A Multi-start Heuristic Algorithm for the Generalized Traveling Salesman Problem

V. Cacchiani, A. E. Fernandez Muritiba and P. Toth

University of Bologna (Italy)

M. Negreiros

Universidade Estadual do Cear, Fortaleza (Brasil)

Outline

- Problem Description
- Literature Review
- Multi-start Heuristic Algorithm
- Computational Results

Equality Generalized Traveling Salesman Problem (GTSP)

$\boxed{G = (V, E)}$ undirected graph

m clusters

$\boxed{V}$ set of nodes partitioned into $V_1, \ldots, V_m$ such that $V_1 \cup \ldots \cup V_m = V$

and $V_i \cap V_j = \emptyset, \quad i \neq j$

$\boxed{E}$ set of edges with an associated cost

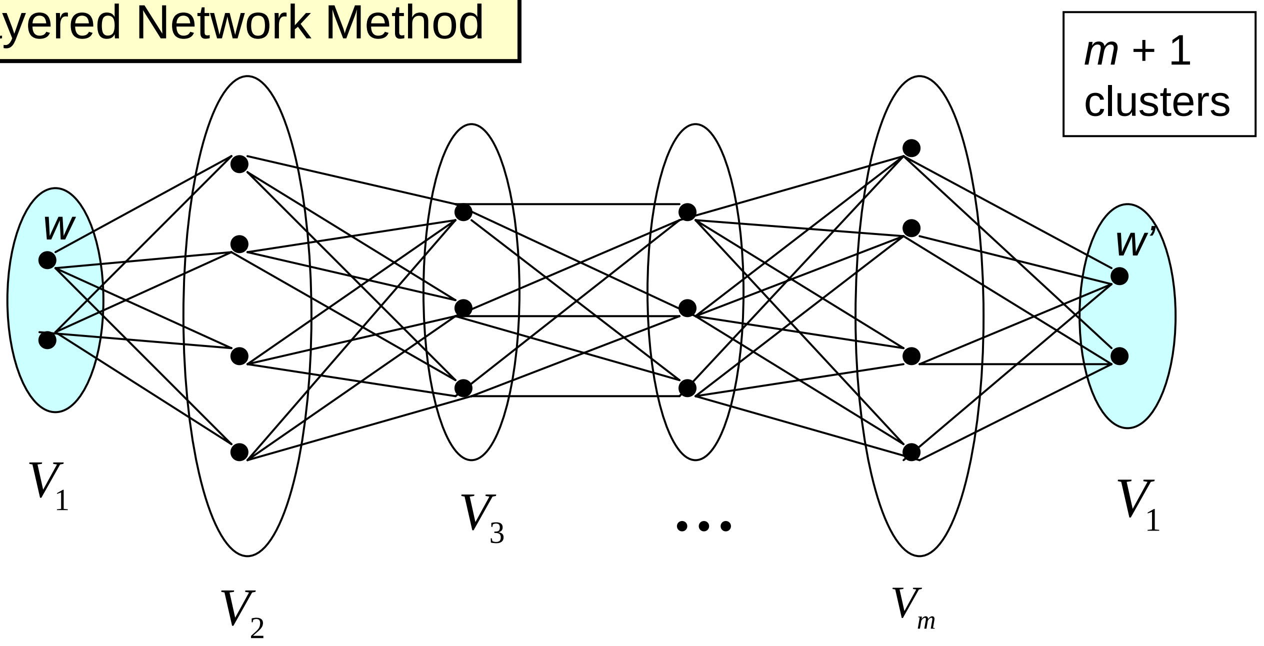
GTSP is to find an <u>elementary cycle</u> visiting exactly one node for each cluster and minimizing the sum of the costs of the traveled edges

GTSP is NP-hard

Literature Review

Authors	Date	Topic
Srivastava et al. (Cors J. 7)	1969	Dynamic programming (SGTSP)
Henry-Labordere (Rairo B2)	1969	Dynamic programming (AGTSP)
Saksena (Cors J. 8)	1970	Application in scheduling problems
Laporte et al. (Infor 21)	1983	Branch & bound (SGTSP)
Laporte et al. (Discrete Applied Math. 18)	1987	Branch & bound (AGTSP)
Noon (PhD dissertation)	1988	Nearest Neighbor heuristic algorithm
Noon and Bean (Oper. Res. 39)	1991	Lagrangian based approach (AGTSP)
Noon and Bean (Infor 31)	1993	-Transformation of the AGTSP into a Clustered TSP -Transformation of the GTSP into an asymmetric TSP
Laporte et al. (J. of Oper. Res. Soc. 47)	1996	Applications
Fischetti et al. (Networks 26)	1995	Facial structure of polytopes
Fischetti et al. (Oper. Res. 45)	1997	Branch-and-Cut, Heuristic and improvement procedures, Benchmark instances up to 442 nodes
Renaud and Boctor (EJOR 108)	1998	Composite Heuristic Algorithm GI^3.
Snyder and Daskin (EJOR 174)	2000	Random-Key Genetic Algorithm
Pintea et al. (J. of Univ. Comp. Sc. 13)	2007	Reinforcing Ant Colony System

Layered Network Method



Any path from *w* to *w'* visits exactly one node for each layer (cluster), hence it gives a feasible solution to GTSP.

Conversely, every GTSP tour visiting clusters according to sequence (*V1,…,Vm)* corresponds to a path in the Layered Network (from a certain *w* to *w'*).

⇓

The best GTSP tour visiting the clusters in the given sequence can be found by determining the shortest path from each *w* to the corresponding *w'*.

Fischetti et al. (Heuristic Algorithms and Branch&Cut)

Two heuristic algorithms:

1. It is an adaptation of the Farthest Insertion TSP procedure and is combined with two improvement procedures:
    1. 2-opt and 3-opt exchange procedures
    2. a procedure which, starting from a given sequence of clusters, computes the best feasible cycle by using a Layered Network
2. It is based on a Lagrangian relaxation of the problem, followed by the second improvement procedure, in a subgradient optimization framework.

Branch&Cut: the lower bound on the optimal solution value is obtained by solving an LP relaxation of the problem, which is tightened by adding valid inequalities.

The heuristic algorithms are applied at the root node to obtain a good upper bound.

Benchmark instances: they were obtained starting from the TSP test problems from the Reinelt TSPLIB library and by using a clustering procedure.

Renaud and Boctor (GI3)

Propose a <u>composite heuristic algorithm</u>, composed of three phases:

1. Construct a sub-cycle
2. Apply an insertion procedure in order to obtain a sub-cycle which visits exactly one node in each cluster
3. Apply a solution improvement procedure

Snyder and Daskin (GA)

Present a <u>random-key genetic algorithm</u>.

It uses reproduction, crossover and immigration operators.

The 20% of the population comes from the previous population via reproduction, the 70% is obtained by crossover and the 10% is generated by immigration.

The genetic algorithm is then combined with improvement heuristic algorithms (2-opt and swap procedures).

Pintea et al. (RACS)

Present a meta-heuristic algorithm, based on <u>Ant Colony Systems</u>.

The algorithm presents new pheromone rules.

Multi-start Heuristic Algorithm

Decomposition Algorithms (according to Renaud and Boctor classification)

The problem is subdivided into two phases.

In the first phase: the algorithm selects the nodes to be visited.

In the second phase: it constructs a cycle (by using a TSP algorithm).

Alternatively:

In the first phase: the algorithm determines the order for visiting the clusters

In the second phase: it constructs the shortest cycle by using the Layered Network Method

- It is an extension of the Decomposition Algorithms since it combines the two alternative approaches for decomposing the problem.
- It applies a preprocessing in order to delete the dominated nodes.
- It introduce randomness in order to explore a greater solution space and iteratively applies the decomposition approach.
- It applies local search procedures in order to improve the solution found.

Multi-start Heuristic Algorithm

Notation

C	*Feasible Node Set:* a node subset of *V* such that each node belongs to a different cluster
T	*Feasible Solution:* a sequence of the nodes belonging to a feasible node set
S	*Best Solution:* the best solution found so far
H	*Subgraph:* the node subgraph induced by a feasible node set

Multi-start Heuristic Algorithm

Initialization

It deletes the dominated nodes

Random Phase

It defines a feasible node set *C*

First Phase

It determines the visiting order of the clusters

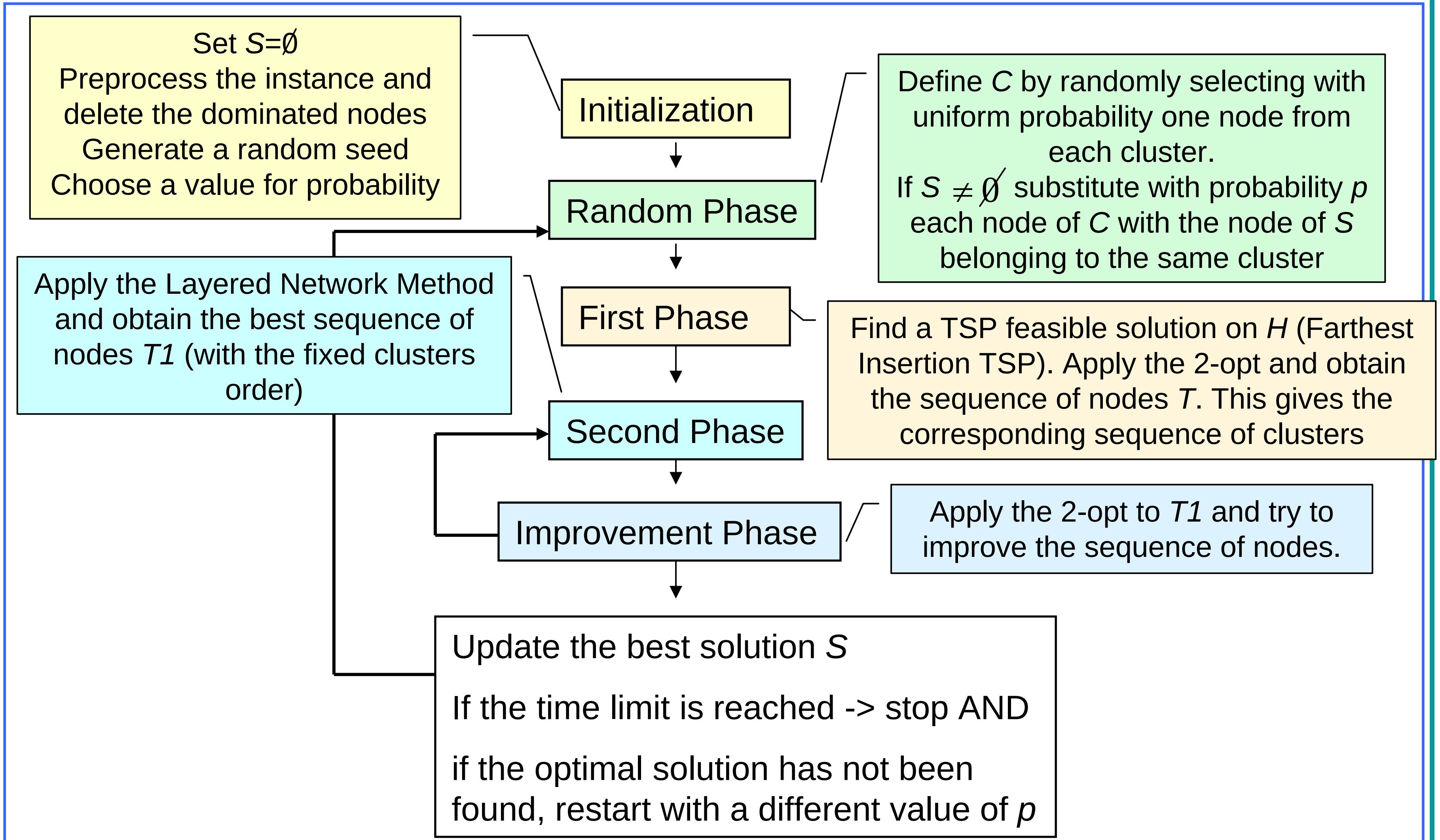
Second Phase

It finds the minimum cost cycle

Improvement Phase

It applies local search procedures

Multi-start Heuristic Algorithm



Example of the first iteration of multi-start heuristic algorithm

Instance 10att48

- It contains 10 clusters and 48 nodes
- The graph is complete
- The costs of the edges are proportional to the Euclidean distances

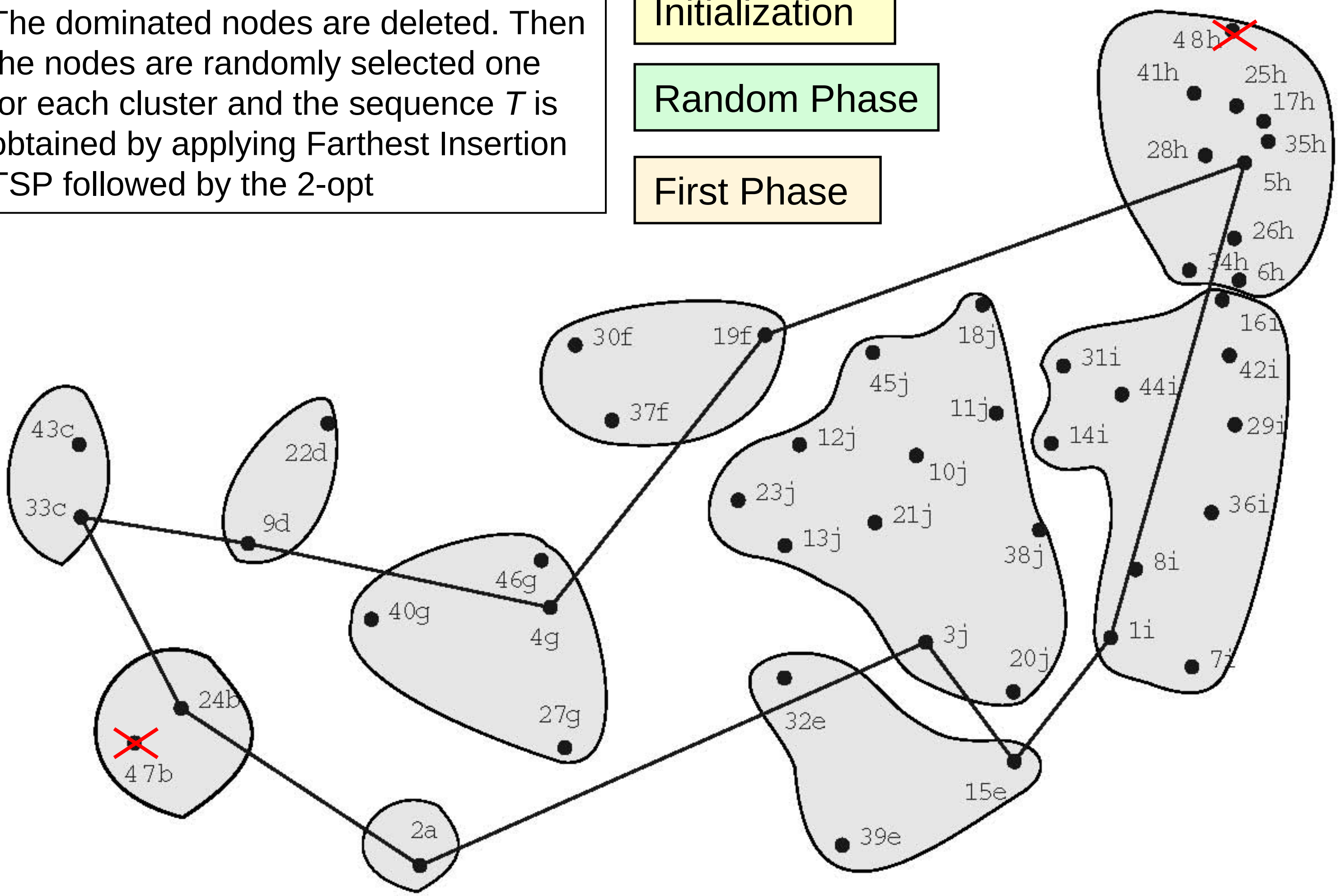
Instance 10att48

The dominated nodes are deleted. Then the nodes are randomly selected one for each cluster and the sequence *T* is obtained by applying Farthest Insertion TSP followed by the 2-opt

Initialization

Random Phase

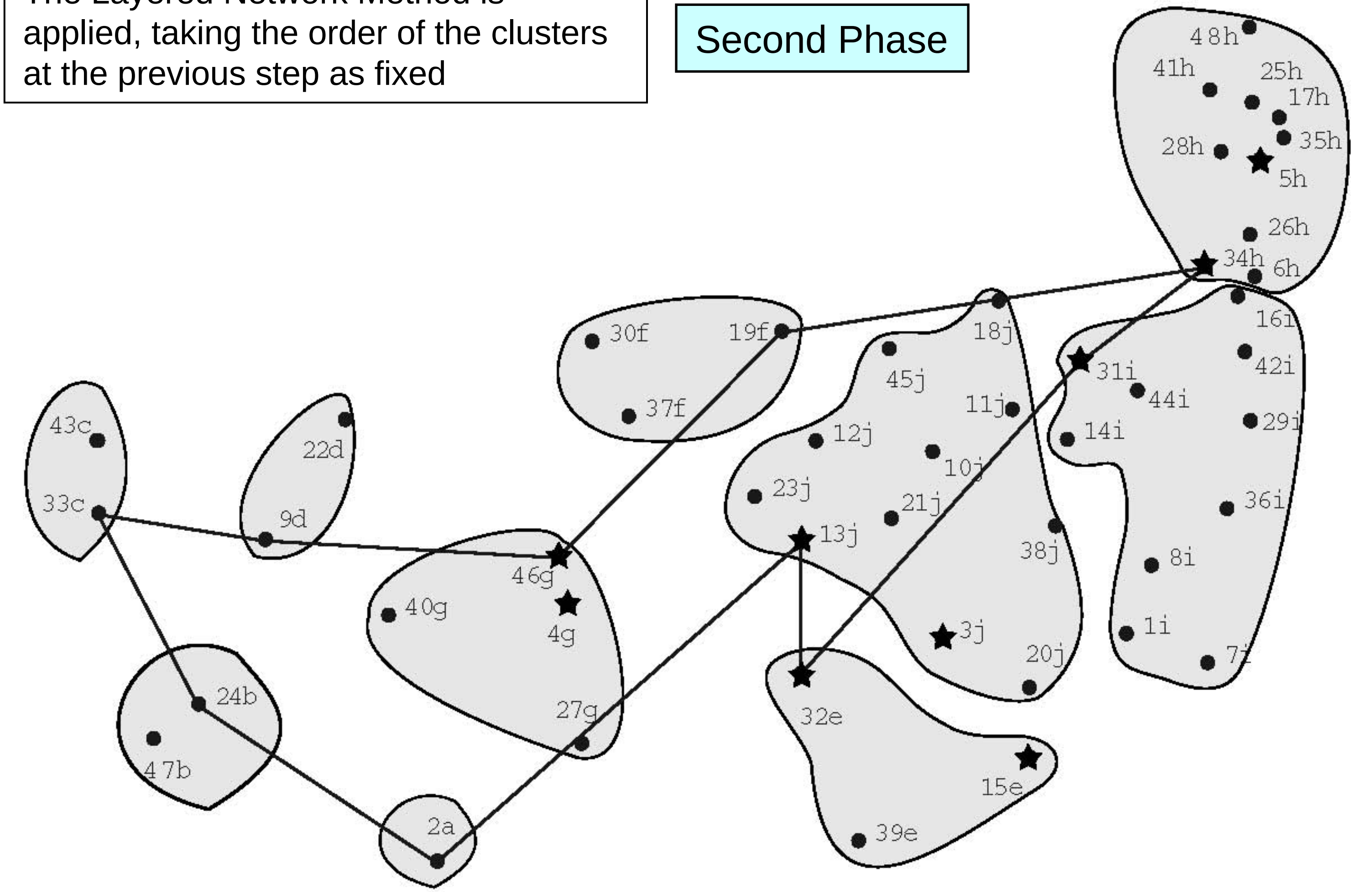
First Phase



Instance 10att48

The Layered Network Method is applied, taking the order of the clusters at the previous step as fixed

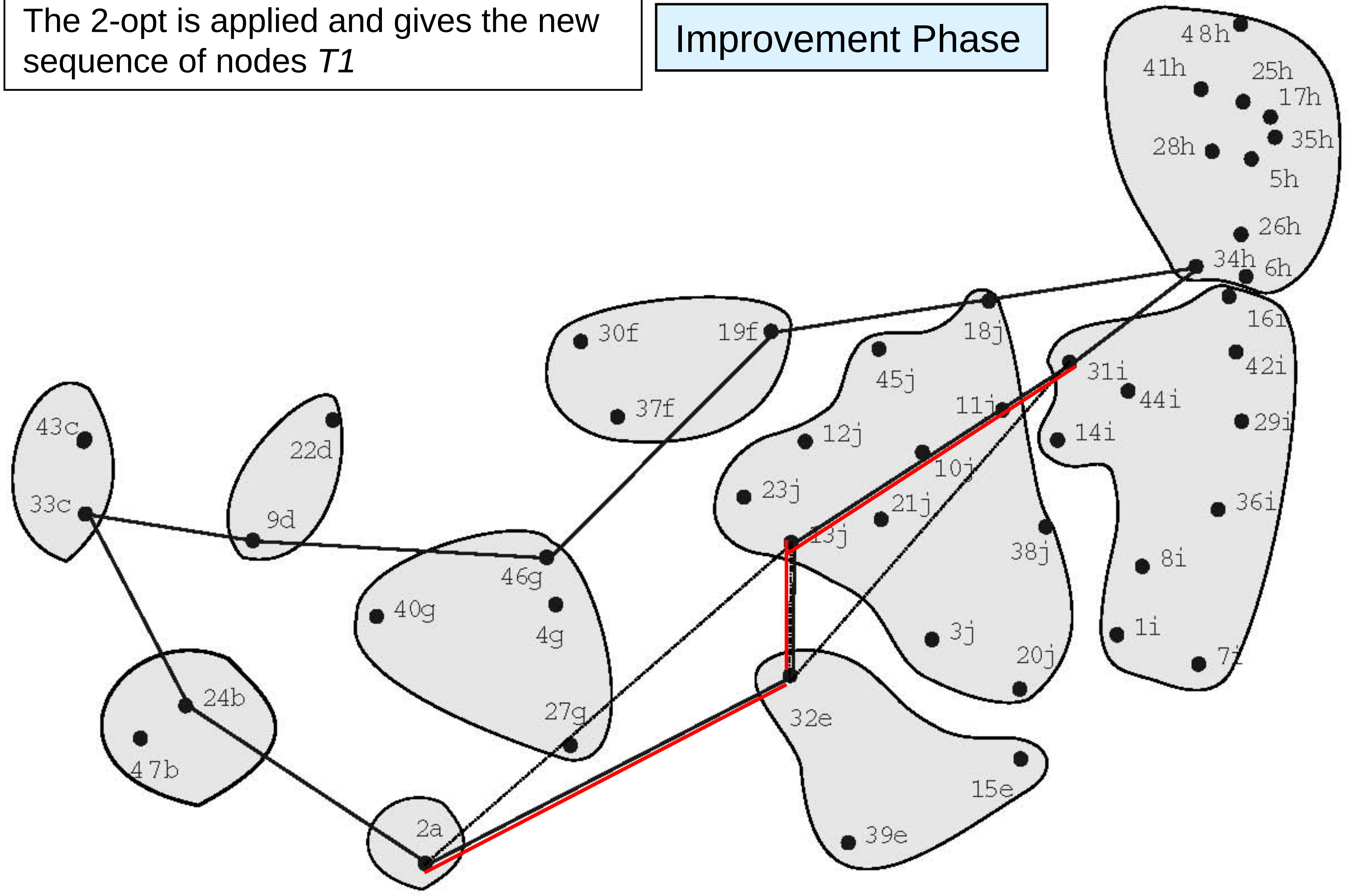
Second Phase



Instance 10att48

The 2-opt is applied and gives the new sequence of nodes *T1*

Improvement Phase

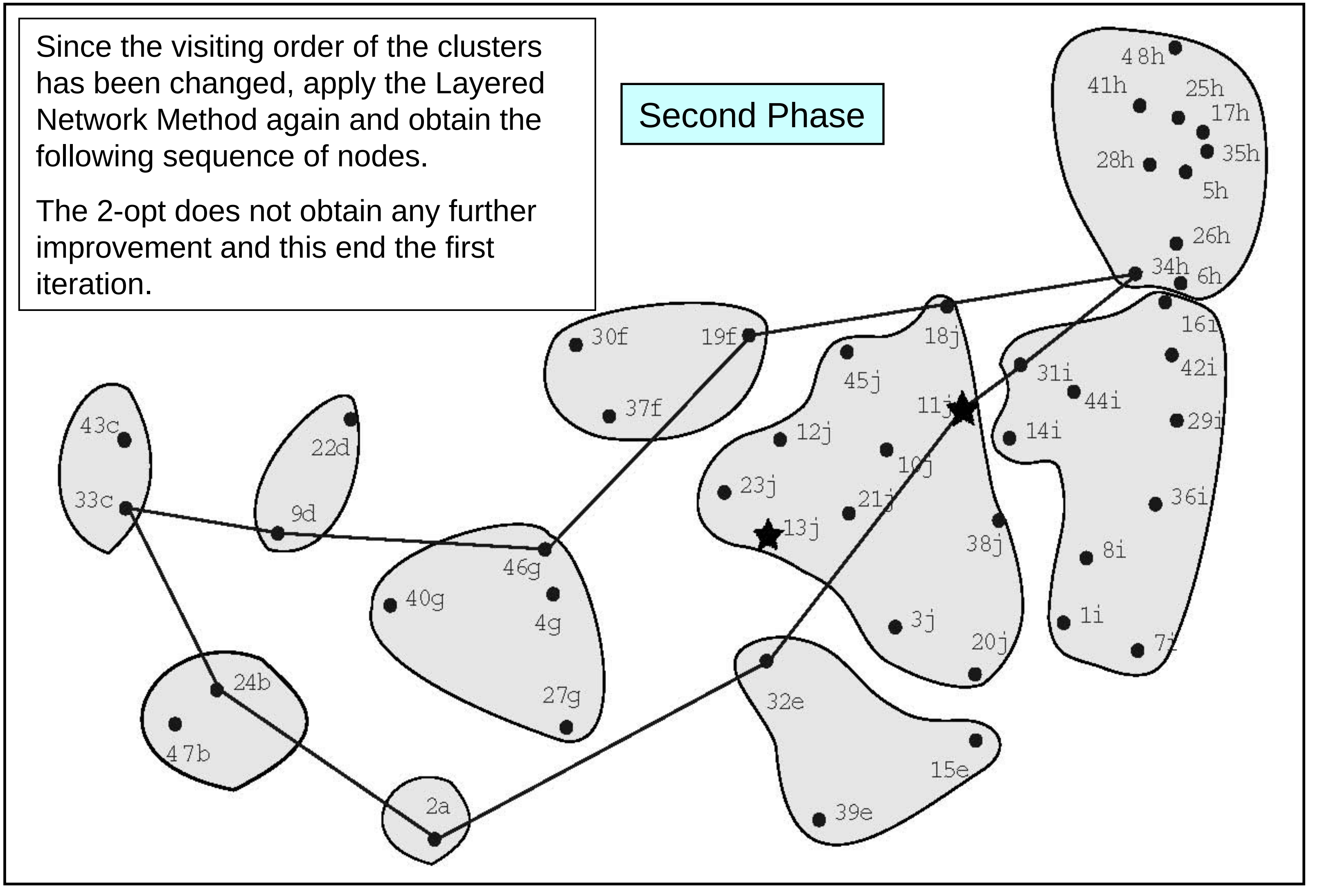


Instance 10att48

Since the visiting order of the clusters has been changed, apply the Layered Network Method again and obtain the following sequence of nodes.

The 2-opt does not obtain any further improvement and this end the first iteration.

Second Phase



Computational Results

- The presented multi-start heuristic algorithm is tested on <u>benchmark instances</u> of the literature, proposed by Fischetti et al. (1997). The instances were obtained starting from the TSP test problems from the Reinelt TSPLIB library and by using a clustering procedure.
- These instances are generally used to test heuristic or exact algorithms for the GTSP.
- The multi-start heuristic algorithm was implemented in Java and run on a pc Pentium IV, 1 Gb Ram, 3.4 Ghz.
- We considered <u>3 values for probability</u> *p*: *p*=70%, *p*=75% and *p*=80%
- We considered <u>5 seeds</u>: 1013, 171, 1181060152437, 118106285062, 1181118564906
- <u>Time limit</u> is used as stop condition (8 seconds)
- If the optimal solution is not reached within the time limit, select another value for the probability *p* and restart
- Time is expressed in <u>milliseconds</u>.

Instance	p = 70%							p = 75%							p = 80%						
	min		avg		max		opt	min		avg		max		opt	min		avg		max		opt
	gap	time	gap	time	gap	time		gap	time	gap	time	gap	time		gap	time	gap	time	gap	time	
10att48	0	2	0	5	0	18	5	0	2	0	2	0	2	5	0	2	0	5	0	18	5
11eil51	0	1	0	1	0	1	5	0	1	0	1	0	1	5	0	1	0	1	0	1	5
14st70	0	3	0	6	0	19	5	0	3	0	3	0	3	5	0	3	0	3	0	3	5
16eil76	0	5	0	5	0	5	5	0	5	0	5	0	5	5	0	5	0	5	0	5	5
16pr76	0	0	0	0	0	0	5	0	0	0	6	0	16	5	0	0	0	6	0	16	5
20kroA100	0	15	0	15	0	15	5	0	15	0	15	0	15	5	0	15	0	15	0	15	5
20kroB100	0	13	0	13	0	13	5	0	13	0	16	0	29	5	0	13	0	16	0	29	5
20kroC100	0	1	0	1	0	1	5	0	1	0	1	0	1	5	0	1	0	4	0	17	5
20kroD100	0	0	0	6	0	16	5	0	0	0	0	0	0	5	0	0	0	3	0	16	5
20kroE100	0	0	0	3	0	15	5	0	0	0	3	0	15	5	0	0	0	3	0	16	5
20rat99	0	0	0	6	0	15	5	0	0	0	6	0	16	5	0	0	0	9	0	31	5
20rd100	0	0	3	15	3	31	5	0	3	0	21	0	50	5	0	3	0	9	0	19	5
21eil101	0	0	0	9	0	19	5	0	0	0	12	0	47	5	0	0	0	9	0	31	5
21lin105	0	3	0	3	0	3	5	0	3	0	6	0	19	5	0	3	0	6	0	18	5
22pr107	0	0	0	0	0	0	5	0	0	0	0	0	0	5	0	0	0	0	0	0	5
25pr124	0	0	0	9	0	32	5	0	0	0	6	0	15	5	0	0	0	28	0	110	5
26bier127	0	3	0	6	0	19	5	0	3	0	25	0	50	5	0	3	0	15	0	18	5
28pr136	0	0	0	18	0	62	5	0	0	0	28	0	93	5	0	0	0	34	0	94	5
29pr144	0	0	0	22	0	47	5	0	16	0	31	0	47	5	0	16	0	31	0	47	5
30kroA150	0	15	0	21	0	47	5	0	0	0	18	0	47	5	0	0	0	6	0	31	5
30kroB150	0	6	0	24	0	53	5	0	6	0	9	0	21	5	0	6	0	12	0	22	5
31pr152	0	8	0	11	0	23	5	0	8	0	11	0	24	5	0	8	0	14	0	24	5
32u159	0	5	0	11	0	21	5	0	5	0	14	0	36	5	0	5	0	11	0	21	5
39rat195	0	16	0	153	0	469	5	0	0	0	18	0	31	5	0	31	0	72	0	188	5
40d198	0	47	0	118	0	281	5	0	0	0	37	0	109	5	0	16	0	53	0	125	5
40kroA200	0	0	0	43	0	125	5	0	0	0	21	0	47	5	0	16	0	72	0	110	5
40kroB200	0	63	0	125	0	219	5	0	31	0	150	0	297	5	0	62	0	231	0	375	5
45ts225	0	2422	0.035	5631	0.088	8000	3	0	4031	0.035	6743	0.088	8000	3	0	31	0.053	5024	0.088	8000	2
46pr226	0	13	0	16	0	29	5	0	13	0	19	0	29	5	0	13	0	16	0	29	5
53gil262	0	90	0.395	5049	0.691	8000	2	0	75	0.395	4881	0.691	8000	2	0	294	0.494	6468	0.691	8000	1
53pr264	0	36	0	39	0	52	5	0	37	0	49	0	68	5	0	37	0	49	0	67	5
60pr299	0	219	0	1646	0	3234	5	0	719	0	2081	0	6141	5	0	1500	0	2003	0	2766	5
64lin318	0	94	0	540	0	968	5	0	78	0	418	0	922	5	0	390	0	934	0	1531	5
80rd400	0.566	8000	1.069	8000	1.698	8000	0	0.141	8000	1.085	8000	2.327	8000	0	0.849	8000	1.541	8000	2.232	8000	0
84fl417	0	95	0	642	0	1173	5	0	220	0	954	0	2002	5	0	80	0	683	0	1580	5
88pr439	0.045	8000	0.146	8000	0.393	8000	0	0	7969	0.070	7993	0.298	8000	1	0	5959	0.030	7633	0.093	8000	1
89pcb442	1.311	8000	1.556	8000	2.069	8000	0	1.210	8000	2.119	8000	2.738	8000	0	1.367	8000	2.203	8000	2.867	8000	0

Computational Results

In order to perform a fair <u>comparison on the computing times</u> with the best state-of-the-art algorithms, we refer to <u>J.J. Dongarra</u>, "*Performance of various computers using standard linear equations software*" *(*Technical Report CS-89-85, Computer Science department, University of Tennessee, 2004) for the evaluation of the speed of the systems used in the experiments.

System	Mflop	r	Method
Gateway Profile 4MX	230	0.78	GA
Sun Sparc Station LX	4.6	0.015	GI^3, NN
HP 9000/720	2.3	0.007	FST-Lagr, FST-Root, B&C
Unknow	-	1	RACS
Our	295	1	Multi-start

Instance	Optval	MS		$RACS$	GA		GI^3		NN		$FST-lagr$		$FST-Root$		$B\&C$
		gap	time	gap	gap	time	gap	time	gap	time	gap	time	gap	time	time
10att48	5394	0	2	-	0	0	-	-	-	-	0	6	0	15	15
11eil51	174	0	1	0	0	78	0	5	0	6	0	3	0	20	20
14st70	316	0	3	0	0	156	0	26	0	12	0	8	0	51	51
16eil76	209	0	5	0	0	156	0	33	0	17	0	10	0	66	66
16pr76	64925	0	16	0	0	156	0	38	0	29	0	4	0	90	90
20kroA100	9711	0	15	0	0	312	0	102	0	57	0	17	0	128	129
20kroB100	10328	0	13	0	0	312	0	96	0	36	0	22	0	155	155
20kroC100	9554	0	1	0	0	234	0	98	0	95	0	15	0	100	101
20kroD100	9450	0	0	0	0	312	0	129	0	84	0	18	0	99	100
20kroE100	9523	0	0	0	0	624	0	101	0	42	0	6	0	90	91
20rat99	497	0	0	0	0	546	0	75	0	110	0	22	0	360	361
20rd100	3650	0	19	0	0	234	0.08	110	0.08	125	0.08	18	0	116	116
21eil101	249	0	31	0	0	156	0.40	78	0.40	45	0	12	0	179	179
21lin105	8213	0	3	0	0	234	0	216	0	56	0	14	0	113	115
22pr107	27898	0	0	0	0	312	0	131	0	78	0	15	0	51	52
25pr124	36605	0	110	0	0	468	0.43	183	0	180	0	26	0	180	181
26bier127	72418	0	18	0	0	390	5.55	542	9.68	117	0	78	0	163	165
28pr136	42750	0	16	0	0	390	1.28	188	5.54	144	0.82	50	0	300	301
29pr144	45886	0	31	0	0	234	0	245	0	177	0	16	0	56	57
30kroA150	11018	0	0	0	0	1014	0	267	0	344	0	53	0	700	702
30kroB150	12196	0	21	0	0	780	0	213	0	302	0	69	0	422	424
31pr152	51576	0	8	0.001	0	1170	0.47	264	1.80	155	0	67	0	360	664
32u159	22664	0	5	0.007	0	468	2.60	278	2.79	398	0	76	0	977	1025
39rat195	854	0	47	0	0	546	0	558	1.29	1290	1.87	57	0	1719	1721
40d198	10557	0	31	0.006	0	936	0.60	906	0.60	1782	0.48	84	0	5338	5342
40kroA200	13406	0	16	0.009	0	2106	0	446	5.25	795	0	107	0	1283	1312
40kroB200	13111	0	375	0	0	1092	0	537	0	2028	0.05	134	0	1876	1880
45ts225	68340	0	1266	0.023	0	1872	0.61	1335	0	1767	0.09	136	0.09	9089	265131
46pr226	64007	0	13	0.033	0	780	0	383	2.17	1014	0	102	0	743	748
53gil262	1013	0	8075	0.217	0.79	1482	5.03	1731	1.88	1841	3.75	111	0.89	10105	46369
53pr264	29549	0	37	0.002	0	1014	0.36	966	5.73	2208	0.33	170	0	2352	2359
60pr299	22615	0	1672	0.235	0.02	4758	2.23	135	2.01	4227	0	232	0	5680	5690
64lin318	20765	0	1531	0.121	3.50	2730	4.59	3102	4.92	4755	0.36	368	0.36	5935	11703
80rd400	6361	0	31637	0.868	1.37	2730	1.23	6053	3.98	17057	3.16	419	2.97	35221	49150
84fl417	9651	0	1580	0.574	0.07	1872	0.48	6407	1.07	20115	0.13	540	0	117001	117036
88pr439	60099	0	6114	0.785	0.23	7098	3.52	9165	4.02	18584	1.42	1026	0	37932	37960
89pcb442	21657	0	192344	0.690	1.31	7878	5.91	8516	0.22	12576	4.22	552	0.29	37477	411394
Average		0	6623.1	0.096	0.20	1233.2	0.95	1179.9	1.44	2504	0.45	126	0.12	7474.1	26025.8
# Opt		37		> 23	30		20		19		24		32		37

Average Results

		MS		$RACS$	GA		GI^3		NN		$FST-lagr$		$FST-Root$		$B\&C$
		gap	time	gap	gap	time	gap	time	gap	time	gap	time	gap	time	time
Average		0	6623.1	0.096	0.20	1233.2	0.95	1179.9	1.44	2504	0.45	126	0.12	7474.1	26025.8
# Opt		37		> 23	30		20		19		24		32		37

Conclusions

The presented multi-start heuristic algorithm finds the optimal solution for the 86.5% of the tested benchmark instances in less than 10 seconds.

For the 91.9% of the instances it finds the optimal solution for at least one trial.

The presented algorithm turns out to be competitive with the best state-of-the-art algorithms for the GTSP.