StruDiCO: Structured Denoising Diffusion with Gradient-free Inference-stage Boosting for Memory and Time Efficient Combinatorial Optimization (NeurIPS, 2025)

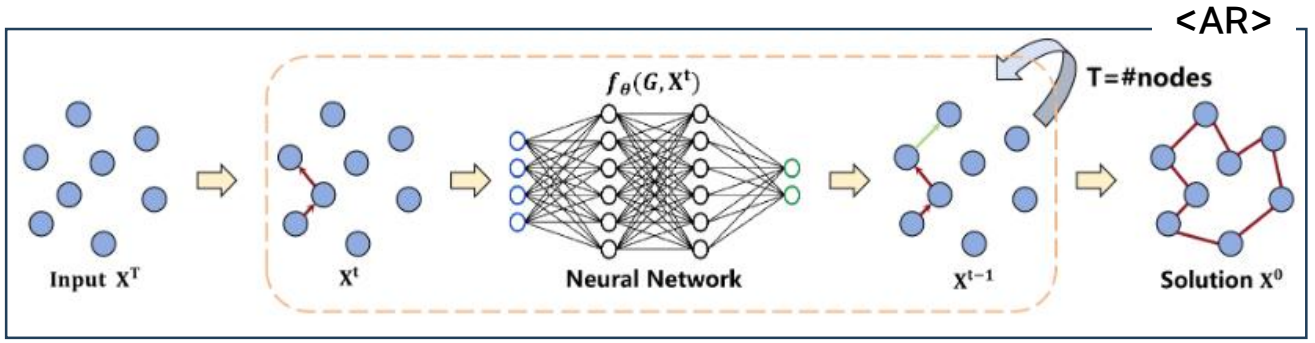
26. 01. 16

김정현

I Introduction

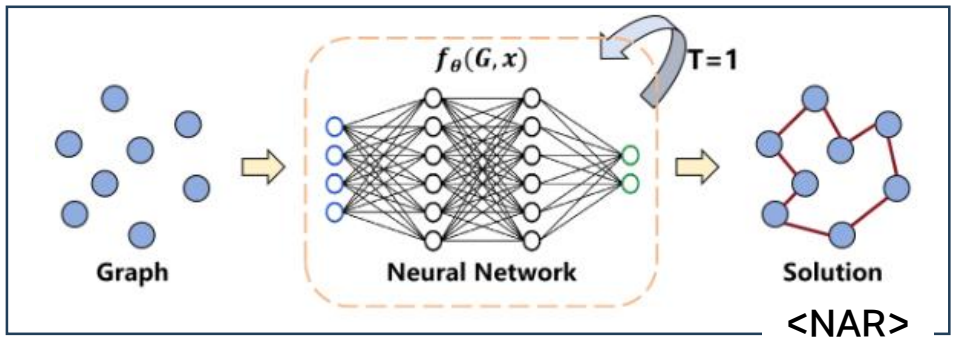
❖ Background: Combinatorial Optimization (CO)

- ✓ 수많은 가능한 '조합' 중에서, 주어진 '조건(constraints)'을 만족하면서 가장 좋은 '해(solution)'를 찾는 문제
- ✓ NP-hard의 특성:
  - 문제가 복잡하고 커질수록, 가능한 경우의 수가 '기하급수적으로' 늘어남
  - 현실적인 시간 안에 완벽한 최적해를 찾는 것이 매우 어려움
- ✓ 전통적으로 정확해법, 휴리스틱, 메타 휴리스틱 등을 사용



❖ Neural Combinatorial Optimization (NCO)

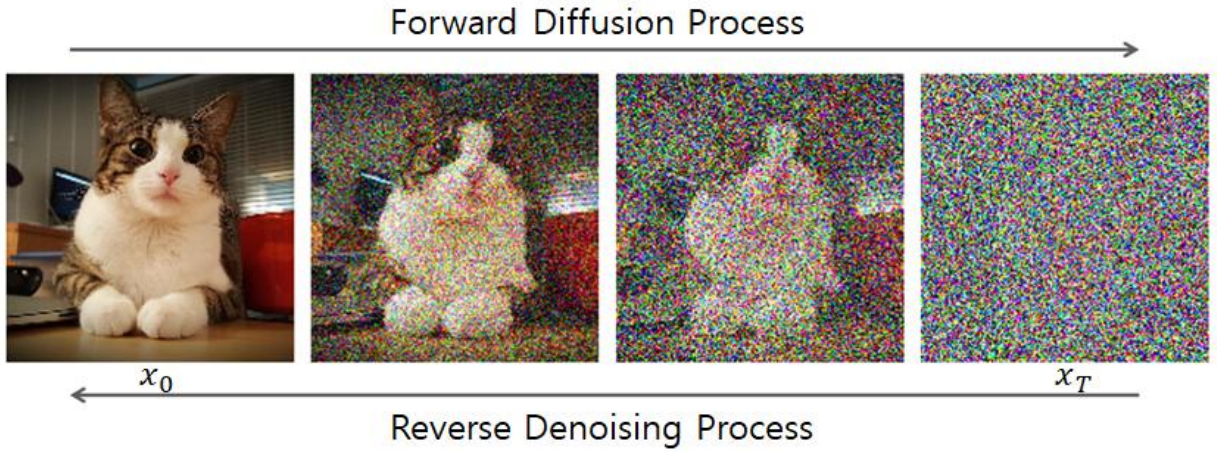
- ✓ Autoregressive (AR) methods:
  - 변수를 순차적으로 하나씩 선택하여 해를 구성하는 방식
  - 모델은 이전 단계에서 선택된 변수들을 기반으로, 다음 변수를 선택할 확률을 예측
  - 각 단계별 해석이 가능하지만, 느린 추론과 대규모 문제로 확장이 어려움
- ✓ Non-Autoregressive (NAR) methods:
  - 모든 변수에 대한 예측을 한 번에(one-shot) 병렬적으로 수행하는 방식
  - 모델은 모든 변수에 대해 선택될 확률(또는 점수)을 포함하는 "히트맵"을 한 번에 생성
  - 이후, 이 확률 분포를 바탕으로 휴리스틱 후처리를 통해 제약 조건을 만족하는 최종 해를 추출
  - 빠른 추론과 전역적 모델링이 가능하지만, 중간 단계에서 해가 어떻게 구성되는지에 대한 과정이 불투명함

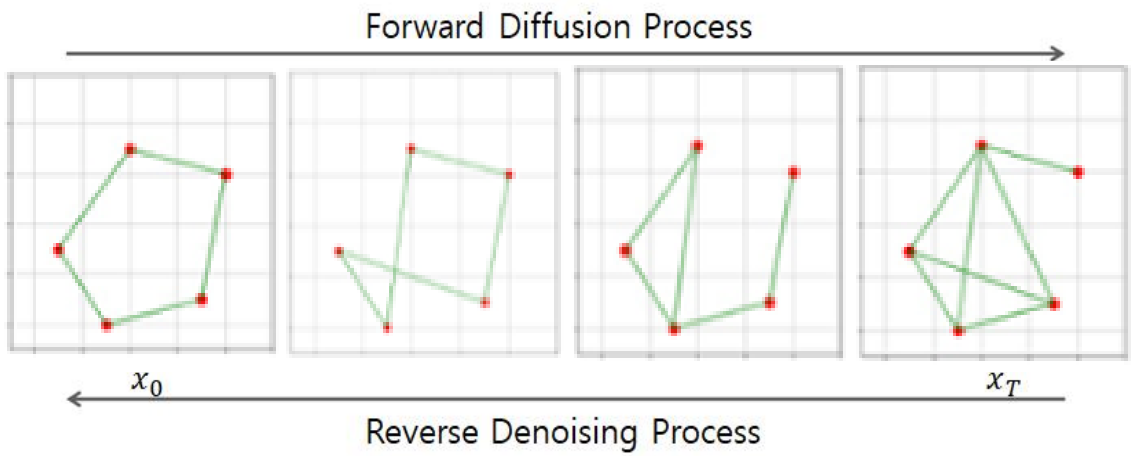


I Introduction

❖ Diffusion Model

- ✓ 최근 이미지 생성 분야에서 놀라운 성능을 보여주며 주목받는 '생성형 모델 ' 의 한 종류
- ✓ 2020년 발표된 DDPMs 을 통해 이미지 처리에서 뛰어난 성능을 입증하며 큰 관심을 받기 시작
- ✓ 2023년 발표된 DIFUSCO 논문에서 Diffusion 모델을 활용하여 CO를 푸는 방식을 제안
- ✓ 핵심 아이디어:
  - '깨끗한 이미지(데이터)'에 점차 '노이즈'를 추가하여 완전히 흐릿하게 만든 후 → Forward Process
  - 반대로 완전히 노이즈화된 상태에서 시작하여 점차 노이즈를 제거하며 깨끗한 데이터를 복원하는 방식 → Reverse Process





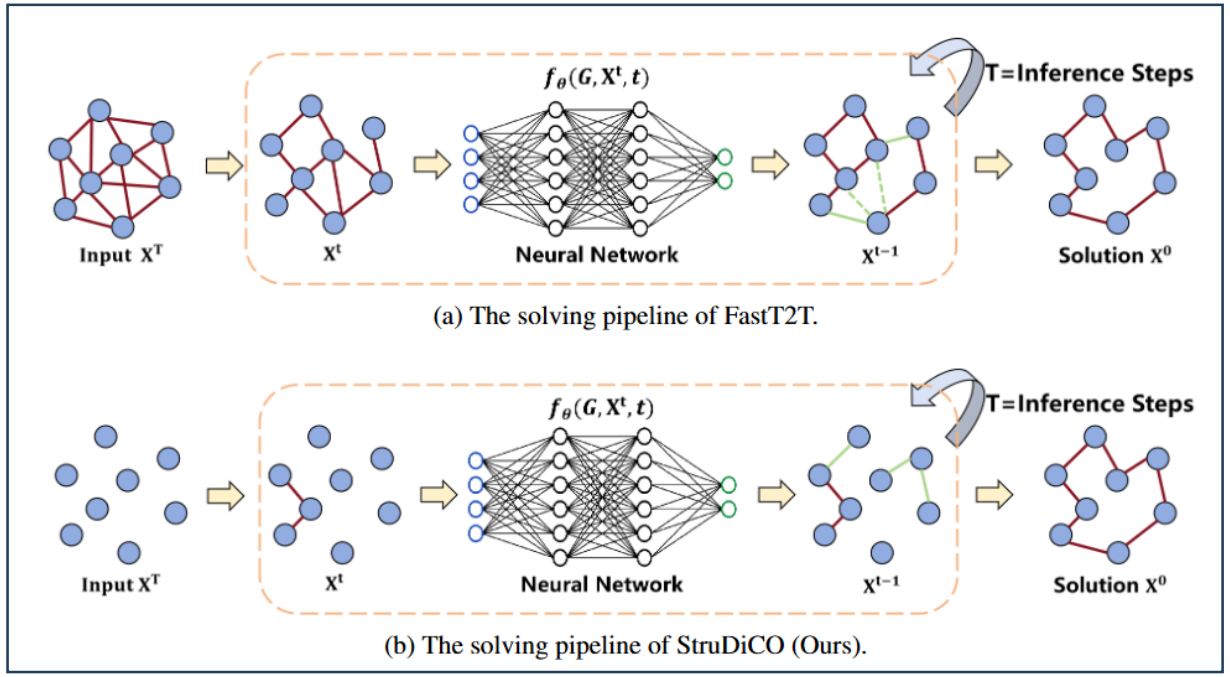
I Introduction

❖ 기존 Diffusion Model의 한계

- ✓ 해석 불가능한 중간 상태 (Opaque Intermediate States):
  - 해가 어떻게 점진적으로 만들어지는지 알 수 없음
  - 추론 과정에서 생성되는 중간 결과물들이 의미를 갖지 않거나, 문제의 제약 조건을 위반하는 경우가 있음
- ✓ 구조적 연속성 부족 (Lack of Structural Continuity):
  - 노이즈를 추가하거나 제거하는 과정에서 CO 문제의 구조적 특성이나 제약 조건을 제대로 보존하지 못함
- ✓ 높은 계산 비용 (High Computational Cost):
  - 좋은 해를 얻기 위해 수백~수천 번의 복잡한 계산(추론 단계)이 필요
  - 느린 추론 속도와 높은 컴퓨팅 자원 요구

❖ Proposed Approach: StruDiCO

- ✓ Variable-Absorption Noising
  - Forward 과정에서 변수를 점진적으로 비활성화
  - 모든 중간 상태가 구조적으로 유효한 부분 해
- ✓ Constrained Consistency Sampling
  - 신뢰도 낮은 변수를 제거하여 탐색 공간 축소 → 안정적인 추론 경로
- ✓ Gradient-Free Objective-Aware Refinement
  - 역전파 없이 목적함수 반영 → 구조를 보존하며 해 개선

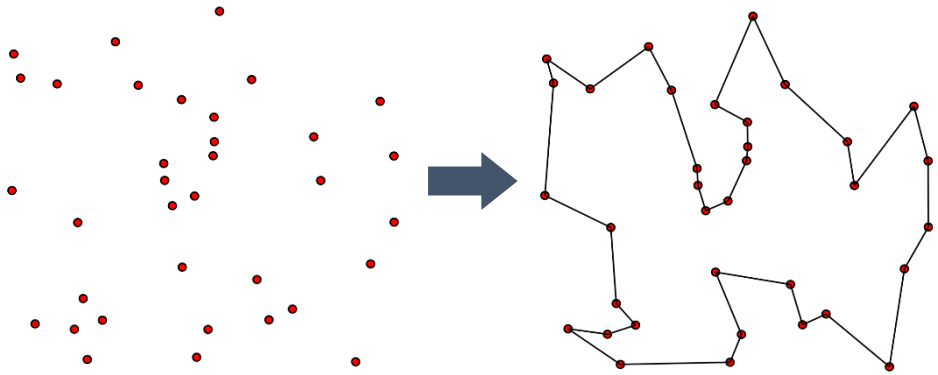


Ⅱ Combinatorial Optimization on Graphs

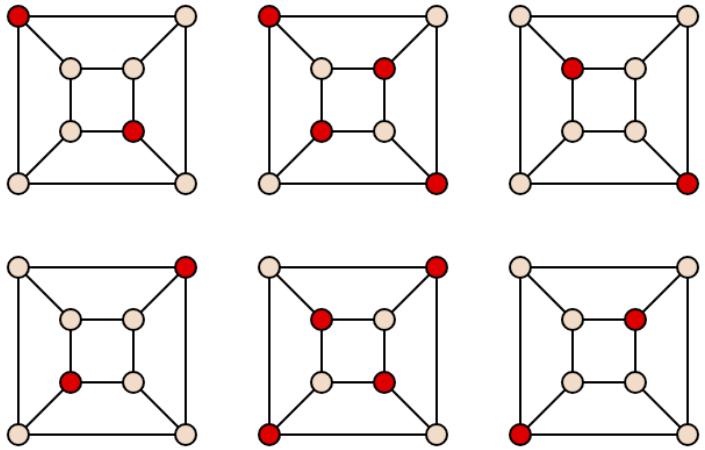
❖ Graph-based CO Formulation

- ✓ CO 문제는 '그래프' 구조로 표현 가능
- ✓ 그래프는 노드들과 이 노드들을 연결하는 엣지들로 구성
- ✓ Ex) TSP : 노드=도시 / 엣지=도시 간의 거리(비용)

<TSP>



<MIS>



❖ Problem Representation

- ✓ 문제 인스턴스: $G = (V, E)$
  - V : 노드 집합 / E : 엣지 집합
- ✓ 최적화 변수(variable): $x \in \{0,1\}^N$
  - TSP (엣지 선택 문제): 변수 = "엣지(I, j)를 선택했는가?"
  - MIS (노드 선택 문제): 변수 = "노드 i를 선택했는가?"

❖ General CO Formulation

- ✓ 목표: $\min_{x \in \Omega} l(x; G)$
  - $l(x; G)$: 해 x의 좋고 나쁨을 측정하는 목적함수 (ex: TSP는 경로 길이의 총합)
  - Ω : 문제별 제약 조건을 만족하는 해 공간
- ✓ 제약 조건을 만족하는 해 중에서 목적 함수 값을 최소화 or 최대화하는 해 찾기

Ⅱ Optimization Consistency

❖ Diffusion-based Optimization

- ✓ 기본 아이디어:
  - 최적해에서 노이즈를 점점 추가하고, 다시 노이즈를 제거하며 해 복원
  - 해를 복원하는데 수많은 단계가 필요함 → 계산 비용 큼

❖ Consistency Model

- ✓ 핵심 개념:
  - ✓ 어떤 노이즈 단계(t)에서 시작하든, 항상 동일한 깨끗한 해를 복원하도록 학습
- ✓ CO 문제의 경우, 어떤 중간 노이즈 상태에서도 주어진 문제 인스턴스에 대한 동일한 최적해로 수렴해야 함
- ✓ 추론 단계 수 대폭 감소 → 빠른 추론 가능

❖ 학습 목표

- ✓ Self-Consistency: 서로 다른 노이즈 레벨($t1$, $t2$)에서 예측한 결과가 유사하도록 학습

$$\mathcal{L}(\theta) = \mathbb{E}_{t_1,t_2}\left[d\left(f_\theta(\mathbf{x}_{t_1}, t_1, G),\ f_\theta(\mathbf{x}_{t_2}, t_2, G)\right)\right]$$

- ✓ Ground-truth Consistency: 모델이 예측한 결과와 실제 최적해 사이의 거리를 줄이도록 학습

$$\mathcal{L}(\theta) = \mathbb{E}_t\left[d\left(f_\theta(\mathbf{x}_t, t, G),\ \mathbf{x}^*\right)\right]$$

Ⅲ Variable-Absorption Noising Process

❖ Standard Discrete Diffusion Models

✓ Forward Process:

- 목표: 원본 데이터 x_0에서 시작하여 점차 노이즈를 더해 완전히 랜덤한 노이즈 상태 x_T로 만듦
- 노이즈 주입 방식:
  - 각 스텝 t에서 특정 확률(β_t)로 각 변수의 값을 무작위로 변경
  - Ex) 이진 데이터의 경우, 1을 0으로 바꾸거나 0을 1로 바꾸는 식으로 노이즈 주입
- 전이 행렬:
  - 보통 대칭적(symmetric)이고 이중 확률 행렬(doubly stochastic) Q_t를 사용 → $Q_t = \begin{bmatrix} 1-\beta_t & \beta_t \\ \beta_t & 1-\beta_t \end{bmatrix}$

❖ Variable-Absorption Noising Process

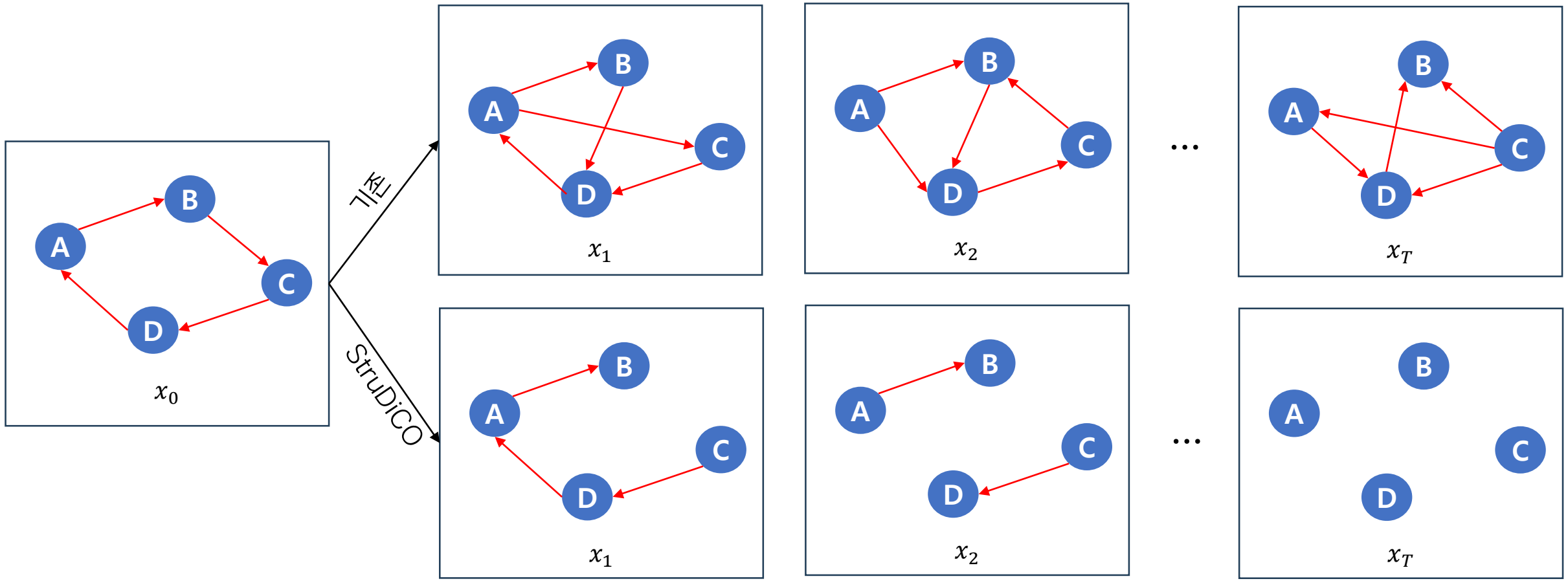
✓ Forward Process:

- 목표: 유효한 해 x_0에서 시작하여 변수를 점진적으로 '흡수'(제거)하여 완전히 빈 상태 $x_T(0^N)$로 수렴하도록 만듦
- 노이즈 주입 방식:
  - 선택된 변수만 확률적으로 제거(drop) → 선택되지 않은 변수는 절대 활성화되지 않음
  - 이 과정을 통해 모든 중간 상태 x_t가 항상 유효한 부분 해의 구조를 유지
- 전이 행렬:
  - 비대칭(asymmetric) 전이 행렬 Q_t를 사용 → $Q_t = \begin{bmatrix} 1 & 0 \\ \beta_t & 1-\beta_t \end{bmatrix}$

Ⅲ Variable-Absorption Noising Process

❖ Variable-Absorption Noising Process

- ✓ 도시: A, B, C, D
- ✓ 시작 (x_0): A → B → C → D → A



Ⅲ Constrained Consistency Sampling

❖ Standard Diffusion Sampling

- ✓ 완전히 노이즈가 낀 상태 (x_T)에서 시작: 어떤 무작위 분포에서 x_T 샘플링
- ✓ 단계별 디노이징: 각 타임스텝 t마다 신경망 모델을 통해 이전 단계의 데이터 x_{t-1} 예측
  - $p_\theta(x_0|x_\tau, \tau, G) \approx f_\theta(x_\tau, \tau, G)$
- ✓ 문제점:
  - 노이즈의 영향: Reverse 과정의 중간 단계가 불안정하여 모델이 예측한 확률값이 실제 최적 해와 거리가 멀거나, 노이즈의 영향을 많이 받을 수 있음
  - 모든 예측값 사용: 모델이 예측한 모든 확률값, 즉 신뢰도가 낮은 예측값까지도 그대로 샘플링에 사용

❖ Constrained Consistency Sampling

- ✓ 부분적인 노이즈 상태 (x_τ)에서 시작:
- ✓ 디노이징: 현재 상태 (x_τ)에서 신경망 모델을 통해 x_0 예측
  - $p_\theta(x_0|G) \leftarrow f_\theta(x_\tau, \tau, G)$
- ✓ "제약 적용 (Constrained Sampling)":
  - 임계값 δ 설정: 사용자는 '이 정도 신뢰도 이상은 되어야 선택한다'는 임계값 δ 설정
  - 신뢰도 필터링: 모델이 예측한 확률 분포 $p_\theta(x_0 = 1|G)$ 확인
    - 기존 방식: 모든 확률값을 그대로 사용하여 샘플링
    - CCS: δ 보다 낮은 확률을 가진 변수들은 "무시"하거나 "선택되지 않은 것"으로 간주

Algorithm 1 Multistep Constrained Consistency Sampling

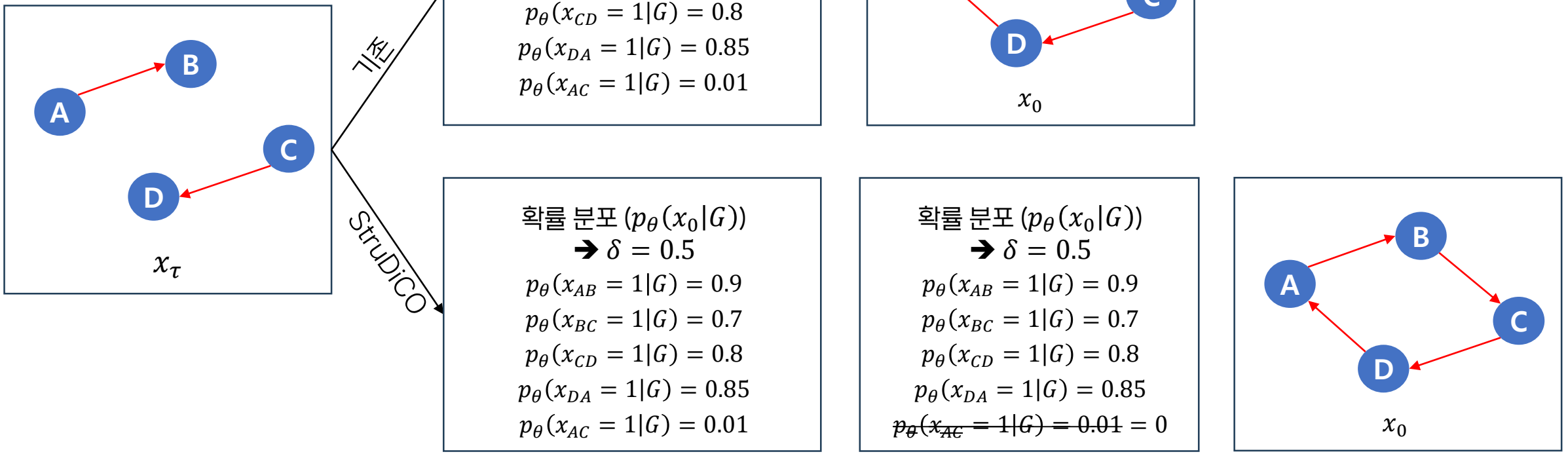
Require: Consistency model $f_\theta(\cdot,\cdot,\cdot)$, graph instance G, time sequence $\tau_1 > \cdots > \tau_{N_\tau - 1}$, threshold δ

$\mathbf{x}_T = \mathbf{0}_N$
$p_\theta(\mathbf{x}_0 \mid G) \leftarrow f_\theta(\mathbf{x}_T, T, G)$
$\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0 \mid G)$
for $n = 1$ to $N_\tau - 1$ **do**
  // variable-absorption process
  $\mathbf{x}_{\tau_n} \sim \text{Cat}(\mathbf{p} = \tilde{\mathbf{x}}_0 \overline{\mathbf{Q}}_{\tau_n})$
  $p_\theta(\mathbf{x}_0 \mid G) \leftarrow f_\theta(\mathbf{x}_{\tau_n}, \tau_n, G)$
  // constrained sampling over $\mathbf{m}$
  $\mathbf{m} \leftarrow \mathbb{I}(p_\theta(\mathbf{x}_0 = 1 \mid G) > \delta)$
  $\mathbf{x}_0 \sim \text{Bernoulli}(\mathbf{m} \odot p_\theta(\mathbf{x}_0 = 1 \mid G))$
end for
return Solution $\mathbf{x}_0$

Ⅲ Constrained Consistency Sampling

❖ Constrained Consistency Sampling for Reverse Optimization

- ✓ 도시: A, B, C, D
- ✓ x_0: A → B → C → D → A



Ⅲ Objective-Aware Gradient-Free Refinement

❖ Refinement

- ✓ Reverse Process + Constrained Sampling 단계를 거치면 구조적으로 안정적이고 해석 가능한 해 후보를 얻을 수 있음
- ✓ Greedy decoding을 통해 Hard feasibility를 만족하는 유효한 해를 얻음
- ✓ 하지만 이 해는
  - Feasible하긴 해도, 목적함수 기준으로는 아직 최적이 아닐 수 있음
  - 따라서 최종 해의 품질을 극대화하기 위한 '반복적 정제(iterative refinement)' 과정이 필요

❖ Standard Diffusion Sampling

- ✓ 기존의 Diffusion 모델(T2T, FastT2T): 해 품질 향상을 위해 Gradient-guided refinement 방식을 주로 사용
- ✓ 작동 방식:
  - 초기 해 생성 → (연속값으로 완화된) 목표 함수에 대한 그래디언트 계산 → 해 업데이트
- ✓ 문제점:
  - 계산 비용: 그래디언트 계산은 복잡하며, 특히 이산적인 CO 문제에서는 근사화나 복잡한 계산이 필요하여 시간 및 메모리 비용이 높음
  - 이산성 문제: 연속적인 변수로 근사하는 과정에서 정보 손실이 발생할 수 있음 → 그래디언트 기반 업데이트가 최적의 이산 해를 찾지 못할 수도 있음

Ⅲ Objective-Aware Gradient-Free Refinement

❖ Objective-Aware Gradient-Free Refinement

- ✓ 핵심 아이디어: 그래디언트 계산 없이, 문제의 목표 함수 정보를 직접 디코딩 과정에 통합하여 효율적으로 해를 개선
- ✓ 구조 인식 교란 (Structure-Aware Perturbation):
  - 현재까지 얻어진 유효한 해 s 준비
  - Variable-Absorption Noising Process를 사용하여 s에 구조를 보존하는 교란(perturbation)을 가함
  - 이미 의미 있는 부분 해에서 출발하여, 선택된 변수 일부만을 '흡수'하는 방식으로 구조적 연속성 유지
- ✓ 정제된 예측 (Denoising via Consistency Model):
  - 교란이 가해진 상태 $s_{\alpha T}$를 Consistency Model f_θ에 입력하여 정제된 해(denoised solution) 예측을 도출
  - $p_\theta \leftarrow f_\theta(s_{\alpha T}|G)$
- ✓ 목표 함수 기반 탐색 (Objective-Guided Greedy Decoding):
  - 정제된 예측 분포 p_θ를 기반으로, 목표 함수를 직접적으로 반영하는 점수 계산
  - $score_i = p_i/(\phi_i + \varepsilon)$
    - $p_i = p_\theta\left({x_0}^i = 1 \middle| G\right)$: 변수 i가 선택될 확률 (모델의 신뢰도)
    - ϕ_i : 변수 i가 목표 함수에 대해 가지는 페널티 또는 비용
- ✓ 탐욕적(Greedy) 선택: 계산된 점수가 높은 순서대로 변수를 선택
- ✓ 위 과정을 반복하여 점진적으로 해 품질 개선

Algorithm 2 Objective-Aware Gradient-Free Refinement

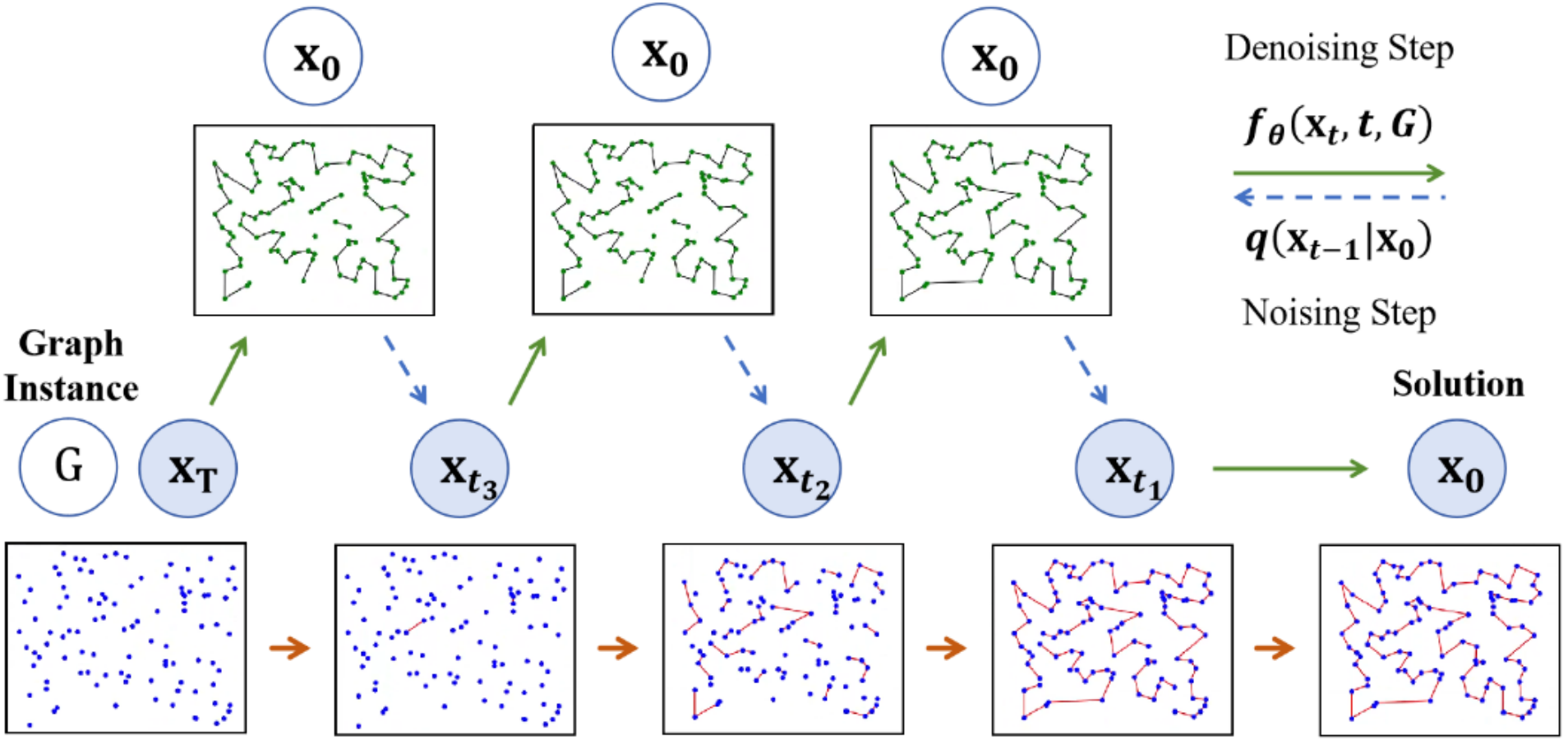
Require: Consistency model f_θ, graph instance G, prediction p_θ, score function $\text{score}_i = p_i/\phi_i$, number of iterations T_g, perturbation ratio αT

for $t = 1$ to T_g **do**
  $\mathbf{m} \leftarrow \mathbb{I}(p_\theta(\mathbf{x}_0 = 1 \mid G) > \delta)$
  $\mathbf{s} \leftarrow \mathbf{m} \odot \mathbf{s}$
  $\tilde{\mathbf{s}} \leftarrow \text{OneHotEncode}(\mathbf{s})$
  //apply variable-absorption noise
  $\mathbf{s}_{\alpha T} \sim \text{Cat}(\tilde{\mathbf{s}} \cdot \overline{\mathbf{Q}}_{\alpha T})$
  $p_\theta \leftarrow f_\theta(\mathbf{s}_{\alpha T} \mid G)$
  $\mathbf{s} \leftarrow \text{GreedyDecode}(p_\theta, \text{score})$
end for
return $\mathbf{s}$

Ⅲ Methodology

❖ Overview of the StruDiCO framework

- ✓ Forward process (0 → T)는 variable-absorption noising model을 통해 선택된 엣지를 점진적으로 제거
- ✓ Reverse process(T → 0)는 엣지를 선택함으로써 해를 점진적으로 재구성



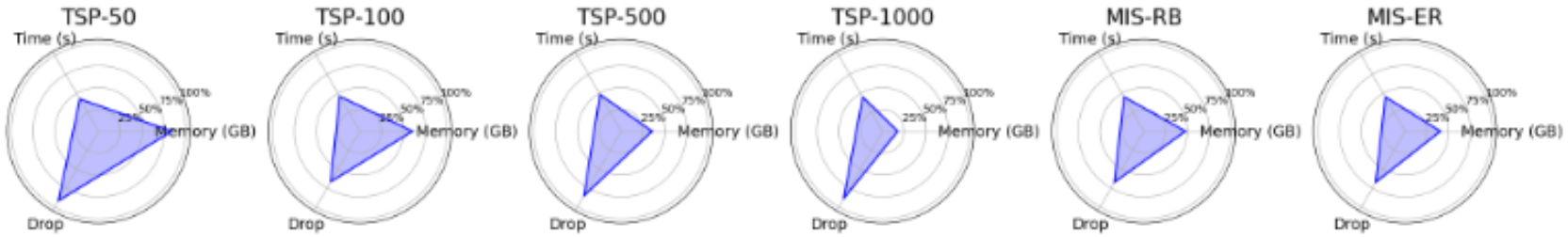
IV Experiment

❖ 테스트 문제

- ✓ Traveling Salesman Problem (TSP): 도시 간 이동 경로 최적화 (해밀턴 순환, 최소 거리)
- ✓ Maximum Independent Set (MIS): 인접하지 않은 노드 집합 최대화

❖ Baselines

- ✓ 정확 해법 (Exact Solvers): Concorde (TSP)
- ✓ 휴리스틱 (Heuristics): LKH3 (TSP), KaMIS (MIS)
- ✓ 기존 신경망 솔버: GCN, DIFUSCO, T2T, Fast T2T 등



Method	Multi-Step Structure Construction	Objective-Guided Gradient-free	Forward	Backward	Complexity Time	Memory
Fast T2T [24]	✗	✗	2	1	$\sim O(3BLN^2d)$	$\sim O(2BLN^2d)$
StruDiCO (Ours)	✓	✓	1	0	$O(BLN^2d)$	$O(BLN^2d)$

IV Experiment

❖ 평가 지표

- ✓ Objective (목표 값): TSP (총 거리), MIS (최대 독립 집합 크기)
- ✓ Drop (성능 저하): 기준 해법 대비 상대적 성능 저하율
- ✓ Time (추론 시간): 인스턴스당 평균 실행 시간

❖ Experiment Setting

- ✓ 모든 모델은 배치 크기 1, 단일 스레드 모드로 평가
- ✓ T_s: 초기 추론 단계 수 / T_g: 개선 단계 수

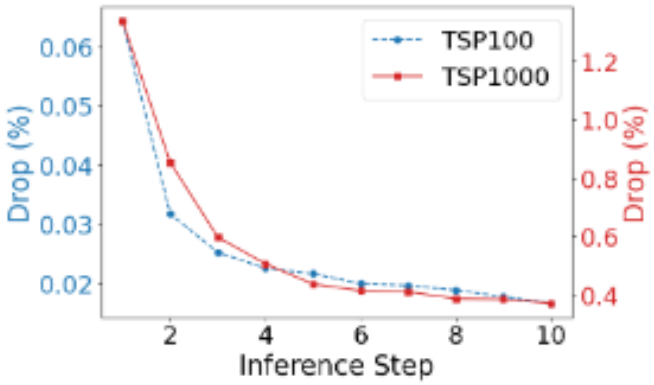
Algorithm	Type	TSP-50			TSP-100		
		Length↓	Drop↓	Time↓	Length↓	Drop↓	Time↓
Concorde [36]	Exact	5.688	0.00%	0.074s	7.756	0.00%	0.404s
LKH3 [26]	Heuristics	5.688	0.00%	0.058s	7.756	0.00%	0.176s
GCN [17]	SL+G+2Opt	5.694	0.115%	0.009s	7.807	0.649%	0.019s
GNNGLS* [37]	SL+G+2Opt	5.707	0.333%	0.019s	7.857	1.295%	0.129s
DIMES* [19]	Meta+RL+G+2Opt	5.823	2.387%	0.018s	8.007	3.232%	0.057s
AM* [12]	RL+G+2Opt	5.679	0.167%	0.048s	7.826	0.898%	0.438s
POMO [13]	RL+G+2Opt	5.693	0.102%	0.019s	7.854	1.253%	0.116s
Sym-NCO [14]	RL+G+2Opt	5.694	0.122%	0.198s	7.818	0.796%	0.634s
BQ-NCO* [15]	RL+G+2Opt	5.795	1.894%	0.205s	7.893	1.772%	0.387s
DIFUSCO (T_s=50) [22]	SL+G	5.692	0.076%	0.229s	7.851	1.216%	0.591s
Fast T2T (T_s=3) [24]	SL+G	5.694	0.111%	0.024s	7.798	0.537%	0.038s
StruDiCO (T_s=3)	SL+G	5.692	0.071%	0.023s	7.786	0.392%	0.037s
T2T (T_s=50,T_g=30) [23]	SL+G	5.688	0.015%	0.717s	7.765	0.125%	1.559s
Fast T2T (T_s=3,T_g=3) [24]	SL+G	5.688	0.014%	0.139s	7.760	0.052%	0.182s
StruDiCO (T_s=3,T_g=3)	SL+G	5.688	0.014%	0.051s	7.758	0.036%	0.082s
DIFUSCO (T_s=50) [22]	SL+G+2Opt	5.690	0.046%	0.23s	7.776	0.262%	0.590s
Fast T2T (T_s=3) [24]	SL+G+2Opt	5.689	0.031%	0.026s	7.764	0.101%	0.037s
StruDiCO (T_s=3)	SL+G+2Opt	5.689	0.019%	0.026s	7.761	0.067%	0.038s
T2T (T_s=50,T_g=30) [23]	SL+G+2Opt	5.688	0.012%	1.098s	7.760	0.058%	1.571s
Fast T2T (T_s=3,T_g=3) [24]	SL+G+2Opt	5.688	0.012%	0.139s	7.759	0.036%	0.180s
StruDiCO (T_s=3,T_g=3)	SL+G+2Opt	5.688	0.011%	0.059s	7.756	0.024%	0.083s

Algorithm	Type	TSP-500			TSP-1000		
		Length↓	Drop↓	Time	Length↓	Drop↓	Time
Mathematical Solvers or Heuristics							
Concorde [36]	Exact	16.546	0.00%	18.672s	23.118	0.00%	84.413s
LKH-3 [26]	Heuristics	16.546	0.00%	1.848s	23.119	0.00%	4.641s
Learning-based Solvers with Greedy Decoding							
GCN [17]	SL+G+2OPT	16.899	2.121%	0.128s	–	–	–
DIMES* [19]	RL+G+2Opt	17.165	3.742%	0.453s	–	–	–
BQ-NCO* [15]	RL+G+2Opt	16.838	1.766%	2.454s	23.647	2.287%	5.722s
DIFUSCO (T_s=50) [22]	SL+G	18.136	9.611%	1.442s	25.667	11.022%	4.982s
Fast T2T (T_s=5) [24]	SL+G	17.467	5.551%	0.251s	24.698	6.831%	0.971s
StruDiCO (T_s=5)	SL+G	17.404	5.172%	0.239s	23.118	5.967%	0.900s
T2T (T_s=50,T_g=30) [23]	SL+G	17.470	5.578%	3.334s	25.168	8.868%	12.871s
Fast T2T (T_s=5,T_g=5) [24]	SL+G	16.919	2.244%	1.426s	23.936	3.539%	5.988s
StruDiCO (T_s=5,T_g=5)	SL+G	16.887	2.045%	0.651s	23.755	2.753%	2.533s
DIFUSCO (T_s=50) [22]	SL+G+2Opt	16.817	1.641%	1.433s	23.567	1.936%	5.036s
Fast T2T (T_s=5) [24]	SL+G+2Opt	16.701	0.922%	0.261s	23.388	1.167%	0.979s
StruDiCO (T_s=5)	SL+G+2Opt	16.669	0.728%	0.247s	23.348	0.996%	0.915s
T2T (T_s=50,T_g=30) [23]	SL+G+2Opt	16.677	0.793%	3.367s	23.397	1.209%	13.089s
Fast T2T (T_s=5,T_g=5) [24]	SL+G+2Opt	16.611	0.383%	1.387s	23.257	0.603%	5.779s
StruDiCO (T_s=5,T_g=5)	SL+G+2Opt	16.602	0.326%	0.674s	23.242	0.535%	2.614s
Learning-based Solvers with 4× Sampling Decoding							
DIFUSCO (T_s=50) [22]	SL+S	17.533	5.959%	2.131s	25.059	8.399%	19.023s
Fast T2T (T_s=5) [24]	SL+S	17.024	2.874%	0.877s	24.096	4.231%	3.635s
StruDiCO (T_s=5)	SL+S	16.995	2.696%	0.839s	24.007	3.849%	3.430s
T2T (T_s=50,T_g=30) [23]	SL+S	17.054	3.070%	9.722s	24.838	7.444%	34.813s
Fast T2T (T_s=5,T_g=5) [24]	SL+S	16.710	0.978%	5.216s	23.712	2.570%	13.346s
StruDiCO (T_s=5,T_g=5)	SL+S	16.688	0.846%	2.312s	23.471	1.527%	6.520s
DIFUSCO (T_s=50) [22]	SL+S+2Opt	16.694	0.893%	4.941s	23.425	1.326%	19.217s
Fast T2T (T_s=5) [24]	SL+S+2Opt	16.633	0.509%	0.911s	23.286	0.726%	3.733s
StruDiCO (T_s=5)	SL+S+2Opt	16.611	0.383%	0.857s	23.273	0.670%	3.541s
T2T (T_s=50,T_g=30) [23]	SL+S+2Opt	16.621	0.453%	12.636s	23.371	1.070%	36.271s
Fast T2T (T_s=5,T_g=5) [24]	SL+S+2Opt	16.580	0.194%	5.095s	23.287	0.419%	13.029s
StruDiCO (T_s=5,T_g=5)	SL+S+2Opt	16.575	0.168%	2.437s	23.178	0.261%	6.836s

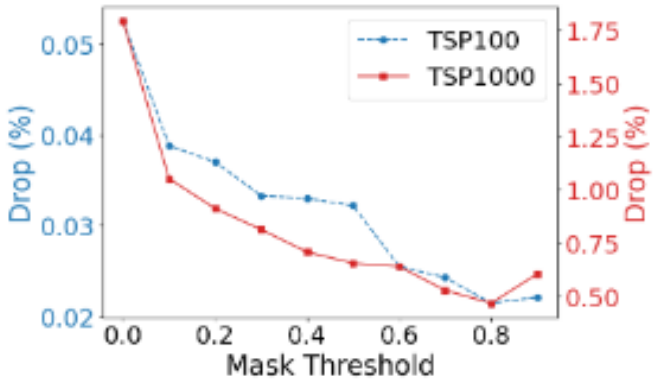
IV Experiment

❖ Hyperparameter Study

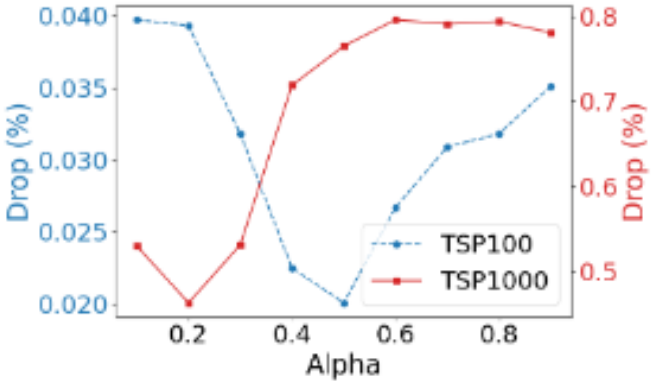
- ✓ StruDiCO의 핵심 하이퍼파라미터들의 영향 분석 (TSP-100, TSP-1000 데이터셋 사용)
- ✓ 분석 항목:
  - 추론 단계 및 정제 단계 (T_s, T_g):
    - 단계 수가 증가함에 따라 성능이 향상되다가, $T_s = T_g = 5$ 정도에서 포화됨
  - 마스크 임계값 (δ)
    - δ 값이 너무 낮으면 노이즈가 많아지고, 너무 높으면 탐색 공간이 제한됨 → $\delta = 0.8$ 이 좋은 성능을 보임
  - 교란 비율 (α)
    - α 값이 너무 작으면 후보들을 충분히 탐색하지 못하고, 너무 크면 기존에 형성된 구조 파괴 → $\alpha \in [0.2, 0.5]$ 범위의 적절한 값 설정



(a) Inference step.



(b) Mask threshold.



(c) Perturbation ratio α.

IV Experiment

❖ Generalization

- ✓ 훈련-테스트 스케일 변화 (Cross-Scale Generalization):
  - TSP: TSP-1000으로 훈련 후 TSP-50 테스트 시, StruDiCO (0.80% drop) < Fast T2T (1.26%)
  - StruDiCO가 다양한 문제 크기에 걸쳐 더 나은 일반화 성능을 보임
- ✓ MIS: 다양한 그래프 밀도 및 크기에 대한 일반화:
  - ER 그래프 (p=0.15)로 훈련 후, 다른 p 값 및 노드 수의 그래프 테스트
  - StruDiCO가 이전 확산 기반 솔버 대비 우수한 성능을 보임

Testing \ Training		TSP-50	TSP-100	TSP-500	TSP-1K
TSP-50	DIFUSCO (T_s=50)* [22]	0.09%	0.25%	2.55%	2.71%
	T2T (T_s=50, T_g=30)* [23]	0.02%	0.11%	1.60%	1.10%
	Fast T2T (T_s=5, T_g=5) [24]	0.09%	0.36%	1.02%	1.26%
	StruDiCO (T_s=5, T_g=5)	**0.01%**	**0.01%**	**0.31%**	**0.80%**
TSP-100	DIFUSCO (T_s=50)* [22]	1.44%	0.23%	3.44%	3.31%
	T2T (T_s=50, T_g=30)* [23]	0.56%	0.17%	2.47%	2.19%
	Fast T2T (T_s=5, T_g=5) [24]	**0.12%**	0.02%	0.40%	0.55%
	StruDiCO (T_s=5, T_g=5)	0.16%	**0.02%**	**0.34%**	**0.55%**
TSP-500	DIFUSCO (T_s=50)* [22]	4.16%	3.04%	1.40%	1.85%
	T2T (T_s=50, T_g=30)* [23]	3.79%	2.25%	0.91%	1.22%
	Fast T2T (T_s=5, T_g=5) [24]	**2.67%**	**1.77%**	0.38%	0.95%
	StruDiCO (T_s=5, T_g=5)	3.46%	2.96%	**0.33%**	**0.49%**
TSP-1K	DIFUSCO (T_s=50)* [22]	4.54%	3.98%	2.65%	2.21%
	T2T (T_s=50, T_g=30)* [23]	4.66%	3.81%	1.61%	1.30%
	Fast T2T (T_s=5, T_g=5) [24]	**3.46%**	**3.08%**	1.06%	0.58%
	StruDiCO (T_s=5, T_g=5)	4.31%	3.22%	**0.75%**	**0.54%**

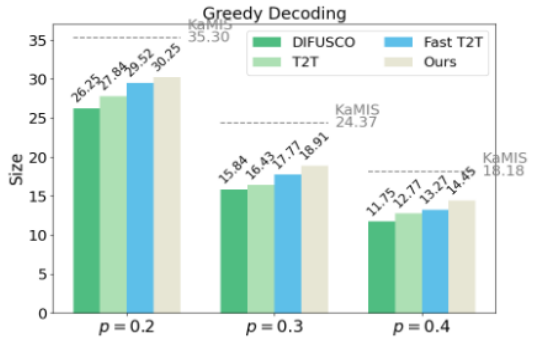


Figure 6: Generalization performance from $p = 0.15$ to $p = 0.2$, $p = 0.3$, and $p = 0.4$.

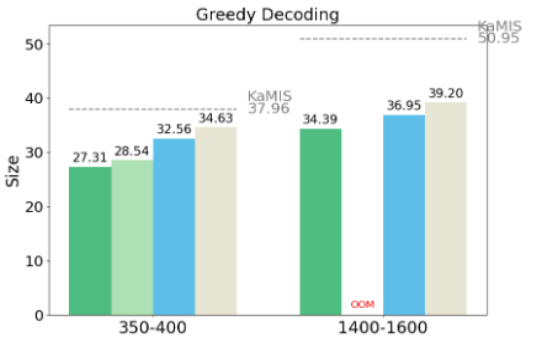


Figure 7: Generalization performance from ER 700–800 to ER 350–400 and 1400–1600.

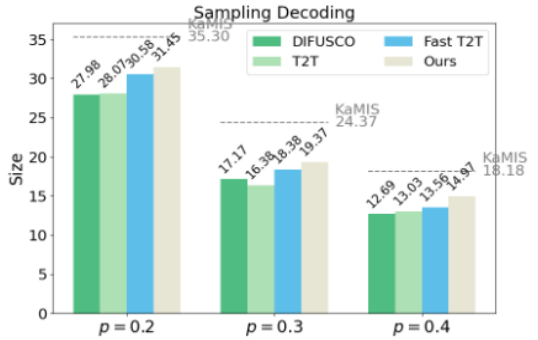


Figure 8: Generalization performance from $p = 0.15$ to $p = 0.2$, $p = 0.3$, and $p = 0.4$.

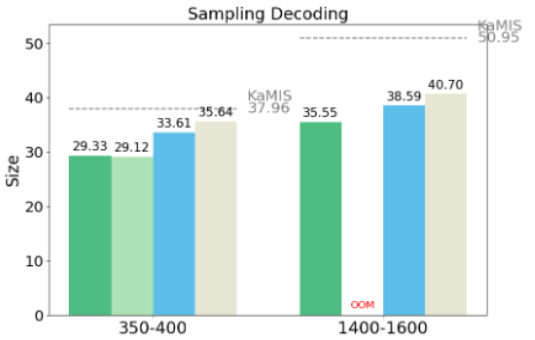


Figure 9: Generalization performance from ER 700–800 to ER 350–400 and 1400–1600.

IV Experiment

❖ Ablation Study

- ✓ 목적: Variable-Absorption (VA) 메커니즘과 Constrained Consistency Sampling (CCS)의 개별적 기여도 분리 및 분석
- ✓ Variable-Absorption (VA) / Constrained Consistency Sampling (CCS) / Consistency Model (CM)
- ✓ 실험 설정:
  - ✓ Uniform CM (Fast T2T baseline): 일반적인 균일 노이즈 diffusion 모델 (VA 없이)
  - ✓ VA CM (StruDiCO's VA only): Uniform CM을 Variable-Absorption CM으로 대체
  - ✓ Uniform + CCS: Uniform CM에 Constrained Consistency Sampling 추가
  - ✓ VA + CCS (StruDiCO): Variable-Absorption CM에 Constrained Consistency Sampling 추가 (완전한 StruDiCO)
- ✓ 주요 결과:
  - ✓ VA만으로는 성능 저하 발생 (안정적이나 탐색 범위 제한)
  - ✓ CCS는 Uniform 모델에서도 성능 향상
  - ✓ VA와 CCS 결합 시 가장 큰 성능 향상

Variant	TSP-50	TSP-100	TSP-500	TSP-1000	MIS-RB	MIS-ER
Uniform CM (Fast T2T, T_s=5)	0.031%	0.101%	0.922%	1.167%	6.023%	15.898%
VA CM (StruDiCO)	0.032%	0.163%	1.091%	1.183%	7.476%	23.461%
Uniform + CCS	0.025%	0.081%	0.898%	1.074%	5.878%	16.522%
VA + CCS (StruDiCO, T_s=5)	0.019%	0.067%	0.728%	0.797%	3.899%	11.450%

IV Experiment

❖ Supplementary Experiments

- ✓ Capacitated Vehicle Routing Problem (CVRP)
- ✓ 실제 데이터셋 평가 (TSPLIB)
  - 학습: 무작위로 생성된 100개의 노드를 가진 TSP 문제
  - 테스트: 50개에서 200개 사이의 노드를 가진 TSP 인스턴스

Algorithm	Type	CVRP 50			CVRP 100			CVRP 200		
		Length↓	Drop↓	Time	Length↓	Drop↓	Time	Length↓	Drop↓	Time
HGS [46]	Heuristics	10.37	0.00%	1s	15.56	0.00%	20s	19.63	0.00%	60s
Sym-NCO [16]	RL+LS	10.57	1.95%	0.09s	15.93	2.37%	0.19s	20.19	2.86%	0.36s
COExpander [47]	SL+LS	10.77	3.90%	0.04s	16.22	4.25%	0.06s	20.59	4.89%	0.15s
StruDiCO (T_s=3,T_g=3)	SL+LS	10.48	1.12%	0.05s	15.85	1.88%	0.11s	20.25	3.19%	0.26s
StruDiCO (T_s=5,T_g=5)	SL+LS	10.45	0.85%	0.07s	15.80	1.53%	0.17s	20.16	2.71%	0.38s

Instances	AM*	GCN*	Learn2OPT*	GNNGLS*	DIFUSCO*	T2T*	Fast T2T*	Ours
eil51	16.767%	40.025%	1.725%	1.529%	2.82%	0.14%	0.00%	0.00%
berlin52	4.169%	33.225%	0.449%	0.142%	0.00%	0.00%	0.00%	0.00%
st70	1.737%	24.785%	0.040%	0.764%	0.00%	0.00%	0.00%	0.00%
eil76	1.992%	27.411%	0.096%	0.163%	0.34%	0.00%	0.00%	0.00%
pr76	0.816%	27.793%	1.228%	0.039%	1.12%	0.40%	0.00%	-0.00%
rat99	2.645%	17.633%	0.123%	0.550%	0.09%	0.09%	0.00%	0.00%
kroA100	4.017%	28.828%	18.313%	0.728%	0.10%	0.00%	0.00%	0.00%
kroB100	5.142%	34.686%	1.119%	0.147%	2.29%	0.74%	0.65%	0.00%
kroC100	0.972%	35.506%	0.349%	1.571%	0.00%	0.00%	0.00%	0.00%
kroD100	2.717%	38.018%	0.866%	0.572%	0.07%	0.00%	0.00%	0.00%
kroE100	1.470%	26.589%	1.832%	1.216%	3.83%	0.27%	0.13%	2.15%
rd100	3.407%	50.432%	1.725%	0.003%	0.08%	0.00%	0.00%	0.11%
eil101	2.994%	21.776%	1.529%	0.03%	0.03%	0.00%	0.00%	0.00%
lin105	1.739%	34.902%	1.867%	0.606%	0.00%	0.00%	0.00%	0.18%
pr107	3.933%	80.564%	0.898%	0.439%	0.91%	0.61%	0.62%	0.18%
pr124	2.677%	70.146%	10.232%	0.755%	1.02%	0.08%	0.08%	-0.00%
bier127	5.908%	45.561%	3.044%	1.948%	0.94%	0.54%	1.50%	0.04%
ch130	3.182%	39.090%	0.709%	3.519%	0.29%	0.06%	0.00%	0.24%
pr136	5.064%	58.673%	0.000%	3.387%	0.19%	0.10%	0.01%	0.04%
pr144	7.641%	55.837%	1.526%	3.581%	0.80%	0.50%	0.39%	0.00%
ch150	1.584%	49.743%	0.321%	2.113%	0.57%	0.49%	0.00%	0.04%
kroA150	3.784%	45.411%	0.724%	2.984%	0.34%	0.14%	0.00%	0.24%
kroB150	2.437%	56.745%	0.886%	3.258%	0.30%	0.00%	0.07%	0.00%
pr152	7.494%	49.376%	3.119%	3.119%	1.69%	0.83%	0.19%	0.69%
u159	7.551%	38.338%	0.054%	1.020%	0.82%	0.00%	0.00%	0.00%
rat195	6.893%	24.968%	0.743%	1.666%	1.48%	1.27%	0.79%	-0.00%
d198	373.020%	62.952%	0.522%	4.727%	3.32%	1.97%	0.86%	-0.00%
kroA200	7.106%	40.885%	1.441%	2.029%	2.28%	0.57%	0.49%	0.00%
kroB200	8.541%	43.643%	0.646%	2.589%	2.35%	0.92%	2.50%	-0.00%
Mean	**16.767%**	**40.025%**	**1.725%**	**1.529%**	**0.97%**	**0.35%**	**0.28%**	**0.13%**

V Conclusion

❖ Methodological Contributions

- ✓ Variable-Absorption Noise: 순방향에서 변수를 점진적으로 비활성화하여 구조적 연속성과 부분해 유효성 유지
- ✓ Constrained Consistency Sampling: 역방향에서 신뢰도 높은 변수만 선택하여 안정적·해석 가능한 해 구성
- ✓ Objective-Aware Gradient-Free Refinement: 그래디언트 없이 목표 함수 기반 점수로 반복 정제 → 시간·메모리 효율 극대화

❖ Key Results

- ✓ TSP, MIS에서 기존 SOTA 대비 우수한 해 품질 달성
- ✓ 3.5X 빠른 추론 속도, GPU 메모리 사용량 70% 절감
- ✓ 단계별 해 구조에 대한 해석 가능성 확보

❖ Limitations

- ✓ 최적 또는 준최적 해에 대한 대규모 레이블 데이터가 필요하며, 고품질 해를 얻기 어려운 도메인에서는 적용이 제한적임
- ✓ 고정된 문제 크기에서는 일반화 성능을 보이나, 크기가 가변적인 실제 CO 문제에 대한 대응에는 한계가 존재함
- ✓ Non-Autoregressive 구조로 인해 최종 해의 feasibility 확보를 위해 greedy decoding과 같은 휴리스틱 후처리에 의존

V Network Architecture (Appendix)

❖ Input Embedding Layer

- ✓ 노드, 엣지, 타임스텝 정보를 Sinusoidal Positional Encoding 방식으로 임베딩
- ✓ Sin/Cos 함수를 이용해 시간적/위치적 정보를 벡터화
- ✓ 선형 변환 및 ReLU, BN을 거쳐 GNN 레이어 입력 준비

❖ Graph Convolution Layer

- ✓ 12개 레이어로 구성 (기본 GCN 연산 + 시간 정보 통합)
- ✓ 노드 및 엣지 특징을 반복적으로 업데이트하며 정보 집계
- ✓ TSP (Edge-centric):
  - ✓ 타임스텝 임베딩을 엣지 특징 업데이트에 직접 통합
  - ✓ 엣지 선택(경로) 정보에 시간적 변화를 강하게 반영
- ✓ MIS (Node-centric):
  - ✓ 타임스텝 임베딩을 노드 특징 업데이트에 직접 통합
  - ✓ 노드 선택 정보에 시간적 변화를 강하게 반영

❖ Output Layer

- ✓ GNN 최종 레이어의 특징 벡터를 바탕으로 확률 히트맵 생성 → Softmax 함수 사용