	1	0
144	5	1	0	184	12	3	0	224	18	4	0
145	5	6	0	185	12	8	0	225	18	6	0
146	5	14	0	186	12	10	0	226	18	14	0
147	6	7	0	187	12	12	0	227	18	15	0
148	6	8	0	188	12	13	0	228	18	16	0
149	6	9	0	189	12	14	0	229	18	17	0
150	6	13	0	190	12	16	0	230	19	2	0
151	6	15	0	191	12	17	0	231	20	1	0
152	6	17	0	192	12	18	0	232	20	2	0
153	6	18	0	193	12	20	0	233	20	4	0
154	6	20	0	194	13	4	0	234	20	5	0
155	7	4	0	195	13	20	0	235	20	7	0
156	7	6	0	196	14	1	0	236	20	9	0
157	7	8	0	197	14	3	0	237	20	12	0
158	7	14	0	198	14	9	0	238	20	14	0
159	8	1	0	199	14	14	0	239	20	18	0
160	8	2	0	200	15	1	0	240	20	20	0

Table A_{16}: Input Data for Problem 240-2

P	X	Y	Z	P	X	Y	Z	P	X	Y	Z
1	1	0	1	41	1	0	3	81	1	21	2

2	2	0	1	42	2	0	3	82	2	21	2
3	3	0	1	43	3	0	3	83	3	21	2
4	4	0	1	44	4	0	3	84	4	21	2
5	5	0	1	45	5	0	3	85	5	21	2
6	6	0	1	46	6	0	3	86	6	21	2
7	7	0	1	47	7	0	3	87	7	21	2
8	8	0	1	48	8	0	3	88	8	21	2
9	9	0	1	49	9	0	3	89	9	21	2
10	10	0	1	50	10	0	3	90	10	21	2
11	11	0	1	51	11	0	3	91	11	21	2
12	12	0	1	52	12	0	3	92	12	21	2
13	13	0	1	53	13	0	3	93	13	21	2
14	14	0	1	54	14	0	3	94	14	21	2
15	15	0	1	55	15	0	3	95	15	21	2
16	16	0	1	56	16	0	3	96	16	21	2
17	17	0	1	57	17	0	3	97	17	21	2
18	18	0	1	58	18	0	3	98	18	21	2
19	19	0	1	59	19	0	3	99	19	21	2
20	20	0	1	60	20	0	3	100	20	21	2
21	1	0	2	61	1	21	1	101	1	21	3
22	2	0	2	62	2	21	1	102	2	21	3
23	3	0	2	63	3	21	1	103	3	21	3
24	4	0	2	64	4	21	1	104	4	21	3
25	5	0	2	65	5	21	1	105	5	21	3
26	6	0	2	66	6	21	1	106	6	21	3
27	7	0	2	67	7	21	1	107	7	21	3
28	8	0	2	68	8	21	1	108	8	21	3
29	9	0	2	69	9	21	1	109	9	21	3
30	10	0	2	70	10	21	1	110	10	21	3
31	11	0	2	71	11	21	1	111	11	21	3
32	12	0	2	72	12	21	1	112	12	21	3
33	13	0	2	73	13	21	1	113	13	21	3
34	14	0	2	74	14	21	1	114	14	21	3
35	15	0	2	75	15	21	1	115	15	21	3
36	16	0	2	76	16	21	1	116	16	21	3
37	17	0	2	77	17	21	1	117	17	21	3
38	18	0	2	78	18	21	1	118	18	21	3
39	19	0	2	79	19	21	1	119	19	21	3
40	20	0	2	80	20	21	1	120	20	21	3

Table A_{16}: Input Data for Problem 240-2(continue)

P	X	Y	Z	P	X	Y	Z	P	X	Y	Z
121	1	1	0	161	8	8	0	201	13	17	0

122	1	3	0	162	8	11	0	202	13	20	0
123	1	5	0	163	8	16	0	203	14	3	0
124	1	7	0	164	8	17	0	204	14	5	0
125	1	12	0	165	9	3	0	205	14	6	0
126	1	20	0	166	9	7	0	206	14	9	0
127	2	2	0	167	9	8	0	207	14	13	0
128	2	3	0	168	9	15	0	208	14	19	0
129	2	7	0	169	9	19	0	209	14	20	0
130	2	14	0	170	9	20	0	210	15	1	0
131	2	15	0	171	10	1	0	211	15	17	0
132	2	17	0	172	10	2	0	212	16	3	0
133	3	3	0	173	10	9	0	213	16	7	0
134	3	6	0	174	10	11	0	214	16	12	0
135	3	7	0	175	10	12	0	215	16	14	0
136	3	8	0	176	10	13	0	216	16	15	0
137	3	9	0	177	10	15	0	217	16	16	0
138	3	11	0	178	10	16	0	218	16	19	0
139	3	20	0	179	11	6	0	219	17	4	0
140	4	3	0	180	11	8	0	220	17	9	0
141	4	8	0	181	11	10	0	221	17	12	0
142	4	9	0	182	11	11	0	222	17	15	0
143	4	17	0	183	11	16	0	223	18	2	0
144	4	19	0	184	11	17	0	224	18	3	0
145	4	20	0	185	11	18	0	225	18	4	0
146	5	1	0	186	11	21	0	226	18	5	0
147	5	3	0	187	12	4	0	227	18	9	0
148	5	6	0	188	12	8	0	228	18	11	0
149	5	8	0	189	12	9	0	229	18	12	0
150	5	10	0	190	12	14	0	230	18	14	0
151	5	13	0	191	12	16	0	231	18	17	0
152	5	19	0	192	12	17	0	232	18	20	0
153	6	5	0	193	12	18	0	233	19	2	0
154	6	8	0	194	12	20	0	234	19	5	0
155	6	19	0	195	13	0	0	235	19	6	0
156	7	6	0	196	13	6	0	236	19	8	0
157	7	18	0	197	13	11	0	237	19	17	0
158	7	19	0	198	13	12	0	238	20	3	0
159	7	20	0	199	13	13	0	239	20	15	0
160	8	3	0	200	13	14	0	240	20	19	0

Table A_{17}: Labels of Problem 25-15-1 After Solving with the MTZ formulation

point	**Label**	**point**	**Label**	**point**	**Label**	**point**	**Label**
1	1	58	6	**145**	8	140	11

2	2	67	6	13	9	20	12
12	2	68	6	18	9	**49**	12
104	2	75	6	30	9	**87**	12
127	2	76	6	35	9	156	12
3	3	**79**	6	43	9	22	13
8	3	84	6	**45**	9	**32**	13
15	3	85	6	54	9	40	13
105	3	100	6	59	9	53	13
108	3	113	6	61	9	**98**	13
109	3	116	6	74	9	110	13
4	4	**129**	6	81	9	141	13
5	4	131	6	86	9	154	13
33	4	149	6	102	9	24	14
36	4	151	6	111	9	**47**	14
51	4	160	6	**112**	9	**82**	14
69	4	9	7	120	9	92	14
80	4	11	7	132	9	25	15
96	4	**42**	7	133	9	**28**	15
103	4	50	7	142	9	29	15
138	4	57	7	143	9	83	15
147	4	**64**	7	14	10	**88**	15
148	4	72	7	21	10	95	15
155	4	73	7	34	10	**26**	16
159	4	94	7	41	10	**119**	16
6	5	101	7	48	10	37	17
31	5	114	7	66	10	38	17
52	5	121	7	77	10	55	17
62	5	128	7	89	10	**65**	17
63	5	146	7	91	10	**90**	17
93	5	157	7	**122**	10	97	17
115	5	10	8	130	10	150	17
123	5	17	8	137	10	158	17
125	5	**39**	8	**144**	10	**56**	18
134	5	70	8	152	10	60	18
139	5	78	8	153	10	**107**	18
7	6	106	8	**19**	11	126	18
16	6	117	8	**27**	11	**71**	19
23	6	118	8	46	11	**124**	19
44	6	135	8	136	11	**99**	20
Bold digits show non-integer points							

Appendix B: LINGO Programs

Program B_1: The MTZ Model Formulation

```
MODEL:
SETS:
CITY / 1.. N/: U; ! U( I) = sequence no. of city;
LINK( CITY, CITY):
DIST, ! The distance matrix;
X; ! X( I, J) = 1 if we use link I, J;
```

```
ENDSETS
DATA: !Distance matrix, it need not be symmetric;
DIST = @OLE(FILE_ADDRESS, RANGE_NAME);
ENDDATA
N = @SIZE( CITY);
MIN = @SUM( LINK: DIST * X);
@FOR( CITY( K):
! It must be entered;
@SUM( CITY( I)| I #NE# K: X( I, K)) = 1;
! It must be departed;
@SUM( CITY( J)| J #NE# K: X( K, J)) = 1;
! Weak form of the subtour breaking constraints;
@FOR( CITY( J)| J #GT# 1 #AND# J #NE# K:
U( J) >= U( K) + (N-1)*X ( K, J) +(2-N)
));
@FOR( LINK: @BND( 0, X, 1));

@FOR( CITY( K)| K #GT# 1:
                U( K) <= N - 1 - ( N - 1) * X( 1, K);
                U( K) >= 1 + ( N - 2) * X( K, 1));
```

Program B_2: The DFJ Model Formulation

```
MODEL:
! DFJ MODEL FOR DIRECTED GRAPH;
SETS:
point / 1.. N/; ! U( I) = sequence no. of city;
LINK( point, point):X,DIST;
ENDSETS
DATA:
DIST=@ole(FILE_ADDRESS, RANGE_NAME);
ENDDATA
MIN = @SUM( LINK: X*DIST);
@FOR( LINK: @BND( 0, X, 1));
@FOR( LINK: @BND( 0, X, 1));
@FOR( point(K):
      @SUM( point( I)| I #NE# K: X( I, K)) = 1;
      @SUM( point( J)| J #NE# K: X( K, J)) = 1);
@FOR( point( K):
       @FOR(point( I)| I #NE# K: X(I,K)+X(K,I)<=1));
END
```

Program B_3: Cut Model for 10-City Problem (First_Cut Model)

```
model:
min=h0103+h0106+h0205+h0208+h0306+h0307+h0402+h0405+h0603+h0607+h060
9+h0802+h0809+h0906+h0908+h0504+h0701;

x01+x02+x03+x04+x06+x08+x09+x07+x05>=1;
x01+x02+x03+x04+x06+x08+x09+x07+x05<=8;

!Costraints of non-integer arcs;
h0103-.89*x01+.89*x03>=0;
h0106-.11*x01+.11*06>=0;
h0205-.50*x02+.50*x05>=0;
h0208-.50*x02+.50*x08>=0;
h0306-.39*x03+.39*x06>=0;
h0307-.61*x03+.61*x07>=0;
h0402-.50*x04+.50*x02>=0;
h0405-.50*x04+.50*x05>=0;
h0603-.110*x06+.110*x03>=0;
h0607-.39*x06+.39*x07>=0;
h0609-.5*x06+.5*x09>=0;
h0802-.5*x08+.5*x02>=0;
h0809-.5*x08+.5*x09>=0;
h0908-.5*x09+.5*x08>=0;
h0906-.5*x09+.5*x06>=0;

!Costraints for paths;
h0504-x05+x04>=0;
h0701-x07+x01>=0;

@gin(x01);
@gin(x02);
@gin(x03);
@gin(x04);
@gin(x05);
@gin(x06);
@gin(x07);
@gin(x08);
@gin(x09);

end
```

Program B_4: The MTZ Model and Added Cuts

```
MODEL:
! Traveling Salesman Problem with MTZ model and cuts;
SETS:
point / 1..10/: U, cutvalue; ! U( I) = sequence no. of point;
LINK( point, point):
DIST, ! The distance matrix;
X; ! X( I, J) = 1 if we use link I, J;

ENDSETS
DATA: !Distance matrix, it is symmetric;
DIST=@ole(fileaddress, range name);
cutvalue=@ole(fileaddress, rangename);
ENDDATA
N = @SIZE( point);
MIN = @SUM( LINK: DIST * X);
@FOR( point( K):
! It must be entered;
@SUM( point( I)| I #NE# K: X( I, K)) = 1;
! It must be departed;
@SUM( point( J)| J #NE# K: X( K, J)) = 1;
! Weak form of the subtour breaking constraints;
@FOR( point( J)| J #GT# 1 #AND# J #NE# K:
                 U( J) >= U( K) + (N-1)*X ( K, J) +(2-N)));
@FOR( LINK: @BND( 0, X, 1));
 For the first and last stop we know
  FOR(point( K)| K #GT# 1
          U( K) <= N - 1 - ( N - 1) * X( 1, K);
          U( K) >= 1 + ( N - 2) * X( K, 1 );
!added cuts;
  SUM
    (LINK(I,J)|cutvalue(I) #NE#0 #AND# cutvalue(J) #EQ#0:
X(I,J))>=1
END
```

Appendix C: MATLAB Programs

Program C_1: Program of Calculating Distance Matrix

```
clear all;
clc;
NP=Number of cell points;
x=[];
y=[];
z=[];

for i=1:NP
        for j=1:NP
            D(i,j)=((X(i)-X(j+NP))^2+(Y(i)-Y(j+NP))^2+(Z(i)-
Z(j+NP))^2)^.5;
        end
    end
    D2=D;
%D
xlswrite('xls file address', D);
```

Program C_2: Program of Plotting the TSP Solution

```
clear all
clc
V=xlsread('pointfile address');
M=xlsread('mtxfile address');

% It drawes a figure from the TSP solution
 [n,m]=size(M);
clf reset;
hold on;
grid on
for i=1:n
    plot(V(i,1),V(i,2),'b.')
for j=1:m
          if (M(i,j)+M(j,i)>1)

line([V(i,1);V(j,1)],[V(i,2);V(j,2)],'Color','k')

          end
          if ( M(i,j)+M(j,i)> 0.67 && M(i,j)+M(j,i)<=1)

line([V(i,1);V(j,1)],[V(i,2);V(j,2)],'Color','r')

          end
          if ( M(i,j)+M(j,i)> 0 && M(i,j)+M(j,i)<=0.67 )
             line([V(i,1);V(j,1)],[V(i,2);V(j,2)],'Color','g')
             [V(i,1);V(j,1)],[V(i,2);V(j,2)]
          end
      end
```

```
 end
```

Program C_3: Program of Labeling Technique

```
clear all
A= xlsread('LINGO output address');
[n,n] = size(A);
%A IS THE OUTPUT MATRIx OF LINGO (COSTOMIZED);
%B IS SYMMETRIC MATRIX OF A;
for i=1:n
    for j=1:n
        w=A(i,j);
        if A(j,i)>w
            w=A(j,i);
        end
        B(i,j)=w;
        B(j,i)=w;
    end
end
%GETTING THE LABELS;
for i=2:n
    Label(i)=0;
end
Label(1)=1;
L=1;
p=1;
pointer = 1;
counter=1;
while counter<=n && pointer<2*n-1
    for i=1:n
        if  B(p,i)>0 && Label(i)==0
            Label(i)=L;
            counter = counter+1;
        end
    end
    Label(p)=(-1)*(Label(p));
    p=0;
    pointer = pointer +1;
        for i=1:n
            if Label(i)>0
               pointer = pointer +1;
               break
            end
        end
        if p==0
           for i=1:n
               if Label(i)==0
                  p=i;
                  pointer = pointer +1;
                  L=L+1;
                  Label(p)=L;
                  counter = counter+1;
                  break
               end
           end
        end
end
```

```
for i=1:n
    Label(i)=abs(Label(i));
end
Label
```

Program C_4: Program of Proposed Heuristic _ Case (1)

```
clear all;
%clc;
NP=80;%number of the CP/AP points
for pp=1:80
D2=D;
CP(1)=pp;
[minv,idx]=min(D(CP(1),:));
AP(1)=idx;
SD=minv;%SD indicates the shortest path that has been inserted so
far;
T=SD;%the initial value of the Threshold;
D2(CP(1),:)=20000*ones(1,NP);
[minv,idx]=min(D2(:,AP(1)));
D2(:,AP(1))=20000*ones(NP,1);
if minv<T
    CP(2)=idx;
    cnt=3;%number of the nodes that have been inserted so far;
    SD=minv;
else
    cp=idx;
    [minv,idx]=min(D2(cp,:));
    ap=idx;
    if D(CP(1),ap)+D(cp,AP(1))<D(CP(1),ap)+D(cp,AP(1);
        AP(2)=AP(1);
        AP(1)=ap;
        CP(2)=cp;
    else
    D2(CP(2),:)=20000*ones(1,NP);
    d=[SD,D(CP(2),AP(1)),D(CP(2),AP(2))];
    SD=min(d);
    cnt=4;
    D2(:,AP(1))=D2(:,AP(2));
    D2(:,AP(2))=D(:,AP(2));
    D2(CP(1),AP(2))=20000;
    D2(CP(2),AP(2))=20000;
end
B=0;
while B==0
    if mod(cnt,2)==0
        api=cnt/2;
        [minv,idx]=min(D2(:,AP(api)));
        D2(:,AP(api))=20000*ones(NP,1);
        if minv<T
            CP(api+1)=idx;
            cnt=cnt+1;
            SD=minv;
        else
            cp=idx;
            [minv,idx]=min(D2(cp,:));
            ap=idx;
            for i=1:api-1
                td(i)=-D(CP(i+1),AP(i))+D(cp,AP(i))+D(CP(i+1),ap);
            end
            [mint,li]=min(td);
```

```
            CP(li+2:api+1)=CP(li+1:api);
            CP(li+1)=cp;
            AP(li+2:api+1)=AP(li+1:api);
            AP(li+1)=ap;
            cnt=cnt+2;
            for j=1:NP
                if j~=CP
                    D2(j,AP(api+1))=D(j,AP(api+1));
                end
            end
            D2(:,AP(li+1))=20000*ones(NP,1);
            D2(CP(li+1),:)=20000*ones(1,NP);

d=[SD,D(CP(li+1),AP(li)),D(CP(li+1),AP(li+1)),D(CP(li+1),AP(li+1))];
            SD=min(d);
            clear td;
        end
    else
        cpi=ceil(cnt/2);
        [minv,idx]=min(D2(CP(cpi),:));
        D2(CP(cpi),:)=20000*ones(1,NP);
        if minv<T
            AP(cpi)=idx;
            cnt=cnt+1;
            SD=minv;
        else
            ap=idx;
            [minv,idx]=min(D2(:,ap));
            cp=idx;
            for i=1:cpi-1
                td(i)=-D(CP(i),AP(i))+D(cp,AP(i))+D(CP(i),ap);
            end
            [mint,li]=min(td);
            CP(li+2:cpi+1)=CP(li+1:cpi);
            CP(li+1)=cp;
            AP(li+1:cpi)=AP(li:cpi-1);
            AP(li)=ap;
            cnt=cnt+2;
            for j=1:NP
                if j~=AP
                    D2(CP(cpi+1),j)=D(CP(cpi+1),j);
                end
            end
            D2(CP(li+1),:)=20000*ones(1,NP);
            D2(:,AP(li))=20000*ones(NP,1);

d=[SD,D(CP(li),AP(li)),D(CP(li+1),AP(li+1)),D(CP(li+1),AP(li))];
            SD=min(d);
            clear td;
        end
    end
    if cnt>=2*NP-1
        B=1;
    end
end
if cnt==2*NP-1
    AP(NP)=1;
end
ST=0;
for i=1:NP-1
    ST=ST+D(CP(i),AP(i))+D(CP(i+1),AP(i));
```

```
end
CP(1);
APP(pp)= CP(1);

    ST=ST+D(CP(NP),AP(NP))+D(CP(1),AP(NP));

    AST(pp) = ST;

end
for pp =1 :80
    RUN = pp
    APP(pp)
    AST(pp)
end

%PLOTTING THE GRAPH
for j=1:NP
    STR(2*j-1)=CP(j);
    SX(2*j-1)=X(STR(2*j-1));
    SY(2*j-1)=Y(STR(2*j-1));
    STR(2*j)=AP(j)+NP;
    SX(2*j)=X(STR(2*j));
    SY(2*j)=Y(STR(2*j));
end
plot(SX,SY)
```

Program C_5: Program of Proposed Heuristic _ Case(2)

```
clear all;
%clc;
NP=80;%number of the CP/AP points
for ml=1:NP
    D2=D;
    CP(1)=ml;
    [minv,idx]=min(D(CP(1),:));
    AP(1)=idx;
    SD=minv;%SD indicates the shortest path that has been inserted
so far
    T=SD
    D2(CP(1),:)=20000*ones(1,NP);
    [minv,idx]=min(D2(:,AP(1)));
    D2(:,AP(1))=20000*ones(NP,1);
    if minv<T
        CP(2)=idx;
        cnt=3;%number of the nodes that have been inserted so far
        SD=minv;
    else
        cp=idx;
        [minv,idx]=min(D2(cp,:));
        ap=idx;
        if D(cp,AP(1))<D(CP(1),ap)
            li=1;
            AP(2)=ap;
            CP(2)=cp;
        else
            li=0;
            AP(2)=AP(1);
            CP(2)=CP(1);
            AP(1)=ap;
            CP(1)=cp;
        end
        if li==0
            for j=1:NP
                if j~=CP
                    D2(j,AP(1+1))=D(j,AP(1+1));
                end
            end
            D2(:,ap)=20000*ones(NP,1);
            D2(cp,:)=20000*ones(1,NP);
        else
            D2(cp,:)=20000*ones(1,NP);
        end
        cnt=4;
         D2(CP(2),:)=20000*ones(1,NP);
         D2(:,AP(1))=D2(:,AP(2));
         D2(:,AP(2))=D(:,AP(2));
         D2(CP(1),AP(2))=20000;
         D2(CP(2),AP(2))=20000;
    end
    B=0;
```

```
    while B==0
        if mod(cnt,2)==0%a CP must be added
            api=cnt/2;%index of the last AP/CP
            [minv,idx]=min(D2(:,AP(api)));
            D2(:,AP(api))=20000*ones(NP,1);
            if minv<T
                CP(api+1)=idx;
                cnt=cnt+1;
                 else
                cp=idx;
                [minv,idx]=min(D2(cp,:));
                ap=idx;
                for i=1:api-1
                    td(i)=-
D(CP(i+1),AP(i))+D(cp,AP(i))+D(CP(i+1),ap);
                end
                td(api)=D(cp,AP(api));
                td(api+1)=D(CP(1),ap);
                [mint,li]=min(td);
                if li==api
                    CP(api+1)=cp;
                    AP(api+1)=ap;
                elseif li==api+1
                    CP(2:api+1)=CP(1:api);
                    CP(1)=cp;
                    AP(2:api+1)=AP(1:api);
                    AP(1)=ap;
                    d=[SD,D(CP(1),AP(1)),D(CP(2),AP(1))];
                else
                    CP(li+2:api+1)=CP(li+1:api);
                    CP(li+1)=cp;
                    AP(li+2:api+1)=AP(li+1:api);
                    AP(li+1)=ap;
                 end
                if li~=api
                    for j=1:NP
                        if j~=CP
                            D2(j,AP(api+1))=D(j,AP(api+1));
                        end
                    end
                    D2(:,ap)=20000*ones(NP,1);
                    D2(cp,:)=20000*ones(1,NP);
                else
                    D2(cp,:)=20000*ones(1,NP);
                end

                cnt=cnt+2;
                clear td;
            end
        else%an AP must be added
            cpi=ceil(cnt/2);%index of the last CP
            [minv,idx]=min(D2(CP(cpi),:));
            D2(CP(cpi),:)=20000*ones(1,NP);
            if minv<T
                AP(cpi)=idx;
                cnt=cnt+1;
            else
                ap=idx;
                [minv,idx]=min(D2(:,ap));
                cp=idx;
                for i=1:cpi-1
```

```
                    td(i)=-D(CP(i),AP(i))+D(cp,AP(i))+D(CP(i),ap);
                end
                td(cpi)=D(CP(cpi),ap);
                td(cpi+1)=D(CP(1),ap);
                [mint,li]=min(td);
                if li==cpi
                    CP(cpi+1)=cp;
                    AP(cpi)=ap;
                elseif li==cpi+1
                    CP(2:cpi+1)=CP(1:cpi);
                    CP(1)=cp;
                    AP(2:cpi)=AP(1:cpi-1);
                    AP(1)=ap;
                else
                    CP(li+2:cpi+1)=CP(li+1:cpi);
                    CP(li+1)=cp;
                    AP(li+1:cpi)=AP(li:cpi-1);
                    AP(li)=ap;
                end
                if li~=cpi
                    for j=1:NP
                        if j~=AP
                            D2(CP(cpi+1),j)=D(CP(cpi+1),j);
                        end
                    end
                    D2(cp,:)=20000*ones(1,NP);
                    D2(:,ap)=20000*ones(NP,1);
                else
                    D2(:,ap)=20000*ones(NP,1);
                end
                clear td;
            end
        end
        if cnt>=2*NP-1
            B=1;
        end
    end
    if cnt==2*NP-1
        AP(NP)=1;
    end
    ST=0;
    for i=1:NP-1
        ST=ST+D(CP(i),AP(i))+D(CP(i+1),AP(i));
    end
    TL(ml)=ST+D(CP(NP),AP(NP))+D(CP(1),AP(NP));
    for j=1:NP
        STR(ml,2*j-1)=CP(j);
        STR(ml,2*j)=AP(j)+NP;
    end
    clear AP;
    clear CP;

end
[mnv,mn]=min(TL);
 mnv
```

Appendix D: Some Output Plots

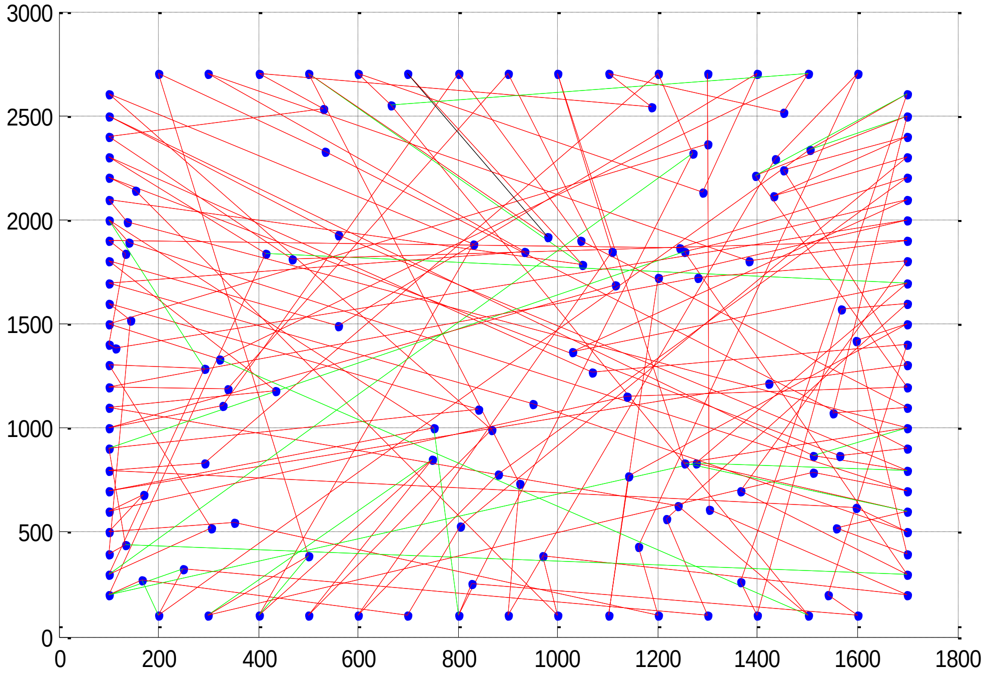


Figure E_1: Plot of Initial MTZ Output for Problem 25-15-1

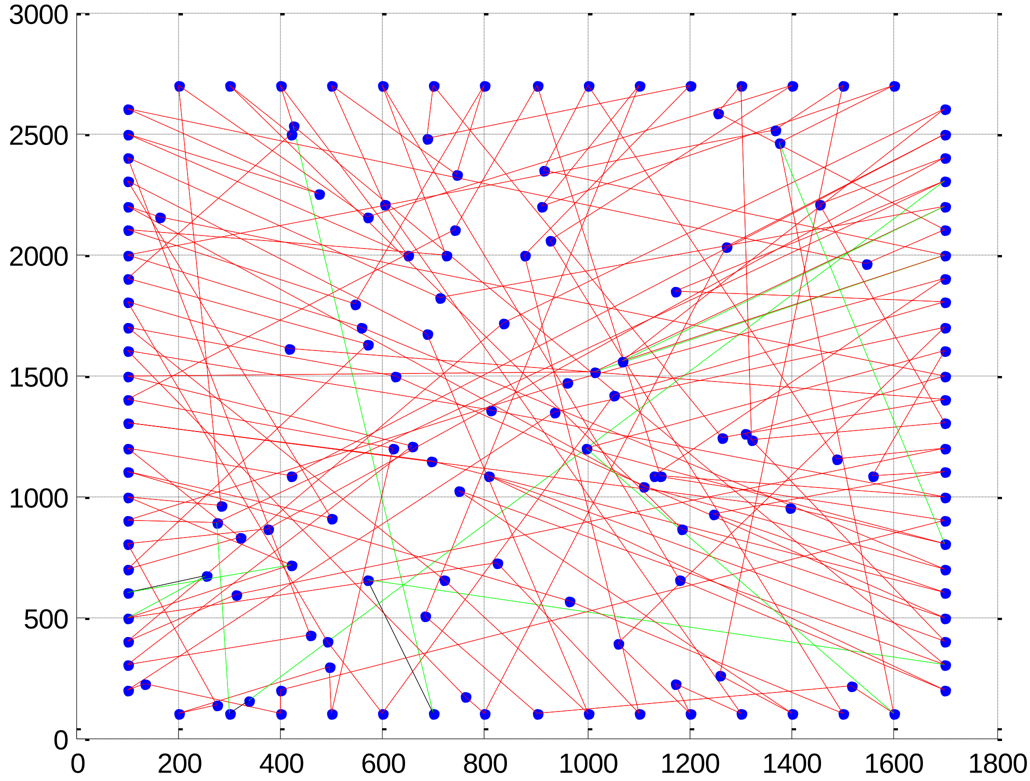


Figure E_2: Plot of Initial MTZ Output for Problem 25-15-2

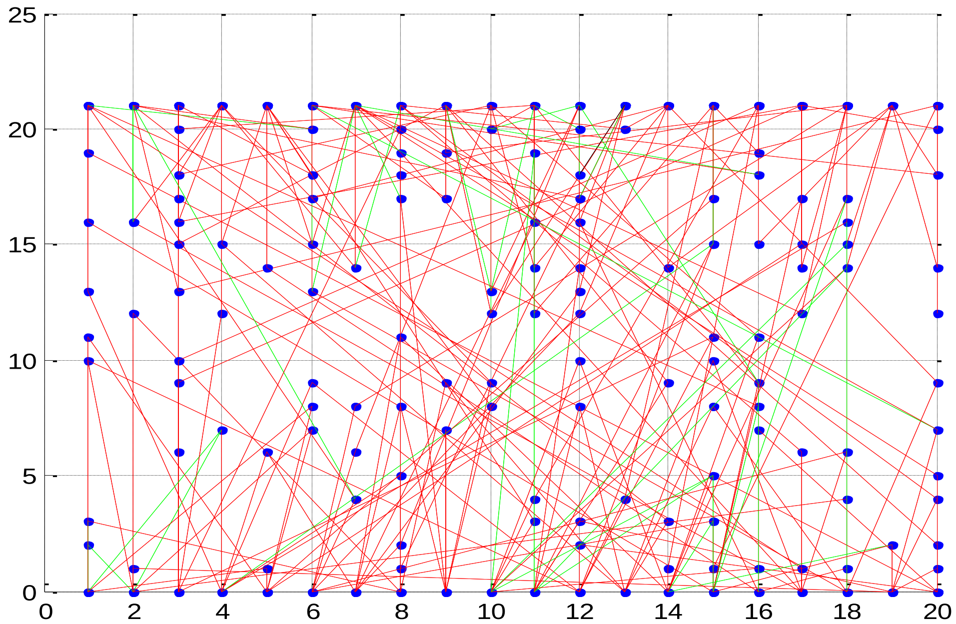


Figure E_3: Plot of Initial MTZ Output for Problem 240-1

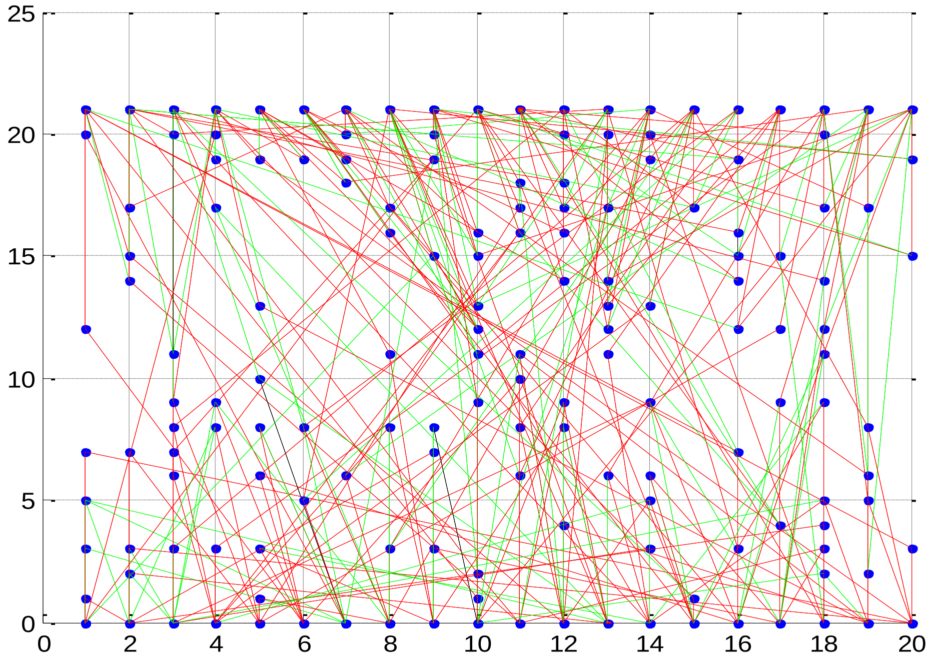


Figure E_4: Plot of Initial MTZ Output for Problem 240-2

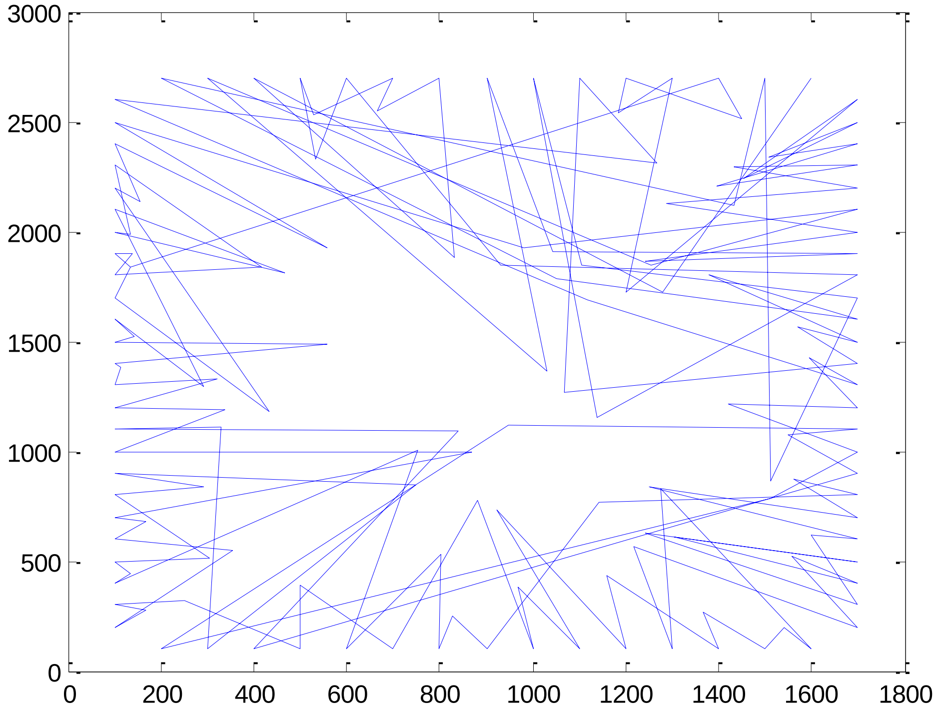


Figure E_5: Plot of the Best Heuristic Output for Problem 25-15-1

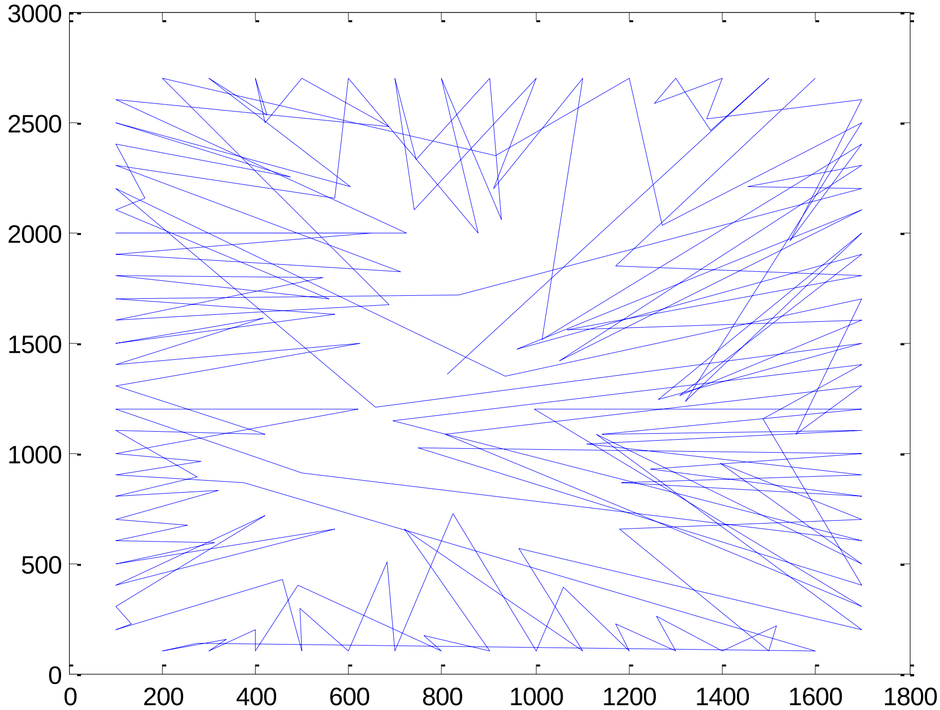


Figure E_6: Plot of the Best Heuristic Output for Problem 25-15-2

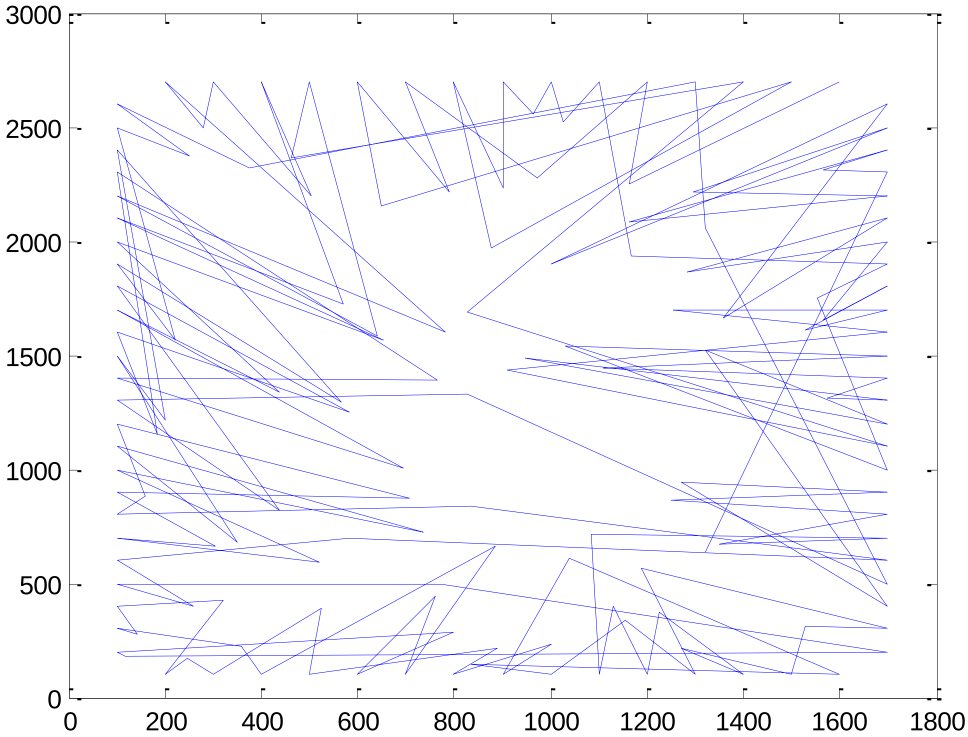


Figure E_7: Plot of the Best Heuristic Output for Problem 25-15-3

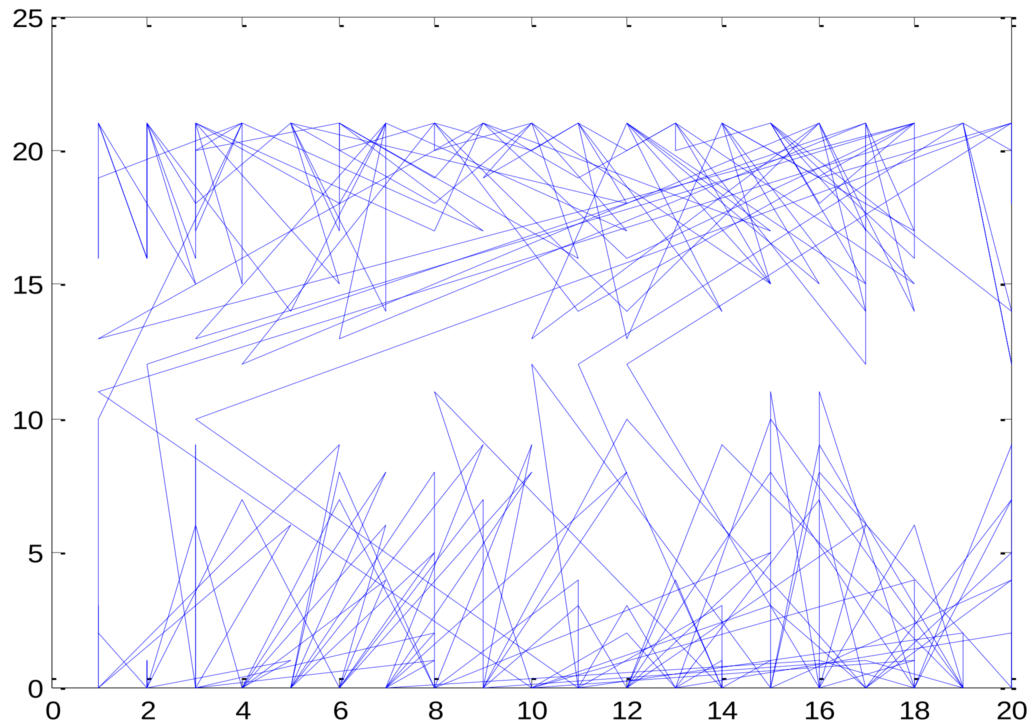


Figure E_8: Plot of the Best Heuristic Output for Problem 240-1

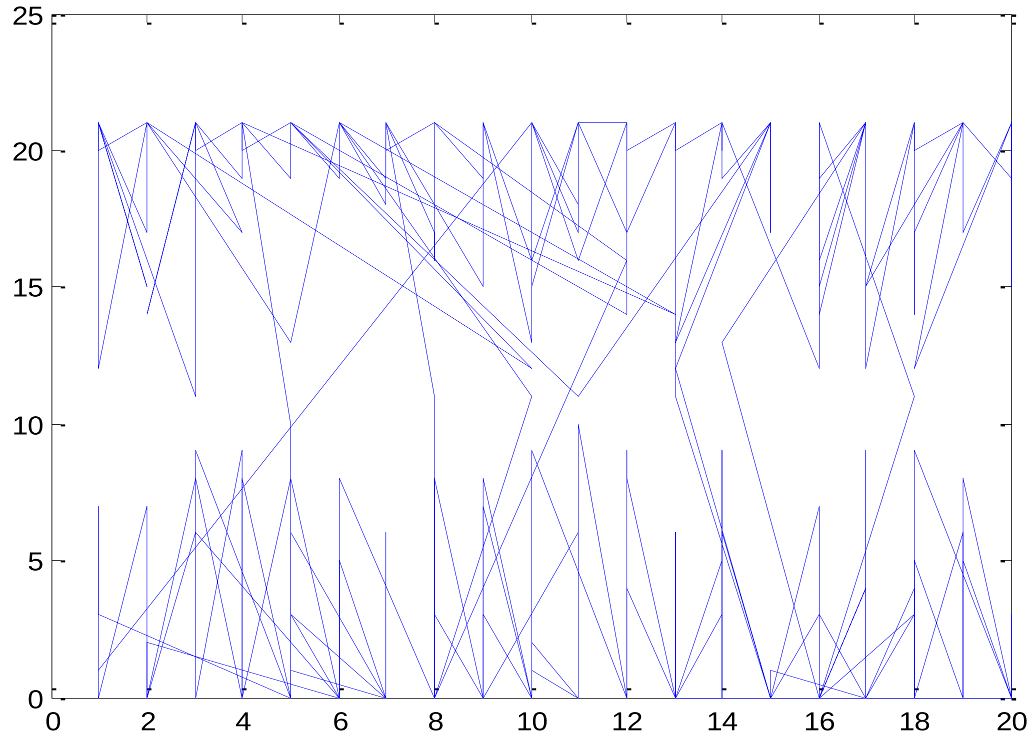


Figure E_9: Plot of the Best Heuristic Output for Problem 240-2

Chapter 1

INTRODUCTION

1. 1 Industrial Robots

According to the robotics research group of Robot Institute of America, "*a robot is a reprogrammable, multifunctional manipulator designed to move materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks.*"

Robots can be found in different fields of applications. These various applications consist of manufacturing industry, military, space exploration, transportation, research area, and medical applications. Typical industrial robots do some kind of jobs that are difficult, dangerous or dull. They can do the same task hour after hour and day after day not only without getting tired or making errors but also with precision. Therefore robots are ideally suited to perform repetitive tasks. Industrial robots are used in most industries such as automobile and manufacturing industries for loading bricks, dying cast, drilling, fastening, forging, making glass, grinding, heating treat, loading/unloading machines, handling parts, measuring, monitoring, running nuts, sorting parts, cleaning, sand blasting, changing tools and welding. The advantages of robots have become more apparent as robotic technology has grown and developed in the last 60 years when the first industrial robot with the name of

Unimate was put into use in the 1950s. Today, almost 90% of the robots in use today are in the industrial robotic sector in the factories. Robotics Industry Association (RIA) estimates that *"some 196,000 robots are now at work in U.S. factories, placing the United States second to Japan in overall robot use. More than one million robots are now being used worldwide. RIA currently represents some 235 robotics manufacturers, system integrators, component suppliers, end users, consulting groups, and research organizations. A total of 9,628 robots valued at $618.4 million were ordered through September by North American manufacturing companies. This represents a gain of 34% in units and 45% in dollars over the same period in 2009. Companies outside of North America ordered another 1,778 robots valued at $102.6 million from North American based robotics companies during the period, a gain of 143% in units and 168% in dollars over the first nine months of 2009."*

1.1.1 Robot Structure

The structure of a robot is directly related to its design purpose. Industrial robots usually take the shape of an arm because many tasks require the flexibility of human hands. Looking back at the history of robot development, a human-size industrial robotic arm called Programmable Universal Machine for Assembly (PUMA) came into existence. Because of the similarities between PUMA's structure and the human arm, it is often termed *anthropomorphic*.

Robotic arms re generally too rigid devices. They perform repetitive tasks under programmed control in the controlled environments.

1.1.1.1 Body of the Robotic Arm

Most of the robotic arms use the following five joint types.

(i.) *Prismatic joints*: create a linear movement.

(ii.) *Rotary joints*: drive by electric motors.

(iii.) *Spherical joints*: needed for a revolving movement.

(iv.) *Screw joints*: follow the thread of the axis in spiral in order to move along the axis.

(v.) *Cylindrical joints*: are used in some equipment like parallel robots.

Different robotic arms configurations are formed by combination of the above joints. The motion of the arm is up and down, generally. The robot can perform this motion by extending a cylinder. Cylinder is built into the arm. A robot is stopped when it hits a stop. The cylinders are moved using air pressure that is controlled by solenoid values. Additional movement can be done by attaching a wrist to the end of this arm cylinder. The wirst will be complex enough to provide some additional degrees of freedom.

1.1.1.2 Robot Head(s)

Every arm is equipped with one or more heads. Head is responsible for picking and placing components. A head for an industrial robot consists of:

(i.) A head body mounted on an end of an arm,

(ii.) An internal motor for generating a rotational torque,

(iii.) A nut member supported by head body,

(iv.) A guide member rotatable supported by head body,

(v.) A screw rod for passing through and threaded engaging with mentioned nut member,

(vi.) A shaft having a non-circular shape,

And some devices needed to support above components.

1.1.2 Robot specifications

(i.) *Accuracy*: when robot's program calls the robot to move to a considered point, it does not actually perform as specified. The accuracy measures such a gap. In other words, the distance between the considered position and the actual achieved position is defined as the accuracy of the robot.

(ii.) *Repeatability*: the ability of a robotic mechanism to repeat the same motion is called repeatability. In fact, repeatability measures the variability of repeatedly reaching for a single position.

(iii.) *Degree of freedom*: every axis on the robot defines a degree of freedom. Each degree of freedom can be n the slider, rotary or other types of actuator. The number of degrees of freedom introduces the number of independent ways in which a robot arm can move.

(iv.) *Resolution*: the smallest increment of motion that can be controlled by the robotic control system is called resolution. Resolution is dependent on the distance between the tool center point and the joint axis.

(v.) *Envelope*: a three-dimensional shape that introduces the boundaries that the robot can reach is called envelope.

(vi.) *Reach*: the maximum horizontal distance from the center of the robot is called reach.

(vii.) *Maximum Speed*: the theoretical full speed which does not consider under loading condition defines the maximum speed of the robot.

(viii.) *Payload*: the amount of weight carried by the robot manipulator at reduced speed without loosing the rated precision is known as payload.

1.1.3 Robot Classifications

Industrial robots have already been classified by mechanical structure as follows:

(i.) Cartesian/Gantry Robots: a Cartesian coordinate robot has three directions of movement in such a way that three prismatic axes (*X*, *Y*, and *Z*) are at right angles to each other. Gantry robots are such Cartesian robots with the horizontal member supported at both ends. Both of them, Cartesian and gantry robots, have a rectangular work envelope. These types of robots are highly rigid but they are very accurate and repeatable but lack of flexibility is seen in reaching around objects. These robots are very easy to perform and visualize. Cartesian robots are suited for pick and place applications. Gantry robots also have a wide range of applications in material handling such as pick and place, machine loading and unloading, stacking and palletizing. A sample Cartesian robot is shown in Figure 1.1.

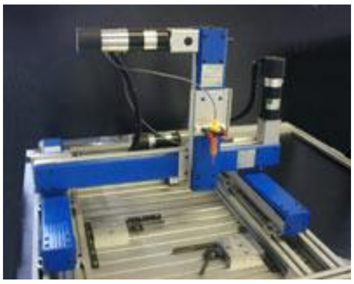


Figure 1.1: Cartesian Robot

(ii.) SCARA Robots: Selective Compliant Assembly Robot Arm (SCARA) robots can move to any direction of *X*, *Y*, and *Z* axes within their work envelope. Since the controlling software of SCARA robot requires inverse kinematics for linear interpolated moves, these robots are so expensive. Because of the rigidity in the vertical direction and flexibility in the horizontal plane, SCARA robots are suited for assembly operations such as inserting a round pin in a round hole without binding. They are also used for pick and place works and handling machine tools. A sample SCARA robot is shown in Figure 1.2.

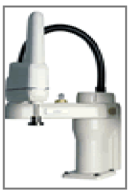


Figure 1.2: SCARA Robot

(iii.) Articulated Robots: the mechanical structure of articulated robots has at least three rotary joints which form a polar system. This structure is very flexible and can achieve any position and orientation within the working envelope. Articulated robots are used for paint spraying, spot welding, machine tending, die-casting, packing, gluing, etc. A sample articulated robot is shown in Figure 1.3.

Figure 1.3: Articulated Robot

(iv.) Parallel Robots: these robots have arms that each one has three concurrent prismatic joints. Parallel robots are able to manipulate large loads. They are used in a large number of applications ranging from astronomy to flight simulators. Less flexibility of parallel robots results in high repeatability. A sample parallel robot is shown in Figure 1.4.

Figure 1.4: Parallel Robot

(v.) Cylindrical Robots: the body structure of cylindrical robots is such that the robotic arm can move up and down along a vertical member. In the other words, these robots have at least one rotary joint and at least one prismatic joint. This construction makes the robot able to work in a cylindrical shape. Cylindrical robots are used for assembly operations, spot welding, die-casting and handling machine tools. A sample cylindrical robot is shown in Figure 1.5.

Figure 1.5: Cylindrical Robot

(vi.) Polar Robots: the other name of polar robots is spherical. These types of robots have an arm with two rotary joints and one prismatic joint. Polar coordinate system results short vertical reach. Because of long horizontal achievement, polar robot is useful for spot welding, felting machines, arc welding and gas welding. A sample polar robot is shown in Figure 1.6.

Figure 1.6: Polar Robot

1.2 Pick and Place Robots

Our focus in this thesis is on Pick and Place machine for placement of electronic components on Printed Circuit Board (PCB). A PCB is a board on which several resistors, transistors and diodes are mounted. For the manufacturing of PCB, the components are stored in one or more *feeders* from which a computer-controlled pick and place machine transfers them to a location on the PCB where they are to be fixed. Placement machines are also called "*chip shooters*". In the aspect of Surface Mount Technology (SMT), there are many types of placement machines available, such as sequential pick and place, concurrent pick and place, rotary disk turret, etc. Since different types of SMT placement machines have different characteristics and restrictions, the PCB production scheduling process is highly influenced by the type of placement machine being used. Most of the placement machines used in PCB assembly industry are Cartesian robots.

In general, each placement machine has a PCB table, feeder carrier, head, nozzle, and a tool magazine. Each of the feeder carrier, PCB table and head can be either fixed or moveable depending on the specification of the placement machine. Usually several tape reels or vibratory ski slope feeders or both of them construct a common feeder carrier. Positioning of the feeder reels or vibratory ski slope feeders is done according to the arrangement given by feeder setup. The role of the nozzle is grasping the component from the feeder and then mounting it on the PCB. Picking and placing the components is the responsibility of the placement arm that is equipped with head(s). Every placement machine may have more than one head and every head of the placement machine may have more than one nozzle. Placement machines have various types of heads such as rotating turret head, positioning arm head, etc. The PCB table is needed to position printed circuit boards during placement operation. Different sizes of nozzles are required for different sizes of surface mount devices to pick and place them. A tool magazine is required to provide the exact size of nozzles. A sample pick and place machine is shown in Figure 1.7.

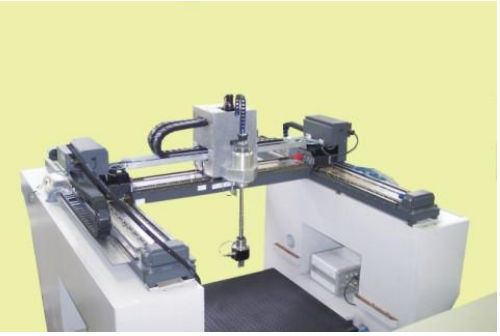


Figure 1.7: Pick and Place Machine

In fact, pick and place machine is the heart of SMT. A pick and place machine picks electronic components and places them onto the PCB. Some of them are capable of placing many different components used in electronics, while others are limited to a few component types. In our concentrated cases, pick and place machine can pick only one component at a time, which should be fixed first before the machine can handle another component. Vacuum pick up tools are used in pick and place machines in order to hold the components. Vision-assisted alignment is also used in few others of such machines. Some of the famous pick and place robots in addition of the manufacturer and the important specifications of them have been collected in Table A_1.

1.3 Traveling Salesman Problem (TSP)

When hundreds of electronic components of different shapes and sizes have to be placed at specific positions on a PCB, finding an optimal robot traveling path is so complex and time consuming. The problem to be solved here is finding a sequence in which the assembly points are to be assembled in order to minimize the total assembly time and increasing the productivity. The problem of determining the optimum sequence of points can be considered as an extension to TSP.

One of the most intensive studied problems in computational mathematics is the traveling salesman problem, the task of finding the shortest tour through a given list of cities and their pairwise distances that visits each city exactly once. It is a well-known NP-Complete combinatorial optimization problem. TSP has received much attention from mathematicians and computer scientists, especially since it is so easy to describe but is very difficult to solve optimally. The importance of the traveling salesman problem starts not only from a need of salesman wishing to minimize traveled distance, but comes from a wealth of other applications, many of which seem completely unrelated to traveling routes. Many practical applications can be modeled as TSP or a variant of it.

It is clear that theoretical and practical insight achieved in the study of TSP can often be useful in the solution of real-world problems. It is also valuable to mention that an important driving force in the development of the computational complexity theory was research on TSP in the beginning of the 1970s.

In the last three decades an improved progress has been made with respect to solving traveling salesman problems to optimality which is the main goal of every researcher. The number of cities in practical applications ranges from some small number up to even millions that is far beyond the capabilities of any exact algorithm available today. Due to this manifold area of applications of TSP, there should be abroad collection of algorithms to treat with the various instances of TSP. Landmarks in the search for optimal solutions have been shown in Table 1.1. The time has been needed to solve the last mentioned instances in Table 1.1 is more than several years using the big processors. It should be considered how is easy or difficult to solve a problem depends on many factors. The mathematical properties of the distance matrix are important, i.e. whether or not the triangle inequality and symmetry are satisfied. The structure of the positions of the cities is also very important, i.e. problems arising from chip design are much easier than the problems containing real cities. In spite of these achievements, the traveling salesman problem is still far from being solved. Many aspects of the traveling salesman problem still require to be considered and the questions are still left to be answered.

Table 1.1: Milestones in the Solution of TSP Instances

Year	Research Team	Size of instance	Name
1954	Dantzig, Fulkerson and Johnson	49 cities	dantzig42
1971	Held and Karp	64 cities	64 points
1975	Camerini, Fratta and Maffioli	67 cities	67 points
1977	Grotschel	120 cities	gr120
1980	Crowder and Padberg	318 cities	lin318
1987	Padberg and Rinaldi	532 cities	att532
1987	Grotschel and Holland	666 cities	gr666
1987	Padberg and Rinaldi	2,392 cities	pr2392
1994	Applegate, Bixby, Chvatal and Cook	7,397 cities	pla7397
1998	Applegate, Bixby, Chvatal and Cook	13,509 cities	usa13509
2001	Applegate, Bixby, Chvatal and Cook	15,112 cities	d15112
2004	Applegate, Bixby, Chvatal, Cook and Helgaun	24,978 cities	sw24978
2006	Applegate, Bixby, Chvatal and Cook	85,900 cities	pla85900

1.4 Outline of the Thesis

As it is mentioned at the outset the primary goal of this work is to find an acceptable technique to solve medium-size arm assembly problems. Continuing some of the discussions began in this chapter, we also cover a brief history of traveling salesman problem, exact and heuristic algorithms were proposed to solve various types of TSP and explaining the proposed technique for solving medium-size bipartite TSPs.

In chapter 2 we begin with the origin of the TSP, and follow with the existing methods for solving traveling salesman problems with the discussion about the history of the algorithms. In chapter 3 we will have a brief survey of exact and heuristic algorithms in detail and will give the relation between discussed contents and proposed technique. The proposed method to optimize the production time (or cost) caused by the distance that the robotic arm has to travel in the printed circuit board assembly problem is presented in chapter 4. Results of computational tests are given in chapter 5. Finally, in conclusion we discuss some of the research objectives and achievements. Required coding programs and computational documents will be given in the appendices.

Chapter 2

Literature Review

2.1 Traveling Salesman Problem Origin

The origin of the name TSP is a bit of mystery. There is not any authorittative documentation pointing out the creator of the traveling salesman name for this problem, and there is not good guesses as to when it first came into use. The numerous salesmen on the road were interested in the subject of the planning of economical routs according to customer area of them. A most important reference in this context is the German Handbook *Der Handlungsreisende* in 1832[4]. This handbook first brought to the attention of the traveling salesman problem research community by Heiner Muller-Merbach[4]. The mentioned book was not alone in considering planned tours. In the late 1800s, Spears and Friedman described how a salesman used guidebooks to map out routs through their regions. One of such guidebooks is L.P.Brockett's commercial traveler's guide book [4]. In the 1920's, Karl Menger (the mathematician and economist) publicized it in Vienna [4]. In the 1930's, traveling salesman problem reappeared in the mathematical circles of Princeton. It was studied by statisticians (Mahalanobis (1940) and Jessen(1942)) [4] in connection with an agricultural application. Then Merrill Flood, who was a mathematician, popularized it at the RAND Corporation in the 1940's [4]. At last, the TSP became as the prototype of a hard problem in combinatorial optimization. Over the years wealth of algorithmic creativity has been applied to TSP, and excellent surveys of TSP algorithms can be found in many articles. We hope to provide a useful review of widely known algorithms, divided into two main classes: exact algorithms, and heuristic algorithms which the heuristics can be divided into three types of algorithms.

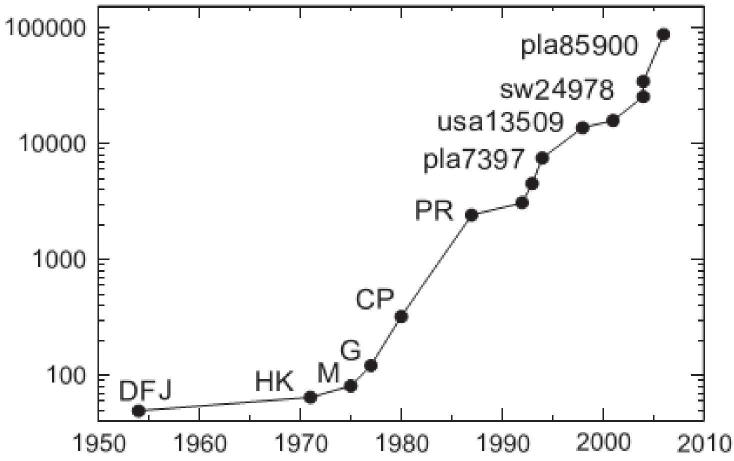
2.1.1 Exact Algorithms

These algorithms are guaranteed to find the optimal solution in a bounded number of iterations. Linear programming is very useful tool in this way. An important feature of linear formulations is that even very large s can be solved efficiently with a variety of new and old solution methods. The most important of these solution techniques is simplex method which was proposed by George Dantzig in 1947. The simplex method was also vital in the context of TSP. Many of the state of the art LP solvers which are available today use simplex method.

In 1954 when George Dantzig, Ray Fulkerson, and Selmer Johnson published a description of a method for solving the TSP, a breakthrough came in solving this problem. They illustrated the power of this method by solving an with 49 cities that was an impressive size at that time. This of the TSP was included of the 48 states of the U.S.A in that time and Washington D.C.; such that the costs of travel between different cities were defined as pairwise distances of cities taking from an atlas. Rather than solving this 49-city problem, Dantzig, Fulkerson, Johnson firstly solved the 42-city problem obtained by removing 7 states. Since the shortest route between Washington D.C. and Boston passes through the seven removed cities, also in the optimal tour of the 42-city problem had an edge of passing through the mentioned two cities; the solution of the 42-city problem yielded a solution of the 49-city problem. Using the simplex method and following the studies of Robinson (1949) and Kuhn (1955) they attacked the salesman with linear programming as follows.

Each TSP with *n* cities can be specified as a vector whose components specify the traveled costs and each tour through the *n* cities can be represented as its incidence vector in order to minimize the total costs of the tour. Thus the first exact mathematical model of TSP was developed by Dantzig, Fulkerson and Johnson. The main disadvantage of their method was having exponentially constraints. An alternate linear formulation that reduced the number of constraints at the expense of additional real variables was developed by Miller, Tucker, and Zemlin (1960). It was originally proposed for a vehicle routing problem where the number of vertices of each route is limited.

In 1962, Held and Karp solved a problem with 48 cities using dynamic programming. Because of many computation steps and large storage locations, dynamic programming was not so practical. Consequently, practical application of dynamic programming in the context of TSP is restricted to tours with few cities. In the 1960's, Little et al. proposed an algorithm for TSP in such a way that *branch and bound* term coined in conjunction with their algorithm. The branch and bound method can handle large case problems but the disadvantage is unpredictable computing time and it increases rapidly when the size of the problem increases. Also, other integer and mixed integer formulations have been proposed based on DFJ formulation in the next years. For an extensive list of such formulations the paper of Langevin et a addressed. One of the well known variant formulations of DFJ belongs to Padberg and Sung (1991). They solved some large problems in such a way that DFJ linear relaxation is properly contained in their linear relaxation. These efforts yielded to find an exact solution for 15,112 German cities in 2001 using cutting plane method proposed by Dantzig et al. (1954). It is interesting to know the computations were performed on a network of 110 processors and its computation time was equivalent to 22.6 years on a single 500 MHz Alpha processor. In April 2004, the instance of 24,978 cities in Sweden was solved but for solving this problem with a large number of processors was spent more than 10 years. Applegate et al. (2006) solved the biggest size instance of TSP library that is called *pla85,900*. Solving this problem was run on sun Microsystems with 250 processors and the total CPU time was 568.9 hours. In Figure 2.1 progress in TSP with the log scale has been shown [44].



Year

Figure 2.1: Progress in TSP, Log Scale

2.1.2 Heuristic Algorithms

Despite of exact algorithms, heuristic algorithms obtain good solutions but do not guarantee that optimal solutions will be found. Heuristics are usually very simple and have short running times. Some of the heuristic algorithms provide solutions such that in average differ only by a few percent from the optimal solution. Therefore, when running time is limited and a small deviation from optimum is acceptable, it may be appropriate to use a heuristic algorithm. TSP heuristic algorithms can be roughly partitioned into the following four classes: constructive algorithms, iterative improvement algorithms, composite algorithms, and randomized improvement algorithms. All classes and their performances in computational experiments will be discussed below.

2.1.2.1 Constructive Algorithms

Constructive algorithms determine a tour according to some construction rule, but do not try to improve upon this tour. In other words, a tour is successively built from scratch and stop, when one tour is produced. In most of constructive algorithms, the initial subtour is simply a randomly selected city. In addition to initial subtour construction, a distinction is made between deciding which city is chosen to be inserted into the current subtour and where the city is to be inserted. The choice of selection and insertion criteria in the selection and insertion steps of tour construction can be critical to the success of a heuristic algorithm.

Many of the construction heuristics presented here are known and computational results for some s are available. These heuristics consist of so many algorithms such as: Nearest Neighbor Heuristics, Insertion Heuristics, Heuristics based on Spanning Trees, and Saving Heuristics. Golden and Stewart (1985), Arthur and Frendeway (1985), Johnson (1990), Bentley (1992) have proposed such heuristics and the results of computational efforts are available in lecture notes of Gerhard Reinelt in 1994. The simplest and most obvious construction algorithm is the Nearest Neighbor algorithm. Computational experiments in [14] indicate that in most real-world problem s of ATSP (Asymmetric TSP), nearest neighbor performs better than the other algorithms even greedy algorithm which is one of the most important construction heuristics. Although, the computational experiments in [15] displays that both of the nearest neighbor and greedy algorithms perform well on Euclidean s but are poor in other cases of general STSP (Symmetric TSP). It should be noted that tour construction heuristics are important in the context of this thesis not only for the perspectives they provide but also because they can be used to generate the initial tours needed by other heuristics that will be explained.

2.1.2.2 Iterative Improvement Algorithms

Improvement heuristics improve upon a tour by performing different exchanges until there is no feasible exchange that improves the current solution. Since the construction heuristics were only of moderate quality, the improvement heuristics were proposed. In general, iterative improvement algorithms are characterized by a certain type of basic move to change the current tour. These algorithms are faster than exact algorithms and often produce solutions close to the optimal solution. The mentioned algorithms are referred to as r-Opt, where r is the number of edges exchanged at each step. Generally, the larger the value of r, the more likely it is that the final solution is optimal. Unfortunately, the number of operations is needed to test all r exchanges increases exponentially as the number of cities increases; hence, the most common values of r are 2 or 3.

The most famous iterative improvement heuristics are as follows: Node and Edge Insertion, 2-Opt Exchange, 3-Opt heuristics and variants, and Lin-Keringhan type heuristics. Computational experiments in [36] shows that Lin-Keringhan heuristics obtain better solutions than the others. These results indicate that if one wants to get solutions at most 1-2% above the optimal solutions, he/she has to implement Lin-Keringhan heuristics.

Further improvement heuristics have been proposed. E.g., Gendreau, Hertz and Laporte (1992) and Glover (1992) [36] discussed additional types of exchange moves. Moreover, the effect of the choice of the starting tour on the final result of improvement has been considered in Perttunen (1991) [36].

2.1.2.3 Composite Algorithms

Composite algorithms combine the features of constructive and improvement algorithms to solve the problems. These heuristic algorithms start from a tour in a single attempt generally obtained by constructive algorithms, and then iteratively modify a given starting solution. The obtained solution is dependent on the initial starting point because the choice of the starting city affect on the final result. One of the earliest composite algorithms has been given by Lin in 1965. After Lin, Jacque, and Fayez (1995) [36] gave an extension of it for the symmetric generalized traveling salesman problem.

2.1.2.4 Randomized Improvement Algorithms

At least in principle, every TSP heuristic algorithm has the chance of obtaining optimal tour. However, it is really an impossible event. When an improvement method finds a locally optimal tour, it means that no further improving moves can be generated. The weaker the local moves that can be implemented, the larger is the difference between the length of the optimal tour and of the locally optimal tour found by the heuristic algorithm. A way to get better performance is to start improvement heuristics many times with different starting tours in order to increase the chance of finding better local optimum. Another possibility is to consider the current tour by some modification to restart heuristics.

Randomized improvement heuristics try to use a symmetric rule to escape from local minimum. In the other words, these algorithms utilize local searching to find routes. Examples of this subsection are: Simulated Annealing, Genetic Algorithm, Tabu Search, and Neural Networks. Computational experiments of simulated annealing have been given by Kirkpatrick (1984), Cerny (1985), Van Laarhoven (1988), Aarts and Korst (1989), and Johnson (1990), and Johnson and McGeoch (1995) [36]. Application of genetic algorithm has been reported in Fruhwirth (1987), Muhlenbein , Gorges-Schleuter and Kramer (1988), and Ulder, Pesch, Vav Laarhoven, Bandelt and Aarts (1990) and Johnson and McGeoch (1995). Glover (1989) [36] gives a detailed introduction to tabu search methods. Knox and Glover (1989) [36], Malek, Guruswamy, Owens and Pandya (1989), and Malek, Heap, Kapur and Mourad (1989) [36] report good computational results for using tabu search. A detailed explanation of neural networks is found in the report of Henriques, Safayeni and Fuller in 1987. Fritzke and Wilke (1991) [36] give a further neural network algorithm for the TSP. A survey of different models can be found Potvin (1993) [36].

It can be shown that if running time be not a major concern, then randomized improvement heuristics can be successfully employed since they usually avoid bad local optima and have a chance to even obtain optimal solutions. .

2.1.3 Polyhedral Approaches of TSP

As stated before, combinatorial optimization problems such as TSP are usually relatively easy to formulate mathematically but most of them are computationally difficult due to the limitation that all or a subset of the variables have to take integral values. During the last three decades there has been a remarkable progress in techniques based on the polyhedral description of these problems so those techniques lead to a large increase in the size of the solved problems. The main idea behind polyhedral approaches is to derive a linear formulation of the set of solutions by defining some linear inequalities such that these inequalities must be included in the description of the convex hull of the integer feasible solutions. As we know, the convex hull for a set of points X $(H_{\text{convex}}(X))$ in a real vector space V is the minimal convex set containing X. The convex hull of the integers is the integer hull of set S is shown by $\text{conv}\,(S \cap Z_m)$.

Ideally everyone can then solve the combinatorial optimization problem as the linear programming problem. The computational hardness of traveling salesman problem has motivated researchers to develop formulations or algorithms that are expected to reduce the number of iterations in solving large s. Using the structure of the convex hull of the integer feasible solutions has been one of the most successful techniques so far. The first main work in this direction was done by Dantzig, Fulkerson and Johnson (1954). Their method in solving the 49 cities problem was based on the description of the convex hull of feasible solutions by linear inequalities and is called *polyhedral combinatorics*.

When studying Dantzig, Fulkerson and Johnson, a question arises whether it is possible to develop a method for identifying the inequalities. The answer of the question was done by Gomory (1958), (1960), (1963) who invented a cutting plane algorithm for general integer linear programming. Chvatal (1973) proved inequalities that are needed for the description of convex hull of integer solutions can be obtained by taking linear combinations of original inequalities. Schrijver (1980) proved that the number of operations to the linear formulation containing the integer solutions to generate the convex hull of integer solutions is finite. The results of Gomory, Chvatal, and Schrijver were very important in the sense of the theory of combinatorial optimization but they did not provide tools for solving real-life s within reasonable time. Scientists therefore began to search for inequalities included inequalities that are necessary in the description of the convex hull of feasible solutions and then identified the separation algorithms to find the violated inequalities. There are families of valid inequalities and the corresponding separation algorithms for TSP. The first class of these inequalities is called subtour elimination constraints which were developed by Dantzig, Fulkerson and Johnson. Comb inequalities are such valid inequalities were introduced by Chvatal (1975). These inequalities will be described in chapter 3. After Chvatal, Grotschel and Padberg (1979) were generalized his famous inequalities. Then Grotschel and Pulleyblank (1986) introduced the other useful inequalities called clique tree inequalities. Many exotic classes of valid inequalities have been introduced to date but the search for the new ones is still vivid. Goemans (1993) and Applegate et al.(1994) gave an overview of the various inequalities. Specially, Goemans considers the quality of those inequalities with respect to their induced relaxations.

2.2 Alternating Traveling Salesman Problem

There are some other alternates of the traveling salesman problem. Let us consider the bipartite TSP as a simple but non-trivial class of s of alternating traveling salesman problem. Originally arising from applications involving pick and place robots, the following variant of the famous traveling salesman problem is of independent interest.

Given a set of item types, and a set of locations where items must be brought to by a robot. Each location must be equipped by one item of a specified type. Several locations may require the same type of items and the items are stored in depots such that each item belongs to each type. Here the goal is the finding a shortest tour that visits locations by item types in an alternating fashion in order to equip the printed circuit boards while the edge weights are given by Euclidean distances. In fact, the problem is configuration of two different sets that can be solved with combining an assignment problem with a traveling salesman problem. Bipartite comes from partitioning of the problem to the separated sets. A straightforward reduction to the Euclidean TSP indicates that the bipartite variant of TSP is not easy compared to original TSP. Hence, the bipartite TSP cannot be solved in polynomial time, unless $P = NP$. Thus we are interested in good approximations for this problem.

Approximating the bipartite TSP is too complex. There is no constant factor approximation algorithm in general. Moreover, because of the bipartite analogue of the triangle inequality, i.e. the distances obey the square inequality, this alternate of the TSP is at least as hard to approximate as the original TSP with triangle inequality. It should be considered that good approximation algorithms for the Euclidean TSP are known. The best one was given by Christofides in 1976. Christofides algorithm obtains a locally optimal minimum that is $\frac{3n-1}{2n}$ times longer than the optimal tour. Also, Arora (1996) provided a polynomial-time approximation for constructing a tour at most $1+\varepsilon$ times longer than optimal tour where $0<\varepsilon<1$. The fact is that these techniques are not suitable to produce bipartite tours directly. Anily and Hassin (1992) and Michel, Schroeter and Sirvastav (1993) observed that inserting a perfect matching into a TSP tour yields a bipartite tour with a length that is bounded by the triangle inequality to be at most $2+\varepsilon$. In 1996, Chalasani, Motwani and Rao and, independently, Frank, Korte, Triesch and Vygen in 1998, proved that there is a polynomial 2-factor approximation algorithm using spanning tree strategy for the bipartite TSP. After that, Baltz and Sirvastav (2001) gave a polynomial time approximation algorithm based on cycle cover decomposition. The study on the bipartite variant of TSP is still continued. The focus of this thesis will be in this variant of TSP because the problem arises from the assembly arm of pick and place robot is the same.

Chapter 3

Model Definition and Problem Statement

Consider a weighted complete bipartite graph , where is the union of the two n-point subsets of ^{2} and the edge weights are given by the Euclidean distances between the "Cell Points" in and the "Assembly Points" in A.

What can be said about a shortest tour that visits cell and assembly points in an alternating fashion? This is a typical problem arising in pick and place robot routing. In the other words, the problem of finding placement tours for pick and place robots that are used for the automatically placement of electronic components on printed circuit boards is of interest. A sample printed circuit board and cell together with the assembly and cell points has been given in Figure 3.1. Optimization problem here is to minimize the placement time of the robot. Since the working time of the robot is proportional to the distance trav[illegible] problem [illegible] a combination of traveling salesman proble[illegible] oblem so we have an Euclidean bipartite traveling salesman pr[illegible]

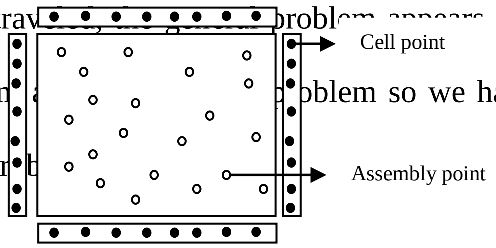


Figure 3.1: Printed Circuit Board

3.1 The Printed Circuit Board Assembly Problem

We are given:

- components which we call cells. In real world, cells are geometrical objects like boxes containing components. Let be the set of cells.
- A finite set of points in the plane or in the space called cell-point locations.
- A set of *m* positions in the plane or in the space called assembly points. Let be the set of assembly points.

For simplicity we give the labels for each component and each assembly point in such a way that the -labeled position corresponds exactly to the locations on a printed circuit board on which the -labeled component must be placed. A placement tour of the robot is defined as follows: the robot travels from a starting point to some non-empty cell like , picks an -labeled component, travels to the -labeled position, places the picked component on this position, travels to some non-empty cell, and continues in this fashion until all components have been placed. Therefore, we have to determine simultaneously a placement tour such that the total working time is minimal.

For the theoretical analysis we consider the standard model where the working time of the robot is assumed to be proportional to the distances traveled. The fact that a placement tour must be alternating between cell points and assembly points seems to be the main difficulty in finding good algorithm. The mentioned problem is a special bipartite TSP.

Definition 3.1: *A bipartite graph is an undirected graph* $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ *in which* $\boldsymbol{V}$ *can be partitioned into* V_1 *and* V_2 *such that* $(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{E}$ *implies either* $\boldsymbol{u} \in \boldsymbol{V_1}$ *and* $\boldsymbol{v} \in \boldsymbol{V}_2$ *or* $\boldsymbol{u} \in V_2$ *and* $\boldsymbol{v} \in V_1$. *That is all edges go between two sets* V_1 *and* V_2.

In the above model, technological features such as robot arm acceleration and insertion/picking time have been suppressed. In addition, all states of assembly assignments are assumed to be feasible. Note that even under these assumptions the model is realistic enough for some real-world assembly robots, and it helps to understand the most complicated situations.

Since the mentioned problem is a combination of TSP and the matching problem we should consider some mathematical model and heuristic algorithms that have been developed in this direction.

3.2 Mathematical Models of TSP

In all of the formulations that are given in this section the set of cities (nodes) is defined as $V = \{1, 2, \dots, n\}$ and the variables are defined as the following:

$$x_{ij} = \begin{cases} 1 & \text{If the arc } (i,j) \text{ is an edge of the tour} \\ 0 & \text{Otherwise} \end{cases}$$

c_{ij} = **the distance between i, j or the length of arc (i, j).**

The objective function is given by:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{3.1}$$

3.2.1 The Dantzig, Fulkerson and Johnson (DFJ) Formulation (1954)

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,2,\dots,n \tag{3.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,2,\dots,n \tag{3.3}$$

$$\sum_{i,j \in s} x_{ij} \le |S| - 1, \quad S \subset N \text{ such that } 2 \le |S| \le n-1 \tag{3.4}$$

This formulation shows that the problem is an integer program which consists of $n(n-1)$ variables and $2^{n-1} + n - 1$ constraints. In this formulation, constraints (3.2) and (3.3) introduce a regular assignment problem. Constraints (3.2) ensure that each city is entered from only one other city and constraints (3.3) ensure that each city is only departed to one other city. Consequently, constraints (3.2) and (3.3) ensure that there are two arcs adjacent to each vertex (city), and one is in and the other one is out. The last constraints (3.4) are the famous subtour elimination constraints and require feasible solutions to be connected. Subtour elimination constraints guarantee the exclusion of subtours in the optimal solution. A cycle length $k < n$ is called a subtour. It means that instead of having one tour, the solution can consist of two or more vertex-disjoint cycles. Subtour elimination inequalities will be explained in detail in subsection 3.3.1.

The exponential number of constraints makes it impractical to obtain the traveling salesman problem solution directly. Therefore, the usual procedure is to apply (3.2) and (3.3) constraints and append just those subtour elimination constraints which are violated. Based on DFJ formulation, many integer and mixed integer programs have been proposed and there are some variants of this formulation.

3.2.2. The Miller, Tucker and Zemlin (MTZ) Formulation

Miller et al. (1960) proposed an alternate formulation which reduced the number of subtour elimination constraints but extended the number of real variables by defining continuous variables u_i. Except for the arbitrarily chosen first city, the depot, associate with each city i a real variable u_i represents i's relative position on the tour. u_i's are referred as the sequencing variables.

u_i = i'position on the tour while $i \neq 1$.

Constraints (3.2) and (3.3) and the objective function as defined in (3.1) are retained. The subtour elimination constraints of the MTZ model are given by

$$u_i - u_j + (n-1)x_{ij} \leq n-2, \; i, j = 2,\ldots,n \tag{3.5}$$

$$1 \leq u_i \leq n-1, \; i = 2,\ldots,n \tag{3.6}$$

In this formulation the elimination of subtours from the feasible set is attained using sequencing variables. If an integer solution is not a tour, it contains a cycle C without vertex (city) 1 (starting city) and by adding the inequalities above corresponding to all arcs ij of cycle, we arrive at a contradiction. The MTZ formulation has $n^2 - n + 2$ constraints with $n(n-1)$ binary variables and $n-1$ continuous variables. Number of constraints has been decreased in this formulation with the price of increasing the number of variables. It is remarkable that the above formulation is used for computational practice, particularly in the moderate-size problems.

3.2.3 The Gavish and Graves (1978) Flow Based Formulation

Constraints (3.2) and (3.3) are retained but instead of elimination constraints, other constraints are defined based on introducing new continuous variables.

$y_{ij} =$' flow' in an arc (i, j) while $i \neq j$,

and flow based constraints are as follows:

$$y_{ij} \le (n-1)x_{ij} \quad ; i, j = 1,2,\ldots,n \tag{3.7}$$

$$\sum_{j=1}^{n} y_{1j} = n-1 \tag{3.8}$$

$$\sum_{i=1}^{n} y_{ij} - \sum_{k=1}^{n} y_{jk} = 1, \quad j = 2,\ldots,n \tag{3.9}$$

In this model city 1 is the only source while the others are sinks. Constraints (3.8) and (3.9) restrict $n-1$ units of a single commodity to flow out of city 1 and one unit to flow out of each of the remained cities. Consider that flow can only take place in an arc if it is included in the tour by virtue constraints (3.7). This formulation has $n(n+2)$ constraints and $n^2 - n$ binary variables and $n^2 - n$ continuous variables.

3.2.4 Multi-Commodity Network Model (Wong (1980) and Claus (1984))

As stated earlier, constraints (3.2) and (3.3) are retained but some continuous variables are introduced and with respect to some new constraints defined below.

$y_{ij}^{k} = \text{'flow' of commodity } k \text{ on the arc } (i,j)$

and constraints are:

$$y_{ij}^{k} \leq x_{ij} \qquad i, j, k = 1,2,\ldots,n, \qquad i \neq j \tag{3.10}$$

$$\sum_{i=1}^{n} y_{1i}^{k} = 1, \qquad k = 2,\ldots,n \tag{3.11}$$

$$\sum_{i=1}^{n} y_{i1}^{k} = 0, \qquad k = 2,\ldots,n \tag{3.12}$$

$$\sum_{i=1}^{n} y_{ik}^{k} = 1, \qquad k = 2,\ldots,n \tag{3.13}$$

$$\sum_{j=1}^{n} y_{kj}^{k} = 0, \qquad k = 2,\ldots,n \tag{3.14}$$

$$\sum_{i=1}^{n} y_{ij}^{k} - \sum_{i=1}^{n} y_{ji}^{k} = 0, \qquad j, k = 2,\ldots,n \qquad j \neq k \tag{3.15}$$

In this formulation constraints (3.10) allow flow only on an arc which is present in the tour. Constraints (3.11) avoid any commodity in city 1. Constraints (3.12) force exactly one unit of each commodity to flow in at city 1. Constraints (3.13) force exactly one unit of commodity *k* to flow in to at city *k* and constraints (3.14) avoid any of commodity *k* to flow out at city *k*. The last constraints, constraints (3.15), force 'material' balance for all commodities at each city apart from city 1 and for commodity k at city k.

The above multi-commodity network model has $n(n^2 - 2n + 6) - 3$ constraints, $n^2 - n$ binary variables and $n^2 - n$ continuous variables.

3.2.5 The Fox, Gavish, and Graves Time Staged Formulation

The next formulation exploits a relationship between traveling salesman problem and machine scheduling. Fox et al. (1980) have proposed three different time-dependent models. One of them has been presented below. In order to facilitate comparisons with the other formulation that were mentioned before, x_{ij} variables and constraints (3.2) and (3.3) are retained. We introduce zero-one integer variables as follows:

$$y_{ij}^t = \begin{cases} 1 & \text{If the salesperson travels from } i \text{ to } j \text{ at iteration } t \\ o & \text{Otherwise} \end{cases}$$

and elimination constraints are:

$$\sum_i \sum_j \sum_t y_{ij}^t = n \tag{3.16}$$

$$\sum_{\substack{j,t \\ t \geq 2}} t y_{ij}^t - \sum_{k,t} t y_{ki}^t = 1, \qquad i = 2,3,\ldots,n \tag{3.17}$$

$$x_{ij} - \sum_t y_{ij}^t = 0, \qquad i, j = 1,2,\ldots,n, \qquad i \neq j \tag{3.18}$$

In addition the other conditions must be imposed:

$$y_{i1}^t = 0, \qquad t \neq n \tag{3.19}$$

$$y_{ij}^t = 0, \qquad t \neq 1 \tag{3.20}$$

$$y_{ij}^1 = 0, \qquad i \neq 1, i \neq j \tag{3.21}$$

Constraints (3.17) guarantee that if a city is entered at stage t it is left at next stage, i.e. $t+1$. Removing certain variables using conditions (3.19), (3.20), (3.21) forces city 1 to be left only at stage 1 and entered just at stage n. Note that in this model there is no need to place upper bounds of 1 on the variables x_{ij} , and this condition may be violated in the linear programming relaxation. The above model has $n^2 + 2n$ constraints and $n^2(n-1)(n+1)$ binary variables. Obviously, for constraints (3.18) and variables x_{ij} this model would be even more compact having only n constraints and $n^2(n-1)$ variables. It is a remarkably drawback in terms of the strength of its Linear Programming relaxation and therefore the slowness of its overall running time.

3.2.6 The Vajda Stage Dependent Model

The same variables as in the previous model and constraints (3.2), (3.3), and (3.18) are used, together with:

$$\sum_{\substack{i,t \\ i \neq j}} y_{ij}^t = 1, \qquad j = 1,2,\ldots,n \tag{3.22}$$

$$\sum_{\substack{j,t \\ j \neq i}} y_{ij}^t = 1, \qquad i = 1,2,\ldots,n \tag{3.23}$$

$$\sum_{i,\, j \neq i} y_{ij}^{t} = 1, \qquad t = 1,2,\ldots,n \tag{3.24}$$

$$\sum_{j=1}^{n} y_{1j}^{1} = 1 \tag{3.25}$$

$$\sum_{i=1}^{n} y_{i1}^{n} = 1 \tag{3.26}$$

$$\sum_{j=1}^{n} y_{ij}^{t} - \sum_{k=1}^{n} y_{ki}^{t-1} = 0, \qquad i,t = 2,3,\ldots,n \tag{3.27}$$

Constraints (3.25) forces city 1 to be left at stage 1 and constraints (3.26) causes it to be entered at stage n. The last constraints (3.27) have the same effect as constraints (3.17). This model has $2n^2 + 3$ constraints in addition of $n(n-1)(n+1)$ binary variables which again could be reduced by leaving out constraints (3.18) and variables x_{ij}.

As it can be seen in all mathematical formulations, with the exception of DFJ model, there is polynomial (in n) number of constraints. This feature makes them more attractive than the DFJ model. However, the number of constraints for big number of n (large scale problems) may still be large, and the linear programming relaxation weaker.

3.3 Polyhedral Approaches of TSP

The TSP polyhedral set is the convex hull of the characteristic vectors of all possible tours. Let $T = \{(i_1, i_2), (i_2, i_3), \dots, (i_{n-1}, i_n), (i_n, i_{n+1} = i_1)\}$ be a tour. Then:

$\{1, 2, \dots, n\} = \{i_1, i_2, i_3, \dots, i_n\}$

The characteristic vectors $\bar{X}^T$ of the tour T is a vector of the $(n^2 - n)$ −dimensional space such

$$\bar{X}^T_{kl} = \begin{cases} 1 & \text{If there is an index } t \text{ such that } k = i_t \text{ and } l = i_{t+1} \\ 0 & \text{Otherwise} \end{cases}$$

Consequently, the TSP polyhedral is: $P^{TSP} = \operatorname{conv}\{\bar{X}^T | T \text{ is a tour}\}$. If all facet defining inequalities of P^{TSP} is given then TSP is reduced to a linear programming problem. An optimal solution always occurs at an optimal extremal point.

3.3.1 Subtour Elimination Inequalities

The subtour elimination constraints (4) in the DFJ formulation are facets of TSP, and testify to the strength of the DFJ formulation. Grotschel and Padberg (1985) proved the following theorem.

Theorem 3.1: For every $S \subseteq N$ the subtour elimination constraint:

$$x(S) \le |S| - 1 \tag{3.28}$$

defines a facet of symmetric TSP for $n \geq 4$. *Additionally, if* $S = \{i, j\}$, *constraint (3.28) reduces to the upper bound facet* $x_{ij} \leq 1$.

In fact, the quality of obtained lower bound by solving the subtour elimination constraints with the LP relaxation of TSP is much better than what can be obtained from the original relaxation but it comes at a price of increasing the number of constraints. Subtour elimination constraints are completely describing the characteristic vectors of the tours but the polyhedral set of the LP relaxation of the DFJ model is strictly larger than the TSP polyhedra.

An example will be useful to understand the concepts. Suppose we are interested in finding a complete tour for 6 cities same the 6-city problem of TSP. We examined it for 6 selected cities of IRAN then for directed graph of TSP, we obtained the following solution:

$x_{12} = 1$; $x_{35} = 1$; $x_{46} = 1$; $x_{21} = 1$; $x_{53} = 1$; $x_{64} = 1$, the graphical view of the solution is given in Figure 3.2.

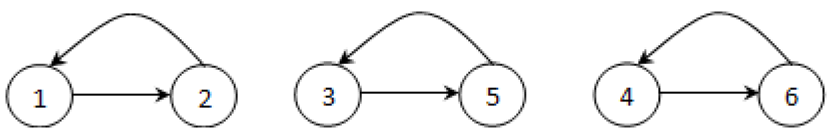


Figure 3.2: Graphical View of 6-City Problem's Solution

According to above explained constraints, we add the subtour elimination constraints as below:

$x_{12} + x_{21} \leq 1$; $x_{35} + x_{53} \leq 1$; $x_{46} + x_{64} \leq 1$.

By adding above constraints to the problem, the optimal solution will be as follows: x_{14}=1; x_{35}=1; x_{46}=1;x_{21}=1; x_{52}=1; x_{63}=1. In Figure 3.3 the graphical view of the optimal solution will be shown.

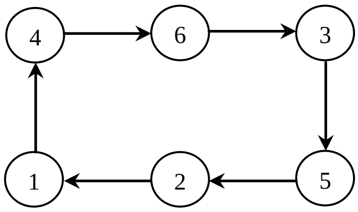


Figure 3.3: Optimal Solution of 6-City Problem

3.3.2 Comb Inequalities

A famous class of facet-inducing inequalities for TSP is the set of the comb inequalities. These inequalities were defined by Chvatal (1975) as the generalization of the 2-matching inequalities. A comb inequality consists of a *Handle* which is denoted by vertex set H and *Teeth* denoting by vertex sets $T_1, \ldots, T_s$ such that:

(i). $s \geq 3$ and is odd number;

(ii). all teeth are disjoint;

(iii). $|H \cap T_i| \geq 1$ for all i.

and the comb inequality is written as:

$$x(E(H))+\sum_{j=1}^{s}x(E(T_j))\leq|H|+\sum_{j=1}^{s}(|T_j|-1)-\frac{1}{2}(s+1). \qquad (3.29)$$

Where E is the edge set of a subset of cities and if W is a subset of edges then

$$X(W)=\sum_{u,v\in W}X_{uv}.$$

A sample comb with 3 teeth is given in Figure 3.4.

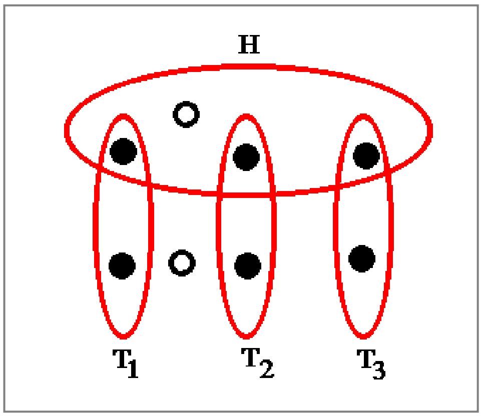


Figure 3.4: A Comb with 3 Teeth

Grotschel and Padberg (1979), introduced such structures where each tooth can have more than one vertex in common with the handle, i.e $H \cap T_i$ and T_i / H are non-empty

for every i. In 1986, Grotschel and Pullyblank introduced the *clique tree inequalities*

that are further generalization of comb inequalities in the sense that clique trees contain multiple handles, which are connected through the teeth. According to the theorem 3.2 (defined by Grotschel and Padberg (1979)) comb inequalities induce facets of STSP (Symmetric TSP) for problems having more than 5 cities.

Theorem 3.2: The comb inequalities define facets of STSP(n) for n ≥ 6.

3.4 Lower Bounds of the optimal value

The main interest of solving TSPs lies in computing good feasible tours. In practice, having some guarantee on the quality of the solutions is of interest. Such guarantees can only be given if a lower bound for the optimal value of the length of possible tour is known. Generally, lower bounds are obtained by solving relaxations of the original problem in such a way that one optimizes over some set containing all feasible solutions of the original case as a subset. Then the optimal solution of that problem gives an acceptable lower bound for the optimal value of the original problem. Different relaxations provide different lower bounds of the main problem.

In this section several fairly simple bounds are considered. For the purpose of this selection, we are mainly interested in lower bounds which can be computed fast enough to decrease the overall computational efforts and running times. These bounds are *combinatorial* in the sense that they are derived directly from the relaxations of the description of tours. The mentioned lower bounds are not to meant give a very good estimation of the achievable optimum but to give indications on the quality of the tours found by heuristic algorithms that will be explained in the next subsections.

3.4.1 The 2-Matching Relaxation

A 2-matching in a graph is a set of edges such that every node (city) is incident to exactly two of them. Every tour is a perfect 2-matching; even a collection of subtours is a 2-matching. The following formulation is a case of the 2-matching problem:

$$\min \sum_i \sum_j c_{ij} x_{ij} \tag{3.30}$$

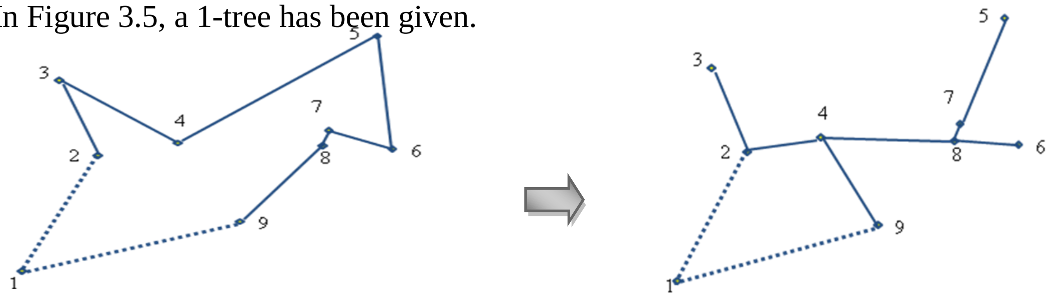
$$x(\delta(i)) = 2, \quad \text{for all } i \in N \tag{3.31}$$

$$x_{ij} = 0 \quad or \quad 1, \qquad \text{for all } i, j = 1, 2, \ldots, n \tag{3.32}$$

Note that $\delta_{(i)}$ is the set of edges incident to a node i such that the number $|\delta_{(i)}|$ is the degree of i. This problem can be solved in polynomial time based on Edmonds and Johnson (1973). The 2-matching constraints were defined to give a description of the polyhedral set of 2-matching. As it can been seen, the 2-matching problem is the DFJ model of the TSP without the subtour elimination constraints for the symmetric case of TSP.

3.4.2 The 1-Tree Bound

The fundamental of 1-tree bound for TSP is based on the following observation: If we select one city, for example city 1, then a Hamiltonian tour consists of a special spanning tree on the remaining cities in addition of two edges connecting city 1 to this tree. Hence a relaxation of TSP is obtained if it is taken as feasible solutions arbitrary spanning tree on the set $U_n : \{n \in N/\{1\}\}$ plus two edges incident to the city 1. In Figure 3.5, a 1-tree has been given.



Length =   Length =

Figure 3.5: A 1-Tree Sample

3.4.3 Geometric bounds

The geometric structure of Euclidean TSPs provides a very simple observation of lower bound for the TSP. A system of circles (disks) around cities and moats around sets of circles and moats is computed in this method. It is done in such a way that circles and moats do not overlap each other. Moreover, there has to be at least one city inside and outside of each circle and moat. Each city should be contained in a tour and each moat should be crossed at least two times. It means that the salesman must visit city 1 at some point in the tour and to do so he will need to travel at least distance *r* (radius of a circle) to arrive at the city and at least distance *r* to leave the city. It can be concluded that every tour has length at least $2 \times \sum_{i=0}^{n} r_i$. In Figure 3.6

an illustration of such a system consisting of 6 circles and 2 moats has been shown.

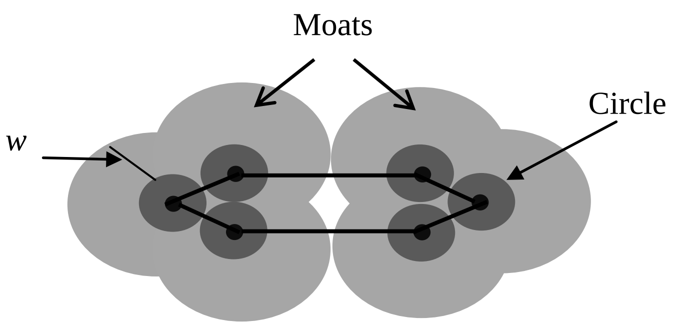


Figure 3.6: A System with 6 Circles and 2 Moats

Different systems of disks and moats are possible for a set of cities. One system can be computed using *Kruskal*'s algorithm [36]. If the radius of the disk around city i

denoted by r_i and the width of the moat around set S denoted by w_s, then the problem

of finding the best bound can be formulated as follows.

$$\max 2\sum_{i=1}^{n} r_i + 2\sum_{s} w_s \tag{3.33}$$

$$r_i + r_j + \sum_{i\in s,\, j\notin s} w_s \le c_{ij,} \quad \text{for all } i, j \in N \tag{3.34}$$

$$r_i \ge 0, \quad \text{for all } i \tag{3.35}$$

$$w_s \ge 0, \quad \text{for all } 2 \le |S| \le n-1 \tag{3.36}$$

Since we want the bound to be as large as possible, we are interested in choosing the radii so as to maximize twice their sum. Constraints (3.34) satisfy the over-lapping conditions. The formulas (3.33) – (3.36) model given by Juenger and Pulleyblank (1993) is the linear programming dual of the LP relaxation of the DFJ model. The bound can be determined in the polynomial time.

3.4.4 The Chistofides Lower Bond

Christofides heuristics method uses a minimum spanning tree as a basis for generating tours. It begins with a minimum spanning tree and gets an Eulerian graph then with some procedure obtains a Hamiltonian tour. Christofides (1976) proposed this method.

Definition 3.2: A cycle of length n in a graph on n nodes is called Hamiltonian tour.

Definition 3.3: A closed walk that traverses every edge of a graph exactly once is called Eulerian tour.

There are classes of instances in the publications of Cornuejols and Nemhauser (1978) showing that Christofides algorithm yields a tour $(3n-1)/(2n)$ times longer

than the optimal tour, thus proving that the above result can be a good lower bound. Christofides algorithm is explained below.

(i). Obtain a minimum spanning tree using Kruskal's algorithm.

(ii). Obtain a Eulerian graph by computing a minimum weight perfect matching on the odd-degree nodes of the tree and add it to the tree.

(iii). Obtain an Eulerian tour.

(iv). Obtain a Hamiltonian tour from the generated tour.

In Figure 3.7 above method has been illustrated.

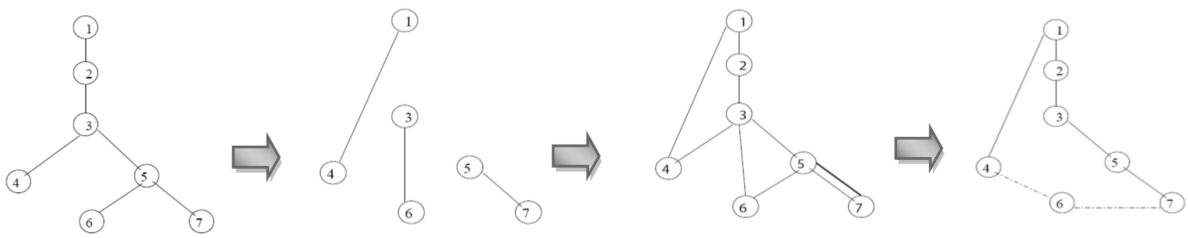


Figure 3.7: Illustration of Christofides Heuristic

3.5 Heuristic Methods

There are more heuristics than what can be discussed in this section. Only the approaches used in the analysis are explained here.

3.5.1 Nearest Neighbor Construction Heuristic

In the nearest neighbor algorithm the salesman starts at some city and goes to the nearest city of the starting city. From there the salesman visits the nearest city that was not visited so far until all cities are covered, and the salesman returns to the starting point. This procedure can be explained as follows:

(i). Choose an arbitrary city as a node l, set $S = l$ and $T = N - \{l\}$.

(ii). Until $T \neq \emptyset$ do the following:

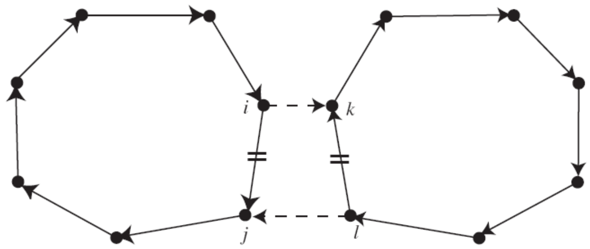
- Let $j \in T$ in such a way that $c_{li} = \min\{c_{lj} | j \in T\}$
- Connect l to j and set $T = T - \{j\}$ and let $l = j$

(iii). Connect l to the first city (choosed in step (i)) to construct a tour.

Due to Rosenkrantz, Stearns and Lewis (1977) [36] there is a proved theorem guarantee s that no constant worst case performance can be given.

3.5.2 Node and Edge Insertion Improvement Heuristic

A further intuitive method for finding the tour is to start with tours on small subsets and then extending these tours by inserting the remained nodes that is called node insertion method. Starting small subset can include one or two nodes. Using this principle results a tour containing more and more nodes of the problem until all nodes are inserted and the final complete tour is obtained. In the edge insertion algorithm an edge is removed from the tour and reinserted at the best possible position. Figure 3.8 shows a simple edge insertion process.



$\min\{c_{ik} + c_{lj} - c_{ij} - c_{lk}\}$

Figure 3.8: Edge Insertion Move

Because of endpoints of an edge, there are two possibilities for connecting the removed edge. The algorithm is given as the following.

Suppose T is the current tour. Do the following until failure is obtained.

(iv). For every node *i= 1, 2, ..., n:* Test all possibilities to insert the edge between i and its successor in the tour. If the decrease in the length of the tour is possible then select the best such edge insertion move and update *T.*

(v). If no improving movement can be found, then stop and declare the failure.

Reinelt (2001) proves that it takes time $O(n^2)$ to check if there is an improving edge insertion move at all because for every edge of the tour every possible insertion point must be checked.

3.6 Cutting Plane Methods

If c^Tbe the distance vector and if S denotes the set of the incidence vectors of all

tours, then the TSP is:

minimize $c^T x$ subject to $x \in S$. (3.37)

In order to solve the above problem, Dantzig, Fulkerson and Johnson (1954) start with the problem that they can solve as the problem below.

$$\text{minimize } c^T x \text{ subject to } Ax \leq b \qquad (3.38)$$

By suitable chosen system $Ax \leq b$ of linear inequalities satisfied by all $x \in S$,

solving problem (3.38) is what the *simplex* method is for. Problem (3.38) is a relaxation of (3.37) i.e. any feasible solution of (3.37) is a feasible solution of (3.38). Therefore the optimal value of (3.38) gives a lower bound on the optimal value of (3.37). It is a characteristic feature of the simplex method that the optimal solution is an extreme point of the polyhedron defined by the system of linear inequalities in (3.38). Optimal solution is denoted by x^*. If $x^* \notin S$ then it lies outside the convex

hull of S. In this way x^* can be separated from S by a hyperplane which is shown in

Figure 3.9.

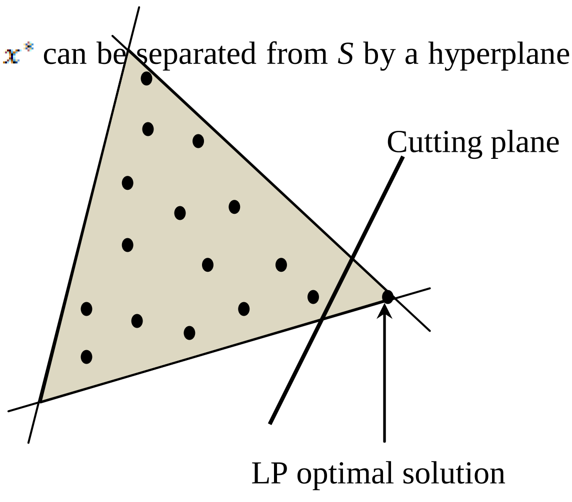


Figure3.9: Illustration of Cutting Plane

The Figure displays that there is a linear inequality which is satisfied by all the points in S and violated by optimal solution (x^*). Such an inequality is called a cutting plane. If a cut is found then it could be added to the linear inequality system in (3.38) to solve the resulting relaxation using simplex method. This process is repeated until the optimal solution of relaxation (3.38) in S is found. In the implementation of cutting plane method, the main step is finding the cuts. Based on Applegate et al. (1998), there are some ways to find the cuts. Two classical ways are:

1) Subtour eliminator cuts, and
2) Gomory cuts.

3.7 Application of Theory for medium-size Bipartite Problems

In order to manufacture some work pieces such as picking and placing some assembly parts on the PCB, a robot has to perform a sequence of operations on it. The task is to determine a sequence to perform the required operations that leads to the shortest total processing time. The robot moves between separated sets of positions in an alternating fashion. Therefore here we have the problem of finding the shortest Hamiltonian path in a bipartite graph. This problem can be treated as an alternating TSP. According to the theorem proposed by Baltz (2001), finding optimal tour of this alternating TSP in the Euclidian plane is also NP-hard.

Our robot problem, a tour for the pick and place robot starts at a depot or arbitrary starting cell point, carries an item to an appropriate location as assembly point, then moves to a depot again, and so on with this property that the robot cannot carry more than one item. It can be observed that the node sets (N) of the problems can be partitioned into two nonempty disjoint sets (C, A) which $C \cup A = N$ such that no two nodes in C and no two nodes in A are connected by an edge. Because of partitioning into two set of nodes, this problem is called bipartite TSP. Since $|C| = |A|$ and adge set of $E = \{ij \mid i \in C, j \in A\}$ then we call the graph of the problem as the complete bipartite graph. The focus of this thesis is on medium sized (up to 500) bipartite TSPs arises the robot problems.

The polyhedral structure of the Dantzig, Fulkerson and Johnson model is better understood and its linear relaxation is properly contained in the linear relaxation of some other formulations. The Miller, Trucker and Zemlin model has been also selected for developing an exact method. The reason is that DFJ formulation has an exponential number of subtour elimination constraints but MTZ formulation contains only a polynomial number of constraints especially in the first steps of solving the LP relaxation, MTZ priority is so considerable in the both sense of number of constraints and time needed.

Since in the first step of solving LP relaxation using MTZ getting to the optimal tour is very difficult for the moderate-size problems then violated constraints (constraints not satisfied by the current LP solution) should be found. In most of the cases, the violated constraint is a subtour eliminator one. Such a violated constraint can be obtained by determining the absolute minimal cut in the graph. The procedure will be discussed in Chapter 4.

As we know from the various formulations, there are many subtour elimination constraints and it is not simple to claim all of them in the LP relaxation. Using the cutting plane method, just needed constraints are added to the problem. Finding the cuts help us to get the LP solution to be a tour, and thereafter solve the TSP. However, it can be a very long procedure even in that case the LP solution a very good lower bound and also an LP solution that can serve as a guide in trying to get a good tour.

Exact methods like as MTZ model require several hours or days of running time even for moderate size instances. When running time is limited or the data of the instance is not exact, using TSP heuristics is needed. Due to find a suitable upper bound for the problem, in this work, a combination of nearest neighbor and edge insertion algorithms has been used. The complete explanation of the above methods will be done in the next chapter.

Chapter 4

Model Development

Explicitly examining all possible TSP tours is impractical even for moderately sized problems because there are different tours in (complete graph of the symmetric TSP) and Different tours in (complete digraph of the asymmetric TSP). Hence, we will not attempt to obtain the optimal solution analytically which is rather impossible.

In order to examine exact methods, it must be explained the different types of graphs and assumptions that they will be taken into account. In any TSP, there are two types of graphs in sense of the direction between two nodes. These two cases are called *directed graph* and *undirected graph*. An undirected graph consists of a finite set of vertices and a finite set of edges such that each edge has two endpoints and and is denoted by . We call such a graph undirected because we do not distinguish between the edges and . In the other words, in this graph each edge is adjacent to two vertices. While in the directed graph, we will speak about head and tail of an edge. It means that the arc is not equivalent to the arc Hence, in the directed graph we arrive into each city (or point) once and leave each city (or point) once. We denote an undirected graph as and a directed graph as

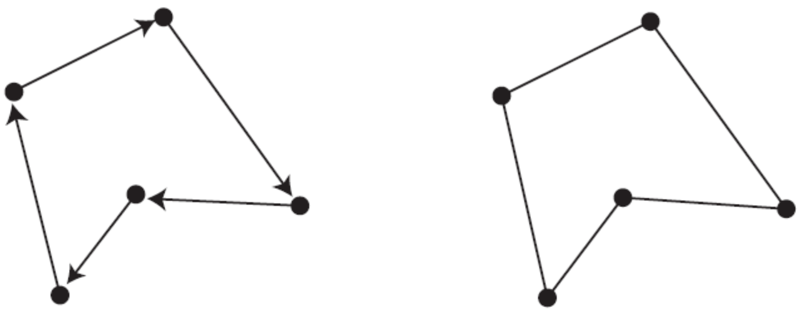
where:

(i).

(ii). $E = \{(i, j):$

(iii). $A = \{(i, j):$

Samples of directed and undirected graphs have been given in Figure 4.1.



Directed graph Undirected graph

Figure 4.1: Samples of Directed and Undirected Graph

4.1 Assumptions

Before explaining the used models and proposed iterative algorithm, the assumptions which belong to the solved problems should be addressed.

4.1.1 Symmetric TSP

In this study symmetric type of the TSP is of interest. A symmetric TSP is said to satisfy the triangle inequality if $c_{ij} \leq c_{ik} + c_{kj}$ for all distinct vertices $i, j, k \in V$.

Since we consider to the special case of the bipartite TSP, the distances obey the square inequality $c_{ij} \leq c_{ik} + c_{kl} + c_{lj}$ for all vertices $i, j, l, k \in V$.

4.1.2 Euclidean Bipartite TSP

The corresponding graph of the TSP in our problems is Euclidean bipartite TSP. An interesting special case of the TSP is to consider the optimal route passing through a collection of n points in the Euclidean plane. In fact, in the Euclidean TSP we are given n nodes (vertices) in $\mathbb{R}^2$ (more generally, in $\mathbb{R}^d$) and desire the minimum distance salesman tour for these nodes (vertices), where the distance of the edge between nodes (x_1, y_1) and (x_2, y_2) is given by d in the formula 4.1.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.1}$$

4.1.3 Edge Distance as the Weight

For the theoretical analysis, we consider distances between points where the moving time of the robot is assumed to be proportional to the distances travelled.

4.1.4 Robot Arm Limited Capacity

It is assumed that the robot cannot carry more than one item in each movement.

4.1.5 One Head Placement Robot

The placement arm of the studied robots is equipped with one hand.

4.1.6 Standard Bipartite TSP

The equal number of cells and assembly points exist on the printed circuit board.

4.1.7 Suppressed Picking/Insertion Times

Picking/insertion times have been suppressed.

4.2 Applied Exact Methods

Both of the DFJ and MTZ models have been used in the experiments.

4.2.1 The DFJ Model for the Directed Cases

Formulas (3.1), (3.2), (3.3), and (3.4) are the same. Since two versions of subtour elimination constraints have been considered we will call formula (3.4) as the version # 1 and the following constraints will give the version # 2. The idea is that if $2 \leq |S| \leq n-1$ then we must leave S.

$$\sum_{j \in S} \sum_{j \in \bar{S}} x_{ij} \geq 1. \tag{4.2}$$

4.2.2 The DFJ Model for Undirected cases

Instead of (3.2) and (3.3) we have:

$$\sum_{j=1}^{n} x_{ij} = 2, i = 1,2, \dots, n. \tag{4.3}$$

And the version # 1 of subtour elimination constraints is same to (3.4), but the version # 2 is given by:

$$\sum_{j \in S} \sum_{j \in \bar{S}} x_{ij} \geq 2. \tag{4.4}$$

The number of variables in the undirected graph is $\binom{n}{2}$.

4.2.3 The MTZ Model

We continue with the MTZ model introduced in subsection 3.1.2, wherein the number of subtour elimination constraints is reduced but the number of real variables is extended by defining continuous u_i variables. The MTZ constraints yield a

compact representation for the TSP, and their use is particularly attractive in various contexts.

The problems are solved by using both of the above formulations and the results are stored. As we expect, the solutions were not integers. To find the subtour structures *labeling technique* is used.

4.2.4 Labeling Technique

With consideration of provided information, some subtours are obtained. As stated before, some violated constraints make to get subtours in the model. To check if the provided graph from the results of LP relaxations is disconnected or not, labeling technique is used. The idea is that going through a path, every node should be given the same label. Different labels will determine the existence paths. The flow chart of proposed labeling technique has been given in Figure 4.2.

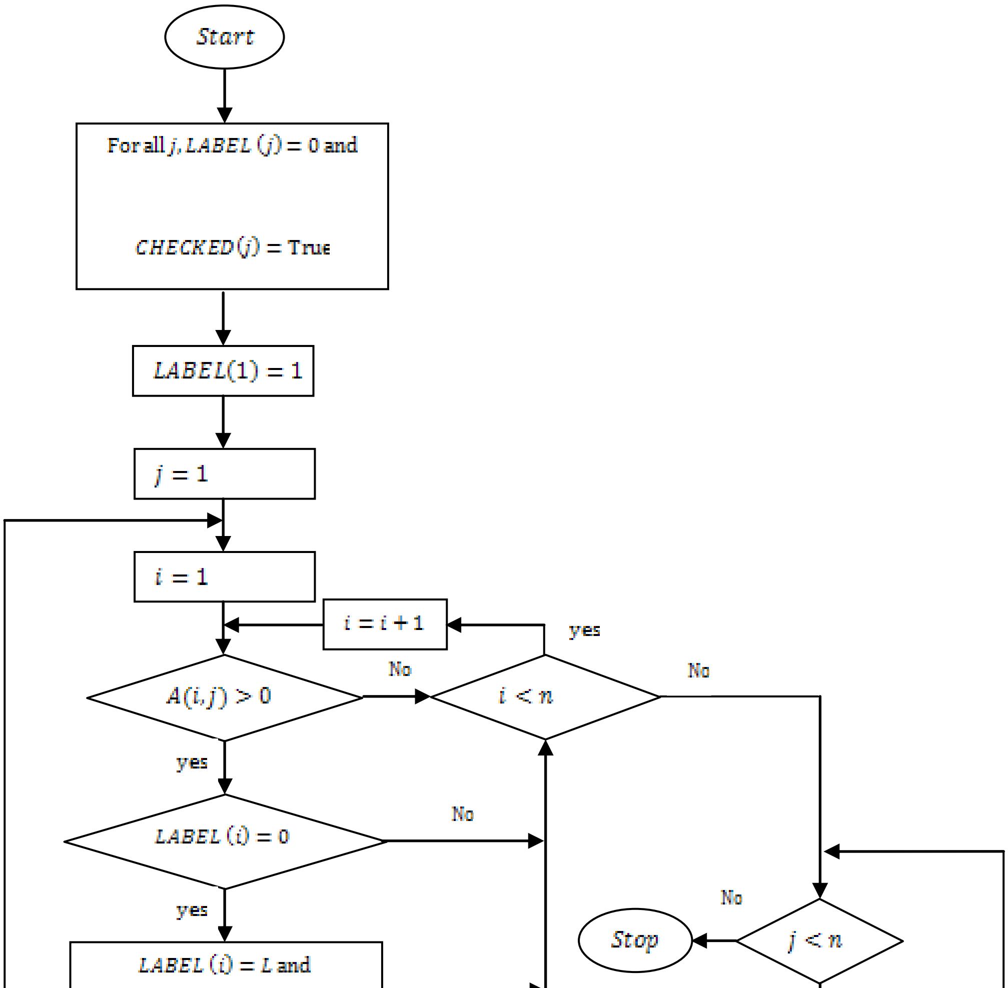


Figure 4.2: Labeling Technique Flow Chart

4.2.5 Finding Minimal Cut

Some inequalities are satisfied by the characteristic vectors of complete tours but are violated by the optimal solution of the current relaxation x^*. Such inequality is called

cut briefly. Having found cuts, one can add them to the linear inequality system of LP relaxation, solve the obtained relaxation, and iterate this process while the optimal solution x^* becomes feasible.

Definition 4.1: A cut of $G(V, E)$ is a partition of V into two sets S, T such that:

(i). $S \neq \emptyset$, and $T \neq \emptyset$.

(ii). $S \cup T = V$, and $S \cap T = \emptyset$.

The capacity of a cut (S,T) is given by:

$$c(S,T) = \sum_{u\in S, v\in T} c(u,v). \tag{4.5}$$

We can also think of a cut as a set of edges that go from S to T, i.e. all edges (u, v) such that $u \in S, v \in T$. If we remove these edges from the graph, no vertex in S will be connected to a vertex in T. The direction of the edges is important in the definition of a cut (see Figure 4.3) because we want the capacity of a cut to limit the flow going through that cut in one direction. In the Figure 4.3, set S consists of un-shaded nodes and $T = V - S$ consists of the shaded nodes. Edges between the partitions of the cut (S,T) are highlighted. The capacity of the cut is $c(s,a) + c(b,a) + c(b,t) = 13$.

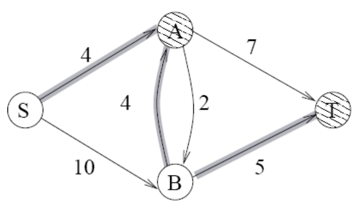


Figure 4.3: An Example of a Cut in a Graph

Considering the above information, we proposed a cutting model to find the minimum capacity cut. Most obviously, the connectivity of the bipartite graph and constructing complete tour is the minimum value of a cut. The obtained minimum cut displays the subtour eliminators that are violated. Found violated constraints are added to the LP relaxation, the LP is solved to get the new solution.

Corresponding to the last information, finding the minimal cut in a bipartite graph is introduced as follows. Constraints are related to:

1) Two non-empty parts (0 and 1)
2) Objective function

The directed cut goes from part '1' to part '0'.

$$x_u = \begin{cases} 1 & \text{if node u is in part '1'} \\ 0 & \text{if node u is in part '0'} \end{cases}$$

Constraints of two parts are:

$$\sum_u x_u \geq 1 \qquad \text{(part '1' is non} - \text{empty)}$$

(4.6)

$$\sum_u x_u \leq n - 1 \quad \text{(part '0' is non} - \text{empty)} \tag{4.7}$$

<u>Remark 4.1</u>: Partitioning into two parts '1' and '0' is based on the values of the variables obtained by LP solver such that x_u having integer values of 1 are in part '1'

and x_u having 0 values are in part '0'.

Related information for the objective function are as follows:

c_{ij} = the weight of the directed arc (i, j)

h_{ij} = the weight of the directed arc (i, j) in the directed cut

$$h_{ij} = \begin{cases} 0 & \text{if i, j are in the same part or i is in part '0' and j is in part '1'} \\ c_{ij} & \end{cases}$$

Note that c_{ij} are parameters and h_{ij} are variables in the minimal cut model.

$$\text{if } x_i = x_j$$

$$\text{if } x_i = 0, x_j = 1$$

$$h_{ij} \geq c_{ij}x_i - c\ x = \begin{cases} c_{ij} \\ 0 \end{cases} \qquad (4.8)$$

$$\text{if } x_i = 1, x_j = 0$$

From above information we will have:

$$h_{ij} \geq \begin{cases} c_{ij} & \\ 0 & \text{otherwise} \end{cases} \qquad (4.9)$$

Thus objective function will be:

$$\min \textstyle\sum_{i,j} h_{ij} \qquad (4.10)$$

Minimization problem implies that in the optimal solution of the minimal cut model h_{ij} will be:

$$\text{if } x_i = 1, x_j = 0$$

$$h_{ij} = \begin{cases} c_{ij} \\ 0 \end{cases} \tag{4.11}$$

To summarize the procedure until now, we solve an initial LP relaxation. Let x^* be

the solution. If it is integer and feasible, the optimal solution has been found, then we stop. If the solution is non-integer and infeasible then we look for one or more violated inequalities. If no violated inequalities are found then the final lower bound is recorded. When some inequalities are found we add them to the LP relaxation. We resolve the LP and again we check the solution in the sense of integrality and feasibility. One important problem here is the finding of violated inequalities. The fact is that we cannot test each one explicitly. In order to overcome this problem we use the minimal cut model. Implementing of such cutting method is called cut and branch algorithm. Our cut and branch algorithm has been shown in Algorithm 4.1.

Algorithm 4.1: Customized Cut and Branch Algorithm

Solve the LP relaxation

While the optimal solution x^* is non − integer **do** the following

If there is a cut violated by x^* **then**

Until the cut value is less than 1 **do** the following

Add the cut to the LP relaxation

Resolve LP relaxation

Find a cut violated by x^*

Else break

End

4.3 Proposed Heuristic Algorithm

Every TSP heuristic can be evaluated in terms of two key parameters: its running time and the quality of tours obtained. Because of the time and cost limitations in the manufacturing and engineering problems, we should consider to some heuristics having low running time in addition of simple structure of the method. A mixed Nearest Neighbor and Insertion constructive heuristic is proposed to obtain a good upper bound for the medium-size bipartite TSP. The proposed algorithm is a combined approach. Nearest neighbor procedure proceeds well and produces connections with short edges in the beginning but several points are forgotten during the algorithm, and they have to be inserted at high cost in the end. To avoid this problem we use an insertion algorithm during the procedure. Since the movement from one point to another point is restricted in bipartite graph, i.e. we are just allowed to go from some cell point to assembly point or vice versa, during the insertion process an edge will be inserted not a vertex (node). For applying the combined method a threshold value denoted by T is defined. Firstly, one starting cell point is selected randomly. Based on the concept of the nearest neighbor algorithm, the nearest assembly point of that point will be found. In the process of finding the next point of the tour, the distance between new point and existence points will be considered. If the distance from the endpoint is less than threshold value the nearest neighbor algorithm is applied. Otherwise edge insertion method is used. As stated before, the logic of applying edge insertion method in our problems comes out of bipartite TSP characteristics. Since any cell point can only be connected to an assembly point and any assembly point can only be connected to a cell point, the insertion process is as follows: Select a cell point and the nearest assembly point not included in the path.

New edge can be inserted to the endpoints of the path or in the middle of the path. Figure 4.4 shows such an edge insertion process.

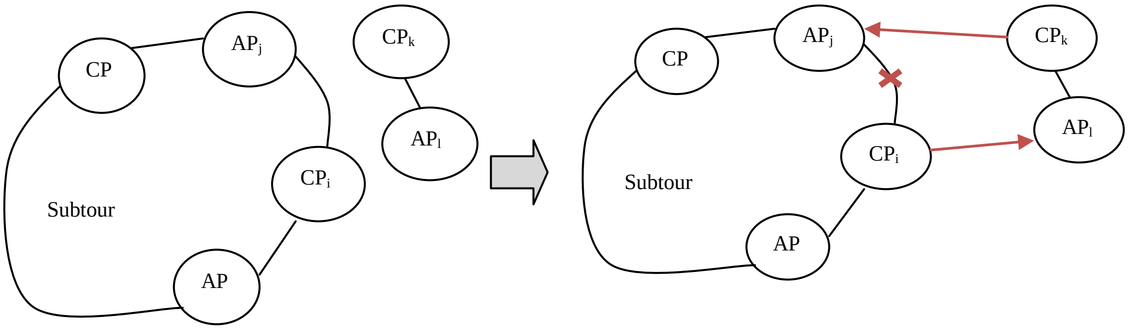


Figure 4.4: Edge Insertion Process

In Figure 4.4, the net cost (distance) is given by: $d_{CP_i AP_l} + d_{AP_l CP_k} + d_{CP_k AP_j} - d_{CP_i AP_j}$. Minimization of this net distance will be of interest.

<u>*Remark 2*:</u> In bipartite cases of the TSP, for the edge insertion process there are some different possibilities which must be considered. In the proposed heuristic all of these

possibilities are checked for each insertion process. This process is done iteratively and switching from nearest neighbor to the edge insertion is performed based on the threshold value. The procedure will be done until a complete tour is obtained. Different possibilities of insertion new edge to the current tour in the bipartite cases are shown in Figure 4.5.

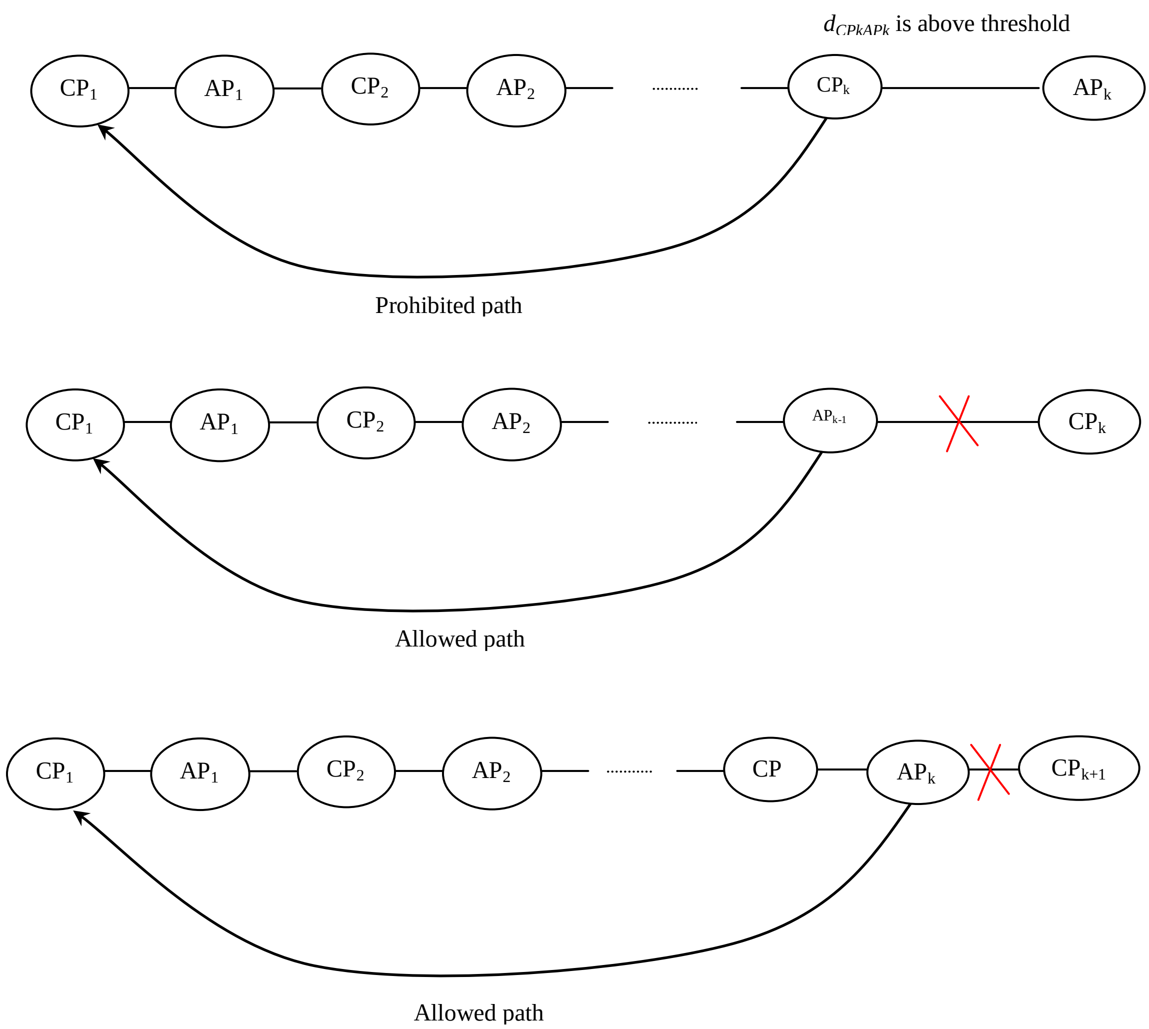


Figure 4.5: Different Insertion Possibilities in the Bipartite Graph

Algorithm 4.2, shows the proposed heuristic in detail.

Algorithm 4.2: Proposed Heuristic Algorithm

$T =$ Thereshold value

$C = \{1,2,\dots,n\}, A = \{n+1, n+2, \dots, 2n\}$ such that $C \cup A = V$

$E = \{ij | i \in C, j \in A \text{ } or \text{ } i \in A, j \in C\}$

Select an arbitrary vertex $i \in C$ and set $C \backslash \{i\}$

Let $j \in A$ such that $d_{ij} = \min \{d_{ik} | k \in A\}$

Set $A = A \backslash \{j\}$

$ST = d_{ij}$

$i_1 = i \text{ } and \text{ } i_2 = j, k = 2$

Insert i_1, i_2 to P such that $P = (i_1, i_2)$

While C and A are not empty **do** the following **until** $A = \emptyset$ and $C = \emptyset$

Find l, m such that $d_{li_1} = \min\{d_{si_1} | si_1 \in E\}$ and $d_{i_2m} = \min\{d_{i_2r} | i_2r \in E\}$

If $d_{li_1} < d_{i_2m}$ **then**

If $d_{li_1} \leq T$ **then**

Connect l to i_1 and set $P = (l, P)$

$ST = ST + d_{li_1}$

Algorithm 4.2: Proposed Heuristic Algorithm (continue)

If $l \leq n$ **then**

set $C = C \setminus \{l\}$

Else set $A = A \setminus \{l\}$

$k = k + 1$

Reindex P such that $P = (i_1, \dots, i_k)$

End

Else find e such that $e = \text{argmin}\,\{d_{el} \mid e \in C \cup A, el \in E\}$

$g = d_{el} + d_{li_1}$

$pos = 0$

$P = (i_1, i_2, \dots, i_k)$

For $t = 1$ to $k - 1$ stepsize 2

If $d_{i_t e} + d_{el} + d_{li_{t+1}} - d_{i_t i_{t+1}} < g$ **then**

$g = d_{i_t e} + d_{el} + d_{li_{t+1}} - d_{i_t i_{t+1}}$

$pos = t$

End

End

If $i_k \leq n$ **then**

If $l \geq n + 1$ **then**

$g' = d_{i_k l} + d_{le}$

set $C = C \setminus \{e\}$

set $A = A \setminus \{l\}$

Else $g' = d_{i_k e} + d_{el}$

set $C = C \setminus \{l\}$

set $A = A \setminus \{e\}$

Else If $l \leq n$ **then**

$g' = d_{i_k l} + d_{le}$

set $C = C\backslash\{l\}$

set $A = A\backslash\{e\}$

Else $g' = d_{i_k e} + d_{el}$

set $C = C\backslash\{e\}$

set $A = A\backslash\{l\}$

End

If $g' \leq g$ **then**

$g = g'$

$k = k + 2$

Reindex P such that $P = (i_1, \dots, i_k)$

End

If $pos = 0$ **then**

$P = (e, l, P)$

$ST = ST + g$

$i_1 = e$

$k = k + 2$

Reindex P such that $P = (i_1, \dots, i_k)$

Else $P = (i_1, \dots, i_t, e, l, i_{t+1}, \dots, i_k)$

$k = k + 2$

Reindex P such that $P = (i_1, \dots, i_k)$

$ST = ST + g'$

Algorithm 4.2: Proposed Heuristic Algorithm (continue)

End

End

Else

If $d_{i_k m} < T$ **then**

Connect m to i_k and set $P = (P, m)$

$ST = ST + d_{i_k m}$

If $m \leq n$ **then**

set $C = C \backslash \{m\}$

Else set $A = A \backslash \{m\}$

$k = k + 1$

Reindex P such that $P = (i_1, \dots, i_k)$

End

Else find v such that $v = \text{argmin}\,\{d_{mv} \mid v \in C \cup A, mv \in E\}$

$g = d_{i_2 m} + d_{mv}$

$pos = k$

$P = (i_1, i_2, \dots, i_k)$

For $t = k - 2$ to 1 stepsize -2

If $d_{i_t v} + d_{vm} + d_{m i_{t+1}} - d_{i_t i_{t+1}} < g$ **then**

$g = d_{i_t v} + d_{vm} + d_{m i_{t+1}} - d_{i_t i_{t+1}}$

$pos = t$

End

End

If $i_1 \leq n$ **then**

If $m \geq n + 1$ **then**

$g' = d_{m i_1} + d_{vm}$

set $C = C \backslash \{v\}$

Else $g' = d_{vi_1} + d_{mv}$

set $C = C \backslash \{\mathrm{m}\}$

set $A = A \backslash \{v\}$

Else If $m \leq n$ **then**

$g' = d_{vi_1} + d_{mv}$

set $C = C \backslash \{m\}$

set $A = A \backslash \{v\}$

Else $g' = d_{mi_1} + d_{vm}$

set $C = C \backslash \{v\}$

set $A = A \backslash \{m\}$

End

If $g' \leq g$ **then**

$g = g'$

Insert m and v to P such that $P = (P, m, v)$

$k = k + 2$

Reindex P such that $P = (i_1, \dots, i_k)$

End

If $pos = 0$ **then**

Insert m and v to P such that $P = (v, m, P)$

$i_{k+1} = m$

Algorithm 4.2: Proposed Heuristic Algorithm (continue)

$i_{k+2} = v$

$ST = ST + g$

$k = k + 2$

Else $P = (i_1, \dots, i_t, v, m, i_{t+1}, \dots, i_k)$

$k = k + 2$

Reindex P such that $P = (i_1, \dots, i_k)$

$ST = ST + g'$

End

End

End

End

Return ST and P

4.4 Proposed Iterative Algorithm for the Medium-size Bipartite TSP

Corresponding to the customized cutting model and proposed heuristic algorithm, our iterative algorithm for solving the bipartite problems is as follows:

1) Solve LP relaxation. If its integer optimal solution was found, stop. Otherwise go to step 2.
2) Prepare the graph consisting of the integer arcs.
3) Apply labeling technique to determine the paths.
4) Prepare the graph consisting of
   (i). Non-integer arcs, and
   (ii). Each path is substituted by an arc from the starting point to the end point of the path. The weight of the arc is 1.
5) Solve the minimal cut model in the generated graph.
6) If the optimal value is at least 1 then no subtour eliminator constraint is found. Switch to heuristic tour constructor.
7) If the optimal value is less than 1 then add to the node sets '1' and '0' the nodes of paths going within the set. These paths have been found in step 3.

The subtour eliminator constraint between the supplemented sets '1' and '0' is violated. Add this constraint to the problem. Go to step 1.

Chapter 5

Computational Experiments

Theoretical analysis is a useful tool in the algorithm design but empirical analysis is absolutely necessary to check the efficiency of the algorithm. We are interested in design and selection of the most efficient algorithms for real-world use and, thus, we pay more attention to experimental evaluation.

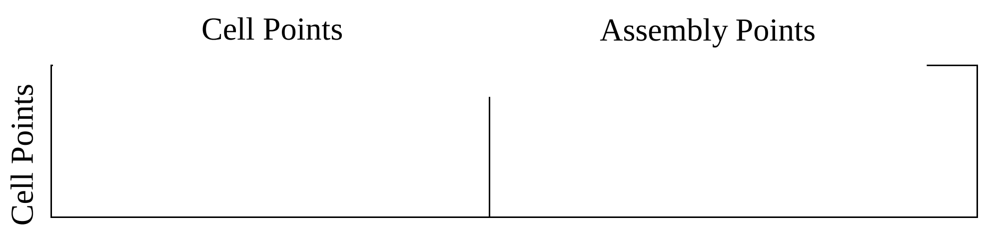
In this chapter, computational results obtained after testing the ideas mentioned in the previous chapter is described. All of the implementations have been written in the "MATLAB R2010a" and "LINGO12.0" programming languages. The computers used to run the implementations were all each with 2GB RAM, running at 2.8 GHz processor.

5.1 Modeling of the DFJ and MTZ Formulations

In order to test the exact methods, first step was writing DFJ and MTZ models in the optimization software. The solver package has been used for these experiments is extended LINGO 12.0/ win 32 (LINDO system 2010). To execute the models in LINGO environment the problems were translated into LINGO language. Basically, LINGO uses branch and bound algorithm for solving the problems.

To prevent the unnecessary long computational times in the first steps, analyzing the corresponding results of DFJ and MTZ models were begun with small sizes of TSP for the cities of Iran. These experiments consist of 6, 10, 15, 20 cities respectively. After them, eleven PCB problems addressed by P-80-1, P-80-2, P-100-1, P-100-2, P-25-15-1, P-25-25-2, P-25-15-3, P-200-1, P-200-2, P-240-1, and P-240-2 have been performed. LINGO models of DFJ and MTZ formulations are available in APPENDIX B. For the simplicity of the models, the variables have been assumed real variables between 0 and 1. In order to avoid time-consuming tasks in LINGO, input matrices for medium sized problems (PCB problems) is read from Microsoft Excel. For example, the input matrix of the problem P-25-15-1 contains 6400 elements. Typing 6400 elements in LINGO is approximately impossible. Also, a solution generated by LINGO is of little use if it could not to be exported to other applications. For these reasons, interfacing with spreadsheets of LINGO is used to move information in and out of LINGO. Input data can be found in APPENDIX A, Tables A_2 TO A_{16}. The distance matrices have been calculated using Euclidean distance, as stated before, and computing these matrices has been done by MATLAB program that is given in APPENDIX C. Distance structure of the mentioned problems is shown in Figure 5.1. Note that there are no variables with indices i, j

where $1 \leq i, j \leq 80$ or $81 \leq i, j \leq 160$.



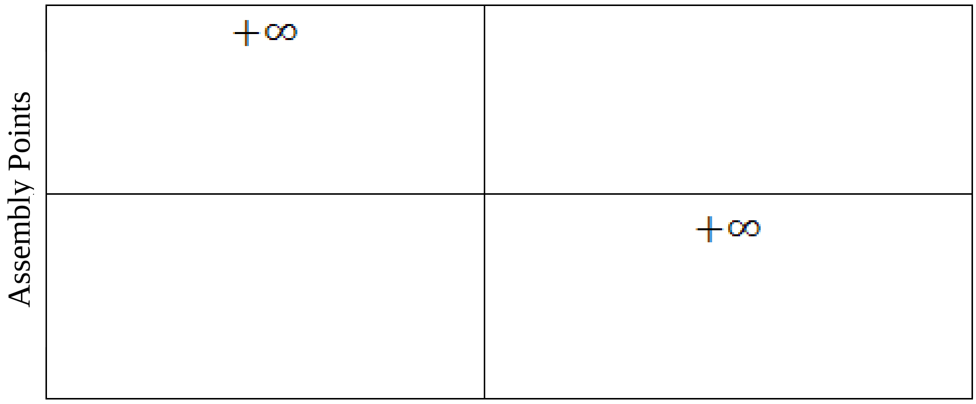


Figure 5.1: Structure of Distance Matrices of PCB Problems

5.2 Plotting the Graphs

The desirable solution of the solving TSP problems is integer solution. Because of the non-integrality in the solutions of PCB problems, it could be useful to check the plotting style of integer and non-integer arcs in the bipartite graph. To do this feature, a MATLAB code has been written in MATLAB R2010a. The program is shown in APPENDIX C. It should be noted that plotting program is based on the values of the variables resulting from LINGO such that the edges of the graph having weights between 0.69 and 1 are shown as *red* edges, and *green* lines display edges having weight between 0 and 0.69. If there be any edge with weight greater than 1 it is shown by *magenta* line in the plot. The plotting results for PCB instances for initial results of MTZ formulation have been shown in APPENDIX E, Figure E_1 to E_4.

5.3 Applying Labeling Technique

As stated before, labeling technique is applied to find the paths in the bipartite graphs in order to check the obtained graph from LINGO results is connected or not. Existence of any disconnected path in the graph somehow will be shown the existence of violated constraints in the problem. MATLAB R2010a has been used to write the proposed labeling technique. The corresponding code is given in APPENDIX C. Output of the program is transferred to spreadsheets. This spreadsheets display the label of every node in the corresponded TSP graph. Each group of the nodes having same labels constructs a path. For example, the initial obtained labels of problem 25-15-1 by this program have been given in APPENDIX A, Table A_{17}.

5.4 Solving the Minimal Cut Model

A subtour elimination constraint is violated if there is a cut in the graph with a cut value less than 1. Therefore, the absolute minimal cut must be found in the graph determined by the fractional solution. It is different from the usual problem to find the minimal cut separating two a priori given vertices. Based on the explained cutting model in the previous chapter, the non-integer arcs and also each path consisting of these non-integers as the starting point and endpoint should be found. Now the required information to write a cutting model for the problem is available. To understand the model exactly, cutting model of the problem 10-city by MTZ formulation is explained below. Using the output worksheet of LINGO for problem 10-city, fractional solution is given in Table 5.1.

Table 5.1: Initial Results of 10-City Problem with MTZ Model

Variable	Value
y(1, 3)	0.89
y(1, 6)	0.11
y(2, 5)	0.50
y(2, 8)	0.50
y(3, 6)	0.39
y(3, 7)	0.61
y(4, 2)	0.50
y(4, 5)	0.50
y(5, 10)	1.00

y(6, 3)	0.11
y(6, 7)	0.39
y(6, 9)	0.50
y(7, 1)	1.00
y(8, 2)	0.50
y(8, 9)	0.50
y(9, 6)	0.50
y(9, 8)	0.50
y(10, 4)	1.00

It should be considered that the values of other variables are zero. Results show that considerable paths with arc weights 1 are as follows.

$P_1 = \{7, 1\}$, $P_2 = \{5, 10, 4\}$. Considering P_2, we must exclude node 10 from the model.

Therefore, in the cutting model for above paths we should have:

$h_{0701} \geq x_{07} - x_{01}$; $h_{0504} \geq x_{05} - x_{04}$. Also for non-integer arcs we should have the following constraints: $h_{ij} \geq c_{ij} \times x_i - c_{ij} \times x_j$,c_{ij} in these constraints display the value of variables in LP relaxation which are gotten using LINGO model. Display model of this example in LINGO has been given in APPENDIX B.

Given two different values of the cutting model variables, we wish to partition the nodes into two non-empty sets so as to minimize the number (or total weight) of edges crossing between them. More formally, a cut (0, 1) of a graph is a partition of the nodes of that graph into two nonempty sets 'o' and '1'. An edge (u, v) crosses cut (0, 1) if u is in set '0' and v is in '1'. The subtour eliminator constraint between the supplemented sets '1' and '0' are violated if the cut value is less than 1. These constraints are added to the LP relaxation in LINGO and the LP relaxation again is solved. LINGO model after adding these constraints will be changed. The program of LP relaxation after adding initially violated constraints has been displayed in APPENDIX B. For example, in 10-city problem the memberships of the two sets are shown in Table 5.2.

Table 5.2: Results of Cutting Model for Problem 10-City

x	Membership
1	1
2	0
3	1
4	0
5	0
6	0
7	1
8	0
9	0
10	0

Since the objective function value of the model is less than 1, we have the violated constraints in LP relaxation. To find the violated constraint, determining the partitions of cut is needed. The Table above gives two sets $'1'$ and $'0'$ as follows:

$$'1' = \{1, 3, 7\}, '0' = \{2, 4, 5, 6, 8, 9, 10\}$$

In Figure 5.2, flow between these two sets can be seen.

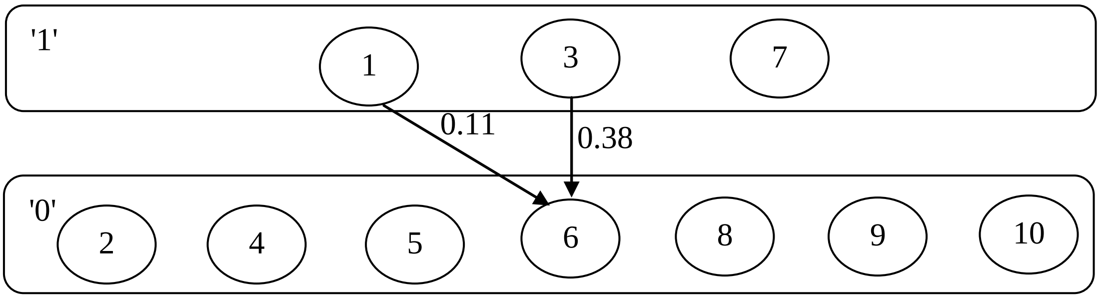


Figure 5.2: Flow Between Set '1' and Set '0'

In-flow of node 6 is 0.5 (less than 1) and we can conclude there are some violated constraints. Thus we should add the following inequalities to the LP relaxation and solve the LP once more.

$$\sum_{i\in '1'\, j\in '0'} x_{ij} \geq 1$$

Obviously, this work should be repeated until no violated subtour elimination constraint exists. Because of some difficulties, the program of cutting model has been not automated yet, so three cuts only have been performed for the medium-size problems. Outputs of the cuts for the city problems and bipartite instances have been shown in Table 5.3 and Table 5.4. Note that DFJ OFV and MTZ OFV indicate the initial objective function value of the DFJ and MTZ models for our directed problems, respectively. LB is the abbreviation of Lower Bound. Consider that in the DFJ model assignment problem constraints are included. Results indicate that there is no significant different between the results obtained by DFJ and MTZ formulations. In some problems DFJ model and in the some other problems MTZ model gives better lower bound.

Table 5.3: Results of the DFJ Model

	Problem	Size	DFJ OFV	Found LB
General TSP	6 city	6 × 6	3068.00	3718.00
	10 city	10 × 10	5675.00	5679.00
	15 city	15 ×15	7616.00	7700.45
	20 city	20 × 20	7943.00	8075.00
2-Dimensional Bipartite	80-1	40 × 40	44318.49	44328.00
	80-2	40 × 40	48382.57	48384.30
	100-1	50 × 50	63747.28	63756.90
	100-2	50 × 50	73340.72	73341.70
	25-15-1	80 × 80	67863.70	67874.90
	25-15-2	80 × 80	77306.70	77322.60
	25-15-3	80 × 80	72823.80	73458.00
	200-1	100 × 100	103319.30	104322.00
	200-2	100 × 100	112905.90	119306.50
3-D. Bipartite	240-1	120 × 120	1406.87	1407.90
	240-2	120 × 120	1374.44	1398.30

Table 5.4: Results of the MTZ Model

	Problem	Size	MTZ OFV	Found LB
General TSP	6 city	6 × 6	3073.11	3722.60
	10 city	10 × 10	5207.56	5355.44
	15 city	15 × 15	6441.14	6732.29

	20 city	20 × 20	7133.68	7237.10
2-Dimensional Bipartite	80-1	40 × 40	44317.12	44321.80
	80-2	40 × 40	48384.37	48397.46
	100-1	50 × 50	63747.93	63770.19
	100-2	50 × 50	73336.71	73341.70
	25-15-1	80 × 80	67864.12	67868.37
	25-15-2	80 × 80	77306.28	77308.74
	25-15-3	80 × 80	72823.67	73600.14
	200-1	100 × 100	103319.50	104319.23
	200-2	100 × 100	112904.40	119307.61
3-D. Bipartite	240-1	120 × 120	1406.87	1407.13
	240-2	120 × 120	1374.31	1375.55

5.5 Applying the Proposed Heuristic Algorithm

After solving the minimal cut model, if the optimal value of cut is greater than 1 then no subtour elimination constraint is found and we should switch to Heuristic tour constructor. Based on the concept of the algorithm in Chapter 4, the proposed heuristic algorithm is written as a program by MATLAB R2010a. The corresponding program is available in APPENDIX C. In this program threshold value has been calculated in some various methods. Methods used in our problems have been given in Table 5.5.

Table 5.5: Different Calculation Methods of Threshold

Method of calculation of Threshold
median of top 10%
min
max

average
min+ ((average-min)/2)
min+ ((average-min)/3)
min+ ((average-min)/4)

As mentioned in the previous chapter, there are some possibilities when we are going to insert an edge to the path in the process of constructing complete tour. We can consider only to the endpoints of the current path and insert the new edge to these endpoints. Heuristic case (1) has been constructed based on this feature. In addition of endpoints, we can also insert the new edge to the middle of the current path during the insertion process. We have denoted this heuristic as case (2). Given algorithm in the previous chapter has been constructed based on case (2) to cover all possibilities in the insertion process. The obtained results can be improved or not according to the characteristics of the problems. To cover this purpose, the heuristic algorithm has been written in both conditions. The written codes have been shown in APPENDIX C. Further, the results of the heuristic method for all our PCB instances tested for both heuristics have been given in Tables 5.6 and 5.7. Fortunately, the average Elapsed Runtime is not more than two seconds in both cases.

Remark 5.3: The best solution for each problem is marked with a (*).

Table 5.6: Heuristic Results Case (1)

	Problem	Method of calculation of Threshold	T value	Min. tour distance	Elapsed Time (sec.)
2-Dimensional Bipartite	80-1	median of top 10%	383.500	**49804***	0.201043
		min	27.893	51021	0.167665
		max	2963.07	51435	0.034562

			2		
		average	1372.437	50471	0.082217
		min+ ((average-min)/2)	700.165	50981	0.143690
		min+ ((average-min)/3)	476.074	51085	0.176550
		min+ ((average-min)/4)	364.029	50790	0.198976
	80-2	median of top 10%	426.500	51069	0.165447
		min	33.422	51021	0.153429
		max	2666.252	51435	0.030246
		average	1350.410	**50294***	0.074502
		min+ ((average-min)/2)	691.916	50981	0.139760
		min+ ((average-min)/3)	472.418	51894	0.157234
		min+ ((average-min)/4)	362.669	50790	0.164312
	100-1	median of top 10%	430.700	69769	0.251398
		min	13.342	69368	0.232419
		max	3363.073	70994	0.040361
		average	1683.267	71677	0.126540
		min+ ((average-min)/2)	848.304	**68146***	0.209060
		min+ ((average-min)/3)	569.983	71587	0.232670
		min+ ((average-min)/4)	430.823	69769	0.264291
	100-2	median of top 10%	508.300	**75949***	0.276789
		min	60.959	78061	0.254490
		max	3402.014	80284	0.043345
		average	1657.04	79264	0.094864

			4		
		min+ ((average-min)/2)	859.002	78999	0.257753
		min+ ((average-min)/3)	592.987	**75949***	0.258766
		min+ ((average-min)/4)	459.980	77035	0.274460
	25-15-1	median of top 10%	69.400	**72722***	0.845689
		min	23.431	75310	0.809081
		max	2838.24 0	75555	0.122093
		average	1333.15 2	75183	0.301803
		min+ ((average-min)/2)	678.291	75725	0.650276
		min+ ((average-min)/3)	460.004	74255	0.670271
		min+ ((average-min)/4)	350.861	76348	0.762123
	25-15-2	median of top 10%	84.150	84816	0.919021
		min	43.278	85587	0.859799
		max	2883.03 2	86243	0.101080
		average	1323.56 6	85559	0.238471
		min+ ((average-min)/2)	683.422	85130	0.613563
		min+ ((average-min)/3)	470.041	**83371***	0.907830
		min+ ((average-min)/4)	363.350	85010	0.825121

Table 5.6: Heuristic Results Case (1)-continue

	Problem	Method of calculation of Threshold	T value	Min. tour distance	Elapsed Time (sec.)
2-Dimensional **Bipartite**	25-15-3	median of top 10%	97.850	82309	0.885239
		min	24.759	83276	0.923543
		max	7180.43	85932	0.122563

			2		
		average	1398.826	83020	0.253613
		min+ ((average-min)/2)	711.792	**81881**	0.724098
		min+ ((average-min)/3)	482.781	84217	0.851676
		min+ ((average-min)/4)	368.286	85943	0.752674
	200-1	median of top 10%	486.100	115580	1.431207
		min	37.162	113120	1.564957
		max	3571.562	115510	0.180534
		average	1671.499	114290	0.351556
		min+ ((average-min)/2)	854.330	112960	0.843481
		min+ ((average-min)/3)	581.941	**111450***	1.318800
		min+ ((average-min)/4)	445.746	115440	1.424837
	200-2	median of top 10%	512.500	127530	1.574342
		min	10.630	**123940***	1.716574
		max	3527.792	128530	0.191936
		average	1666.049	126650	0.419853
		min+ ((average-min)/2)	838.340	125840	1.095427
		min+ ((average-min)/3)	562.436	126090	1.721015
		min+ ((average-min)/4)	424.485	125530	1.565432
3-Dimensional Bipartite	240-1	min	1.414	1473.9	2.772190
		max	27.749	1431.4	0.339103
		average	13.668	**1429.8***	0.405182
		min+ ((average-min)/2)	7.541	1442.3	1.957426
		min+ ((average-min)/3)	5.499	1475.5	2.601191

		min+ ((average-min)/4)	4.478	1463.5	2.675432
	240-2	min	1.000	1492.7	2.534267
		max	27.749	1424.3	0.295641
		average	13.610	1408.8	0.354398
		min+ ((average-min)/2)	7.305	1436.4	2.432106
		min+ ((average-min)/3)	5.203	**1405.6***	2.438765
		min+ ((average-min)/4)	4.153	1416.4	2.820125

Table 5.7: Heuristic Results Case (2)

	Problem	Method of calculation of Threshold	T value	Min. tour distance	Elapsed Time (sec.)
2-Dimensional Bipartite	80-1	median of top 10%	383.50	48352	0.202011
		min	27.89	47905	0.183556
		max	2963.07	48521	0.049896
		average	1372.44	48152	0.093395
		min+ ((average-min)/2)	700.17	**46645***	0.165207
		min+ ((average-min)/3)	476.07	48707	0.198267
		min+ ((average-min)/4)	364.03	48352	0.204957
	80-2	median of top 10%	426.50	51311	0.185905
		min	33.42	52233	0.180424
		max	2666.25	51435	0.051556

		average	1350.41	51466	0.088100
		min+ ((average-min)/2)	691.92	**50755***	0.153471
		min+ ((average-min)/3)	472.42	51311	0.180383
		min+ ((average-min)/4)	362.67	51311	0.192855
	100-1	median of top 10%	430.70	**67815***	0.284357
		min	13.34	69891	0.269563
		max	3363.07	70994	0.066074
		average	1683.27	71477	0.133445
		min+ ((average-min)/2)	848.30	70824	0.227750
		min+ ((average-min)/3)	569.98	70194	0.255577
		min+ ((average-min)/4)	430.82	**67815***	0.294449
	100-2	median of top 10%	508.30	**76776***	0.303430
		min	60.96	79832	0.271011
		max	3402.01	80284	0.066486
		average	1657.04	80204	0.126524
		min+ ((average-min)/2)	859.00	80814	0.273771
		min+ ((average-min)/3)	592.99	**76776***	0.276046
		min+ ((average-min)/4)	459.98	76787	0.308213
	25-15-1	median of top 10%	69.40	73652	0.879720
		min	23.43	75660	0.833873
		max	2838.24	75556	0.134484
		average	1333.15	76153	0.331714
		min+ ((average-min)/2)	678.29	75803	0.689853
		min+ ((average-min)/3)	460.00	75304	0.709963
		min+ ((average-min)/4)	350.86	**73455***	0.792449
	25-15-2	median of top 10%	84.15	**84072***	0.944540
		min	43.28	86509	0.880275
		max	2883.03	86243	0.138416
		average	1323.57	86200	0.279699

		min+ ((average-min)/2)	683.42	86415	0.644214
		min+ ((average-min)/3)	470.04	86651	0.929565
		min+ ((average-min)/4)	363.35	84800	0.835545

Table 5.7: Heuristic Results Case (2)-continue

	Problem	Method of calculation of Threshold	T value	Min. tour distance	Elapsed Time (sec.)
2-Dimensional Bipartite	25-15-3	median of top 10%	97.85	81460	0.916372
		min	24.76	92700	0.957786
		max	7180.43	85932	0.141187
		average	1398.83	85726	0.275497
		min+ ((average-min)/2)	711.79	85000	0.732268
		min+ ((average-min)/3)	482.78	82641	0.874728
		min+ ((average-min)/4)	368.29	**80267***	0.778949
	200-1	median of top 10%	486.10	113270	1.450885
		min	37.16	113160	1.588987
		max	3571.56	115510	0.205895
		average	1671.50	114290	0.369101
		min+ ((average-min)/2)	427.19	**112530***	0.887560
		min+ ((average-min)/3)	407.56	114220	1.328641
		min+ ((average-min)/4)	397.74	113690	1.459964
	200-2	median of top 10%	512.50	**124840***	1.609653
		min	10.63	125450	1.741797
		max	3527.79	128530	0.205053

		average	1666.05	126730	0.438337
		min+ ((average-min)/2)	460.03	125590	1.128979
		min+ ((average-min)/3)	442.54	136790	1.739344
		min+ ((average-min)/4)	433.79	125090	1.583813
3-Dimensional Bipartite	240-1	min	1.41	1930	2.809650
		max	27.75	1935	0.339599
		average	13.67	**1928***	0.429750
		min+ ((average-min)/2)	7.54	1929	1.967688
		min+ ((average-min)/3)	5.50	1929	2.605713
		min+ ((average-min)/4)	4.48	1929	2.689052
	240-2	min	1.00	1866	2.557984
		max	27.75	1863	0.301114
		average	13.61	1853	0.369727
		min+ ((average-min)/2)	7.31	1849	2.443762
		min+ ((average-min)/3)	5.20	**1847***	2.450805
		min+ ((average-min)/4)	4.15	**1847***	2.828947

5.5.1 Sensitivity Analysis of the Proposed Heuristics

We have examined the average quality of each variant (case 1, case 2) for eleven sample problems. To this end we have performed each heuristic for every starting node of cell points $n = 1,2, \ldots, C.$ Table 5.8 Shows the results. Each line corresponds

to one case and gives the length of the best, resp. worst tour, the average tour length obtained, and the span between best and worst tour (i.e., worst quality – best quality).

Table 5.8: Sensitivity Analysis for the Proposed Heuristics

Heuristic	Minimum	Maximum	Average	Span
80-1				

case (1)	49804	51435	50963.83	1631
case (2)	46645	48707	48331.5	2062
80-2				
case (1)	50294	51894	51198.33	1600
case (2)	50755	52233	51511.17	1478
100-1				
case (1)	68146	71677	70527.33	3531
case (2)	67815	71477	70676	3662
100-2				
case (1)	75949	80284	78728.6	4335
case (2)	76776	80814	79584.2	4038
25-15-1				
case (1)	72722	76348	75396	3626
case (2)	73455	76153	75354.67	2698
25-15-2				
case (1)	83371	86243	85390.83	2872
case (2)	84072	86651	86136.3	2579
25-15-3				
case (1)	81881	85943	83796.86	4062
case (2)	80267	92700	85576.5	12433
200-1				
case (1)	111450	115580	114483.3	4130
case (2)	112530	115510	114023.3	2980
200-2				
case (1)	123940	128530	126695	4590
case (2)	124840	136790	128030	11950
240-1				
case (1)	1429.8	1475.5	1457.32	45.7
case (2)	1928	1935	1930	7

240-2				
case (1)	1405.6	1492.7	1435.72	87.1
case (2)	1847	1866	1858	19

The results verify that case (1) more than case (2) leads to the best results but the average quality of the tours obtained by two cases are not significantly different. Therefore, we can conclude two variants perform more or less the same. The span is considerable; the quality of the tours strongly depends on the choice of the starting node.

5.5.2 CPU Times for the Proposed Heuristics

CPU times for the complete set of the instances are shown in Figure 5.3. The running times for the variants do not include the time to plot the graph obtained. Time scale is based on second.

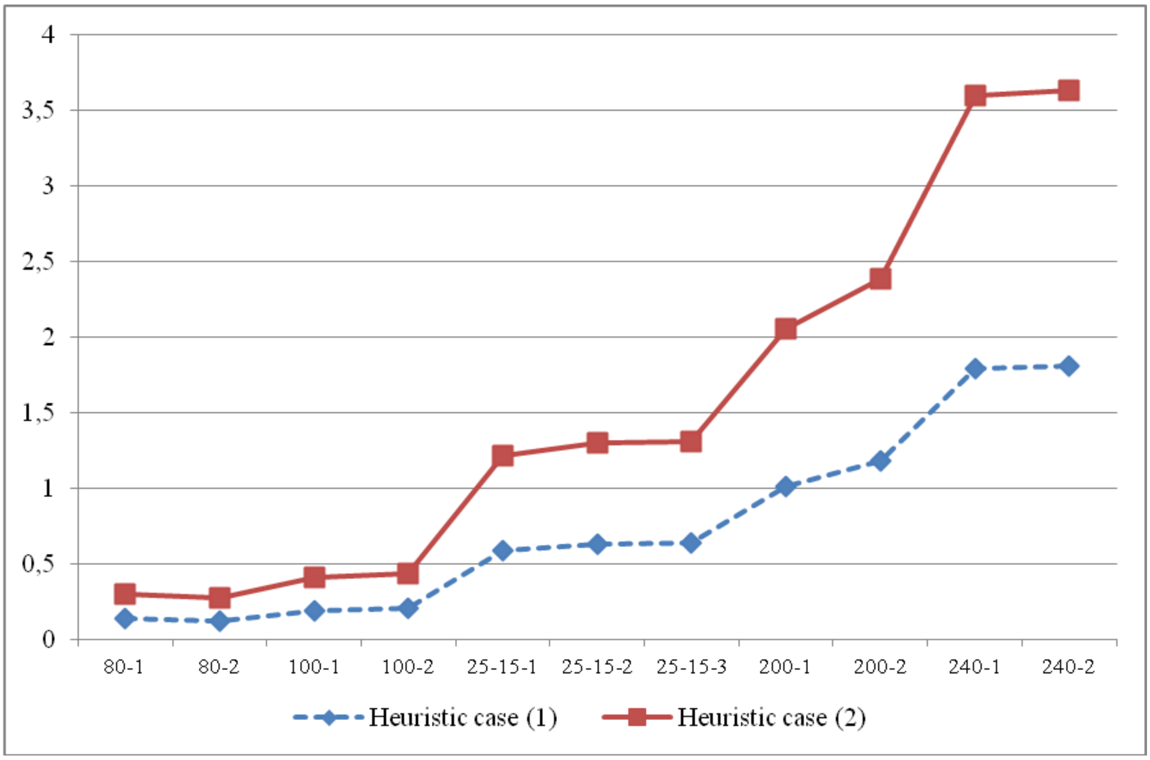


Figure 5.3: CPU Times for Two Cases of Proposed Heuristic

Figure 5.3 clearly visualizes that the time for both heuristics is highly problem-size dependent. As we expect the running time of case (2) is more than case (1) because checking the possibility of inserting to the middle points takes more times.

5.5.3 Comparison of Different Thresholds

We continue this chapter with a comparative assessment of all threshold values used for our instances. Comparison results are listed in Table 5.9. We give number of best solutions found by every threshold obtained from both heuristics mentioned.

Table 5.9: Comparison of Thresholds

Method of calculation Threshold	No. of best solutions
median of top 10%	7
min	1
max	0
average	3
min+ ((average-min)/2)	5
min+ ((average-min)/3)	6
min+ ((average-min)/4)	4

Table 5.9 shows that there is no clear winner comparing all thresholds but it is obvious that maximum distance as the threshold value cannot be a good value in our proposed heuristic. Using maximum distance somehow we would not able to escape from nearest neighbor selection process and the result will be like the nearest neighbor heuristic algorithm. Interestingly, applying the minimum distance as the threshold value is not a good value because in this way we will use insertion process

after first two nodes. It means that combination of these two heuristic can give better result.

5.6 Calculation of the Approximate Performance Ratio

We close the chapter with a relative quality by lower bounds and upper bounds discussed earlier. We now assess the performance of our iterative algorithm. Namely, we compare the best tour generated by the heuristics with the best found lower bound for the respective problem instances obtained using the MTZ and DFJ models. Qualities are computed with respect to these best found lower bounds and are given in Table 5.10. The calculating performance ratio is given by:

$$P.R. = \frac{Best\ UB - Best\ LB}{Best\ LB}$$

Due to Table 5.10 we can expect that, on the average, our proposed heuristic method can produce a solution with a certain 5.1% gap of the best found lower bound.

Table 5.10: Calculation of the Performance Ratio

Problem	Best LB	Best UP	% of P.R.
80-1	44328.00	46645.00	5.23%
80-2	48397.46	50294.00	3.92%
100-1	63770.19	67815.00	6.34%
100-2	73341.70	75949.00	3.56%
25-15-1	67874.90	72722.00	7.14%
25-15-2	77322.60	83371.00	7.82%
25-15-3	73600.14	80267.00	9.06%
200-1	104322.00	111450.00	6.83%
200-2	119307.61	123940.00	3.88%

240-1	1407.90	1429.80	1.56%
240-2	1398.30	1405.60	0.52%

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

Table A_4: Distance matrix of Problem 15-city

	ABADAN	ASTARA	ARAK	ARDABIL	URMIA	ISFAHAN	AHVAZ	BABOL	BIRJAND	TABRI
ABADAN	0	1351	704	1401	1192	868	123	1226	1889	1198
ASTARA	1351	0	766	77	604	953	1228	515	1737	296
ARAK	704	766	0	843	786	288	581	522	1606	785
ARDABIL	1401	77	834	0	527	1030	1305	592	1814	219
URMIA	1192	604	786	527	0	1074	1064	1136	2220	308
ISFAHAN	868	953	288	1030	1074	0	745	668	1173	1038
AHVAZ	123	1228	581	1305	1064	745	0	1103	1918	1075
BABOL	1226	515	522	592	1136	668	1103	0	1222	828
BIRJAND	1889	1737	1606	1814	2220	1173	1918	1222	0	1912
TABRIZ	1198	296	785	219	308	1038	1075	828	1912	0
TEHRAN	997	514	239	591	907	439	874	229	1313	599
JOLFA	1333	431	920	354	308	1173	1210	963	2047	135
CHABAHAR	2088	2475	1872	2552	2614	1584	2153	2190	1166	2560
RASHT	1162	189	577	266	739	764	1039	343	1548	485
ZANJAN	1090	454	505	377	588	757	967	548	1622	280

1	0	1351	704	1401	1192	868	123	1226	1889	1198	997	1333	2088	1162
2	1351	0	766	77	604	953	1228	515	1737	296	514	431	2475	189
3	704	766	0	843	786	288	581	522	1606	785	239	920	1872	577
4	1401	77	834	0	527	1030	1305	592	1814	219	591	354	2552	266
5	1192	604	786	527	0	1074	1064	1136	2220	308	907	308	2614	739
6	868	953	288	1030	1074	0	745	668	1173	1038	439	1173	1584	764
7	123	1228	581	1305	1064	745	0	1103	1918	1075	874	1210	2153	1039
8	1226	515	522	592	1136	668	1103	0	1222	828	229	963	2190	343
9	1889	1737	1606	1814	2220	1173	1918	1222	0	1912	1313	2047	1166	1548
10	1198	296	785	219	308	1038	1075	828	1912	0	599	135	2560	485
11	997	514	239	591	907	439	874	229	1313	599	0	734	1961	325
12	1333	431	920	354	308	1173	1210	963	2047	135	734	0	2695	620
13	2088	2475	1872	2552	2614	1584	2153	2190	1166	2560	1961	2695	0	2286
14	1162	189	577	266	739	764	1039	343	1548	485	325	620	2286	0
15	1090	454	505	377	588	757	967	548	1622	280	319	415	2280	348
16	1233	750	529	828	1143	675	1110	204	1139	835	236	970	2197	561
17	594	1438	773	1515	1559	485	659	1153	1335	1522	924	1658	1494	1249
18	1005	374	303	451	763	480	883	379	1463	455	150	590	2028	185
19	1165	1552	949	1629	1735	661	1230	1267	999	1637	1038	1772	923	1363
20	1674	1256	1187	1333	1801	1222	1768	741	481	1493	894	1628	1647	1067

Table A_5: Distance matrix between cities (ABADAN, ASTARA, ARAK, ARDABIL, URMIA, ISFAHAN, AHVAZ, BABOL, BIRJAND, TABRIZ, TEHRAN, JOLFA, CHABAHAR, RASHT, ZANJAN, SEMNAN, SHIRAZ, GHAZVIN, KERMAN, MASHHAD)

REFERENCES

[1] Wang, R.L., Zhao, L.Q. (2010). A Memory-based Ant Colony Algorithm for the Bipartite Subgraph Problem. IJCSNS International Journal of Computer

Science and Network Security, VOL.10 NO.3.

[2] Cook, W. (2009). Fifty-Plus years of Combinatorial Integer Programming: 1958-2008. *Mathematics and Statistics, Springer, Part 2,* 387-430.

[3] Oncan, T., Altinel, I.K., & Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. Computers and Operations Research 36, 637-654.

[4] Applegate, D., Bixby, R., Chvatal, V., Cook, W.(2007). The Traveling Salesman Problem. *Prinston University Press and copyrighted.*

[5] Laporte, G. (2007). A short history of the Traveling Salesman Problem. *ECCO XX Conference, Limassol, Cyprus, May 2007.*

[6] Volgenant, A., Waal, A. (2006). A heuristic for multiple-feeder PCB manufacturing. *Journal of the Operational Research Society 57*, 1134–1141.

[7] Wastlund, J. (2006). The Limit in the Mean Field Bipartite Traveling Salesman Problem. *Mathematics Subject Classification:* Primary: 60C05, 90C27, 90C35.

[8] Baltz, A., Sirvastav, A. (2005). Approximation Algorithms for the Euclidean Bipartite TSP. *Operations Research Letters,* Volume 33, Number 1.

[9] Orman, A.J., Williams, H.P. (2004). A Survey of Different Integer Programming Formulations of The Traveling Salesman problem. *First published*

in Great Britian by the Department of the Operational Research London School of Economics and Political Scienc, ISBN NO:07530-1689-3.

[10] Crama Y, van de Klundert J and Spieksma FCR (2002). Production planning problems in printed circuit board assembly. *Discrete Appliedl Mathematics* 123: 339–361.

[11] D.Sherali, H., J.Driscoll, P. (2002). On Tightening the Relaxations of Miller-Tucker- Zemlin Formulations for Asymmetric Traveling Salesman Problems. *Operations Research*, VOL. 50, NO. 4, 656-669.

[12] Monnot, J., Paschos, V., Toulouse, S. (2002). Approximation Algorithms for the Traveling Salesman Problem. *Mathematical Models of Operations Research 56*, 387-405.

[13] N.Letchford, A., Lodi, A. (2002). Polynomial-Time Seperation of Simple Comb Inequalities. *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*.

[14] Johnson, D.S., McGeoch, L.A.,. (2002). Experimental Analysis of Heuristics for STSP. *The traveling Salesman Problem and its Variations(G.Gutin And A.P. Punnen, eds.), Kluwer, Dordrecht.*

[15] Johnson, D.S., Gutin, G., McGeoch, L.A., Yeo, A., Zhang, W., and Zverovitch, A. (2002). Experimental Analysis of Heuristics for ATSP. *The traveling Salesman Problem and its Variations(G.Gutin And A.P. Punnen, eds.), Kluwer, Dordrecht.*

[16] Baltz, A., Scchoen, T., Sirvastav, A. (2001). Probabilistic Analysis of Bipartite Traveling Salesman Problem. *Mathematics Seminar, Christian-Albertchts-Universitat Zu Kiel.*

[17] Sirvastav, A., Schroeter, H., & Michel, C. (2001). Approximation Algorithms for Pick-and-Place Robots. *Annals of Operation Research 107*, 321-338.

[18] Altinkemer, K., Kazaz, B., Köksalan, M., and Moskowitz, H. (2000). Optimization of printed circuit board manufacturing: integrated modeling and algorithms. *European Journal of Operations Research 124:* 409–421.

[19] Helsgaun, K. (2000). An Effective Implementation of the Lin-Keringhan Traveling Salesman Heuristic. *European Journal of Operational Research, VOL. 126*, 106-130.

[20] Korte, B., Vygen, J. (2000). Combinatorial Optimization: Theory and Algorithm. *Third Edition. Springer-Verlag Berlin Heidelberg New York*, ISBN: 0937-5511.

[21] Karger, D. (2000). Random Sampling in Cut, Flow, and Network Design Problems.

[22] Renaud, J., Boctor, F.F., Ouenniche, J., (2000). A heuristic for the pickup and delivery traveling salesman problem. *Computers and operations research, 27(9)*, 905-916.

[23] Applegate, D., Bixby, R., Chvatal, V., Cook, W. (1999). FINDING TOURS IN THE TSP. *Institute for Discrete Mathematics, Universitat Bonn.*

[24] Balas, E. (1999). New classes of efficiently solvable generalized Traveling Salesman Problems. *Annals of Operations Research 86*, 529-558.

[25] Karger, D. (1999). Minimum Cuts in Near-Linear Time. *Journal of the ACM*. Vol. 47, Issu. 1, Jan. 2000.

[26] Arora, S. (1998). Polynomial time approximation schemes for Euclidean Traveling Salesman. *Journal of the ACM 45 (5):* 753–782, DOI:10.1145/290179.290180

[27] Frank, A., Triesch, E., Korte, B., & Vygen, J. (1998). On the Bipartite Traveling Salesman Problem. http://www.or.uni-bonn.de/home/vygen/biptsp.ps

[28] Arora, S. (1997). Nearly Linear Time Approximation Schemes for Euclidean TSP and other Geometric Problems. In Proc. 38th Annu. *IEEE Sympos. Found. Comput. Sci. (FOCS 97).*

[29] Dahl, G. (1997). An Introduction to Convexity, Polyhedral theory and Combinatorial optimization. *University of Oslo, Institute of Information Press.*

[30] S. Johnson, D., A. McGeoch, L. (1997). The Traveling Salesman Problem: A Case Study in Local Optimization. *Local Search in Combinatorial Optimization*, 215-310.

[31] Aardal, K., Hoesel, S.V. (1996). Polyhedral Techniques in Combinatorial Optimization. *Statistica Nearlandica*, VOL. 50, Issu 1, 3-26.

[32] Chalasani, P., Motwani, R., & Rao, A. (1996). Approximation Algorithms for Robot Grasp and Delivery. *In Proceedings of the 2nd International Workshop on Algorithmic Fundations of Robotics*, *Toulouse*, *France.*

[33] Johnson, D., McGeoch, L., & Rothberg, E. (1996). Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound. *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms,* 341-350.

[34] Applegate, D., Bixby, R., Chvatal, V., Cook, W. (1995). FINDING CUTS IN THE TSP. *DIAMACS Technical Report 95-05, March 1995.*

[35] Frieze, A., Karp, M., Reed, B. (1995). When is the Assignment Bound Tight for the Asymmetric Traveling-Salesman Problem? *SIAM Journal on Computing 24*, 484-493.

[36] Reinelt, G. (1994). The Traveling Salesman: Computational Solutions for TSP Applications. *Springer-Verlag Berlin Heidelberg New York*, ISBN:3-540-58334-3.

[37] Crowder, H., Padberg, M.W., (1980). Solving large-scale symmetric travelling salesman problems to optimality. *Management science*, *26(5)*, 495-509.

[38] Grotschel, M., (1980). On the symmetric traveling salesman problem: Solution of a 120-city problem. *Mathematical programming study*, 12, 61-77.

[39] Lin, S., Kernighan, B.W., 1973. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, *21(2)*, 498-516.

[40] Sales and Chips. http://www.york.cuny.edu/~malk/modeling/index.html

[41] The Traveling Salesman Problem. http://rodin.wustl.edu/~kevin/dissert/node1.html

[42] A Survey on Traveling Salesman Problem. http://www.pdf-finder.com/The-Travelling-Salesman-Problem.html

[43] Traveling salesman problem. http://www.tsp.gatech.edu/index.html

[44] Traveling salesman problem. http://en.wikipedia.org/wiki/Traveling_Salesman_Problem#Metric_TSP