Online Companion for

"Hybrid Fiber Co-Axial Network Design"

Volume 50, Number 3, May-June 2002

Raymond A. Patterson
The University of Texas at Dallas
Richardson, TX

Erik Rolland
The A. Gary Anderson Graduate School of Management
Univrsity of California
Riverside, CA

APPENDIX A. THE BASICS OF THE ART PROCEDURE

(Adapted from Rolland, Patterson and Pirkul, 1999)

Terminology and Description of the Algorithm

An *iteration* of ART is one execution of EW and one execution of either GCSSC or EXACT, followed by solution evaluation and constraint generation. The *memory array of additional constraints* is a list containing a *time duration* (or tabu time) during which use of an arc in the CMST is prohibited (the time duration is measured in number of iterations). The *depth of learning ingrainment* is the length of commitment (in number of iterations) that the prohibition of a particular arc will endure. We set the depth of learning ingrainment initially to 40 (which is a data dependent parameter setting) and reduce the depth of learning ingrainment by 10% at the end of each phase (a *phase* is a complete cycle of our algorithm with a single depth of learning). We use ten (10) phases. A longer depth of learning ingrainment results in a longer time duration in the memory of additional constraints. The memory of the best solution is the starting memory for each phase. *Forgetting* is the removal of constraints whose time durations have lapsed. Forgetting is accelerated by shortening the existing memory ingrainment. *Shortening the existing ingrainment* reduces the length (time duration) of current memory prohibitions on the use of decision variables. In our experiments, a shortening function of 20% was used. This means that for 20% of the arcs, the length of prohibition was reduced proportionally based on a random draw from a uniform distribution in the range [0,1] at the end of each phase. The *learning rate* is the frequency with which you decide to make a decision variable constrained (prohibited). The learning rate is proportionally increasing, and is set first to 2%, then 3.6%, 6.48%, 11.66%, and finally 20.99%. Five (5) learning rate cycles are made within each phase. For each learning rate, the *learning process* continues until 15 consecutive iterations occur without an improvement to the best feasible solution. The *best feasible solution* is the lowest cost CTSD solution found to that point in the algorithm.

Intelligence is added to increase the probability that certain arcs will be constrained. In the probabilistic constraint establishment process (PCE), the learning rate varies from low to high within each phase, and only impacts whether or not a particular constraint will be added. If a constraint is to be added to prohibit the selection of a particular arc, then the time duration of the additional constraint must be selected. The time duration of a prohibition is dependent on the depth of learning ingrainment, a multiplier selected randomly from a uniform distribution [0,1], the cost of the arc, and the number of additional iterations (either 1, 2, or 5). These iteration increases were determined empirically. The allocation of additional iterations are as follows: one (1) iteration is added to the time duration of the constraint if the PCE is applied (to all arcs chosen at this iteration of ART); two (2) time steps are added to the 3% most costly arcs chosen (among all arcs chosen at this iteration of ART); five (5) additional time steps are added to the highest cost arc on the longest (costliest) CMST branch chosen at this iteration; and five (5) additional time steps are added to the highest cost chosen arc on every CMST branch. Thus, a very costly chosen arc may have 4 chances to be

prohibited: once for being a chosen arc, twice for being among the 3% of the most costly chosen arcs, three times for being the highest cost chosen arc on the longest CMST branch, and four times for being the highest cost chosen arc on this arc's particular branch. More expensive arcs are more likely to be deemed prohibited, and because the time duration is dependent upon the cost of the arc and the reason for the prohibition, their time duration is likely to be longer than for a cheaper arc.

After having completed a reasonable number of iterations, the subset of best solutions can be analyzed to determine the commonalities of their memories. The most fruitful analysis of the subset of "best memories" is to determine which decision variables the best solutions have all left unconstrained. Once the algorithm has performed 200 iterations, we prohibit additional constraints on arcs which are not constrained by all of the ten (10) best solutions found. Intuitively, after 200 iterations, the 10 best solutions contain most of the worthwhile constraints. By limiting the subsequent search to the constraints contained in the top 10 "best memories", we substantially improve the solutions by focusing our search in a limited (and fruitful) search space. As the time duration of memories lapse, decision variables become available for use. If any decision variable is unconstrained in all of the top 10 "best memories" at any point subsequent to the 200^{th} iteration, then that decision variable can no longer be considered for prohibition. In this intensification process, the memory converges to local minimums.

Figure A.1 depicts the flowchart associated with the algorithm described above.

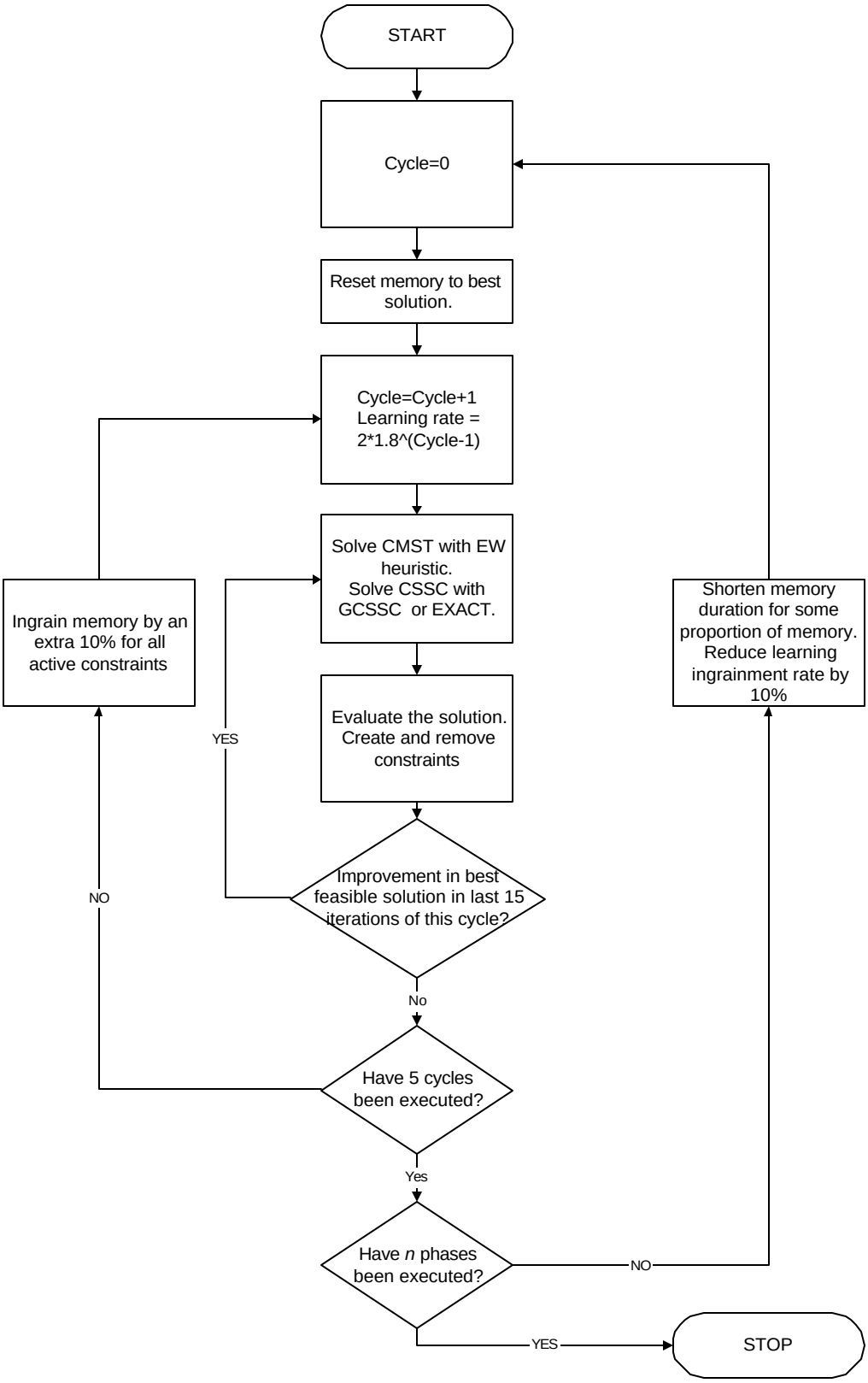

Figure A.1: Flowchart of ART

APPENDIX B. ART APPLIED TO THE TSP

In order to illustrate ART's generalizability to other complex optimization problems, we show the ease by which ART can be applied to the well-known traveling salesman problem (TSP). In the philosophy of ART, only minor modifications of the ART heuristic are necessary. The main component of ART that must be exchanged is the primary heuristic. No decomposition of the TSP problem will be necessary. Thus, we only need one primary heuristic that quickly generates solutions for the TSP. One simple primary heuristic for the TSP is the nearest neighbor algorithm. This procedure is based on connecting a node on the tour to its nearest neighboring node. That is, start with the origination node (*S*) and connect it to its nearest neighbor (*T*). Repeat this by connecting *T* to its nearest neighbor not previously visited, label this node *T*, and continue until all nodes have been visited, and then connect the last node back to *S*. A two-opt greedy improvement heuristic (Syslo et. al, 1983) is then applied to this solution (a two-opt procedure is one that attempts to exchange pairs of nodes until no more cost-reducing changes are possible). We label the combined nearest neighbor and two-opt procedure NNT.

In our implementation we replace the primary heuristics (as outlined in Figure A.1) with the NNT heuristic, and arrive at an ART procedure as outlined in Figure B.1. It is worth noting that in addition to exchanging the primary heuristic in ART, only minor adjustments in memory settings was necessary to implement ART for the TSP. Also, a final three-opt improvement heuristic (Syslo et. al, 1983) was included after the final iteration. Now, applying the ART procedure as depicted in Figure B.1 to a set of 30 well-known Euclidean TSP problem instances (from the TSPLib) we achieve the results as given in Table B.1.

In Table B.1 we report the results from applying the NNT heuristic as a stand-alone procedure, as well as the results from applying this as the primary heuristic within ART. We ran ART 25 times for each problem instance, and report the minimum, maximum, and average solutions, as well as the minimum, maximum and average CPU times. Because the stand-alone heuristic procedure is deterministic, it was executed only once for each problem instance.

From Table B.1 we see that the ART heuristic produces excellent results for the TSP instances tested. On average, the optimality gap for ART is 0.248%, whereas the stand-alone NNT procedure has an average optimality gap of 4.189%. Moreover, ART found an optimal solution to 40% of the test problems, whereas the NNT heuristic did not find a single optimal solution. The CPU time for the ART procedure was a modest average 264 seconds, whereas the NNT heuristic required negligible CPU time in all instances (the ART CPU requirements were heavily influenced by one large outlier [problem Lin318]; the average CPU time was 147 seconds without this outlier).

Given the results from Table B.1, we conclude that the ART solution method is easily generalizable, and provides excellent solution quality across problem domains.

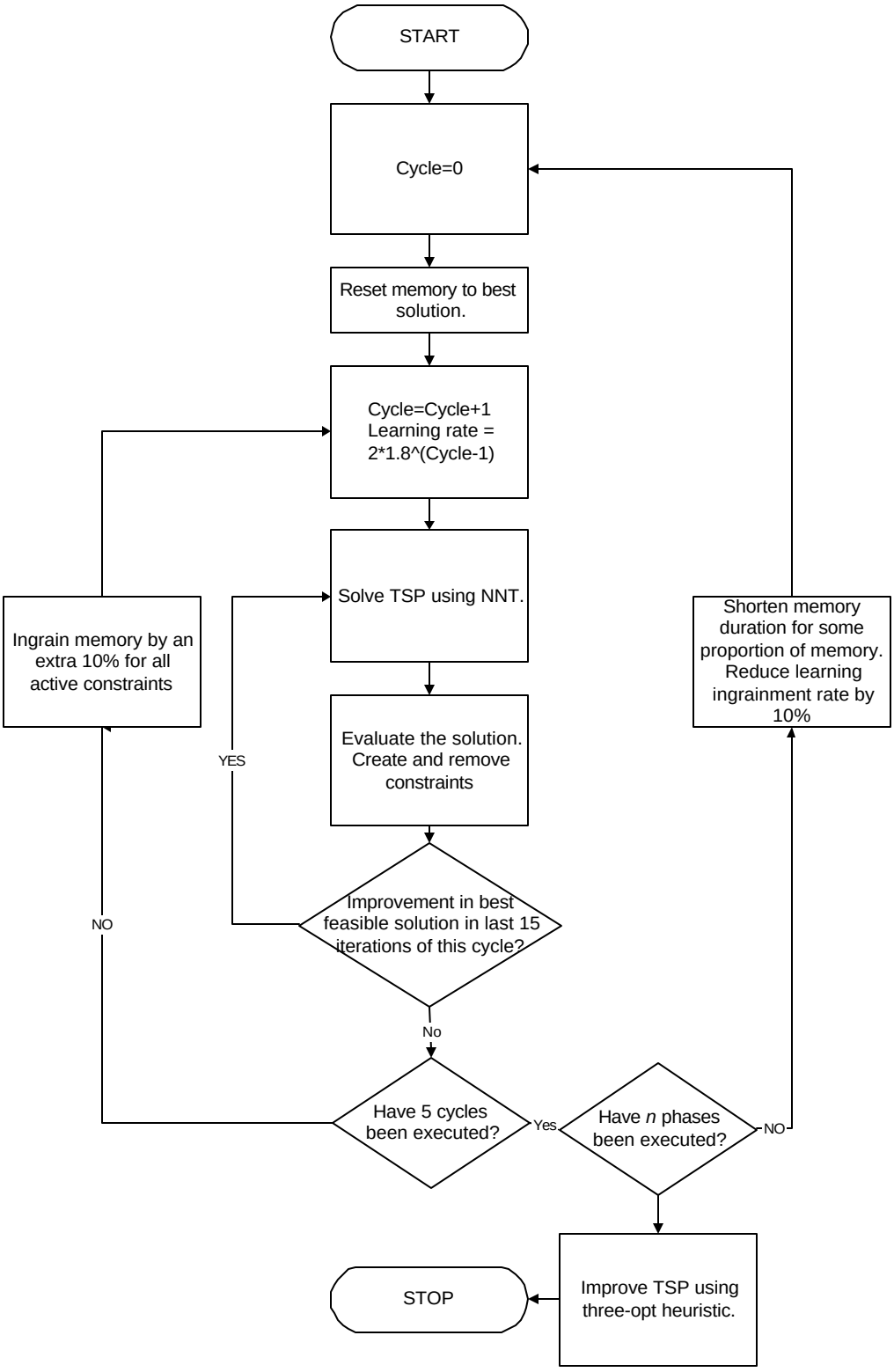

Figure B.1: The ART Procedure as applied to the TSP

TSPLIB PROBLEM		OPTIMAL	NNT HEURISTIC	ART SOLUTION			ART COMPUTATIONAL TIME (IN SECONDS)			OPTIMALITY GAP	
NAME	SIZE	SOLUTION	SOLUTION	MIN	MAX	AVG	MIN	MAX	AVG	NNT HEURISTIC SOLUTION	BEST ART SOLUTION
eil51	51	426	435	426	429	426	10	13	11	2.11%	0.0000%
berlin52	52	7,542	7,842	7,542	7,542	7,542	6	9	7	3.98%	0.0000%
st70	70	675	738	679	683	680	16	25	20	9.33%	0.5926%
eil76	76	538	562	539	548	543	17	49	31	4.46%	0.1859%
pr76	76	108,159	114,283	108,159	109,787	109,009	29	45	40	5.66%	0.0000%
rat99	99	1,211	1,315	1,212	1,249	1,237	33	70	46	8.59%	0.0826%
kroA100	100	21,282	21,919	21,282	21,492	21,353	39	65	49	2.99%	0.0000%
kroB100	100	22,141	22,349	22,246	22,316	22,269	33	47	43	0.94%	0.4742%
kroC100	100	20,749	21,997	20,749	21,257	20,977	37	68	50	6.01%	0.0000%
kroD100	100	21,204	22,444	21,294	21,683	21,414	37	132	60	5.85%	0.4244%
kroE100	100	22,068	23,350	22,068	22,384	22,211	36	66	49	5.81%	0.0000%
rd100	100	7,910	8,118	7,910	8,039	7,932	41	73	55	2.63%	0.0000%
eil101	101	629	637	631	631	631	55	73	64	1.27%	0.3180%
lin105	105	14,379	14,712	14,401	14,544	14,472	51	70	63	2.32%	0.1530%
pr107	107	44,303	44,648	44,303	44,303	44,303	43	47	44	0.78%	0.0000%
pr124	124	59,030	60,372	59,030	59,246	59,131	64	82	69	2.27%	0.0000%
bier127	127	118,282	120,460	118,967	119,537	119,189	95	148	102	1.84%	0.5791%
ch130	130	6,110	6,479	6,153	6,247	6,206	88	205	140	6.04%	0.7038%
pr136	136	96,772	105,916	96,876	100,174	97,990	107	213	150	9.45%	0.1075%
pr144	144	58,537	61,244	58,537	58,763	58,565	124	187	160	4.62%	0.0000%
ch150	150	6,528	6,620	6,533	6,609	6,568	114	341	151	1.41%	0.0766%
kroA150	150	26,524	28,458	26,525	27,052	26,691	162	523	194	7.29%	0.0038%
kroB150	150	26,130	27,131	26,130	26,689	26,397	123	312	187	3.83%	0.0000%
pr152	152	73,682	75,227	74,352	74,639	74,627	165	881	212	2.10%	0.9093%
u159	159	42,080	44,047	42,080	43,076	42,601	180	446	267	4.67%	0.0000%
rat195	195	2,323	2,392	2,338	2,367	2,363	271	775	446	2.97%	0.6457%
d198	198	15,780	16,036	15,845	15,913	15,855	391	684	486	1.62%	0.4119%
kroA200	200	29,368	29,668	29,477	29,664	29,611	247	671	338	1.02%	0.3712%
kroB200	200	29,437	31,873	29,647	30,360	30,094	431	1,026	725	8.28%	0.7134%
lin318	318	42,029	44,349	42,318	43,323	42,905	2,387	5,380	3,675	5.52%	0.6876%
	(nodes)								**AVERAGE**	4.189%	0.2480%

Table B.1 Results from applying ART to the TSP