인공지능 실습 Chapter 6

Constraint Satisfaction Problem

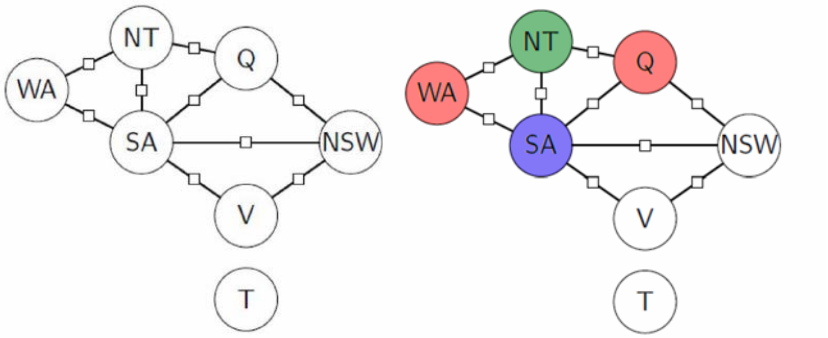
포항공과대학교 컴퓨터공학과

CSP Basic

- **Variable** : Domain(가능한 값의 집합)으로부터 값을 할당 받는 변수
- **Factor (or Constraint)** : variable들 사이의 관계를 나타내는 점수
- **Factor graph** : variable과 factor (or constraint)를 나타낸 그래프
- Weight 값을 최대로 만드는 (모든 제한조건을 만족하는) assignment 탐색

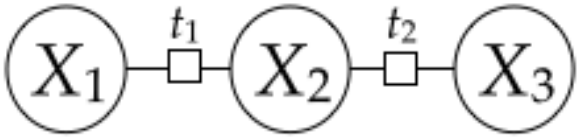
$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

- Example: map coloring problem
  - $X = (\text{WA}, \text{NT}, \text{SA}, \text{Q}, \text{NSW}, \text{V}, \text{T}) \in \{R, G, B\}$
  - $f_1(X) = [\text{WA} \neq \text{NT}]$
  - $f_2(X) = [\text{NT} \neq \text{Q}]$
  - …



CSP Example : Chain CSP

- **Variables (n=3)**
  - $X_1, X_2, X_3 \in \{0, 1\}$
- **Factors (or constraints)**
  - $t_1(X_1, X_2) = x_1 \oplus x_2$
  - $t_2(X_2, X_3) = x_2 \oplus x_3$



- **가능한 assignment는 총 몇 개인가?**
  - $\{X_1: 1,\ X_2: 0,\ X_3: 1\}$ $\{X_1: 0,\ X_2: 1,\ X_3: 0\}$: 2개
- **Backtrack()의 call stack을 그려보자 (Fixed ordering: X_1, X_3, X_2)**
  -

$$\{[0,1],[0,1],[0,1]\} \xrightarrow{X_1=0} \{0,[0,1],[0,1]\} \xrightarrow{X_3=0} \{0,[0,1],0\} \xrightarrow{X_2=1} \mathbf{\{0,\ 1,\ 0\}}$$

$$\{0,[0,1],[0,1]\} \xrightarrow{X_3=1} \{0,[0,1],1\}$$

$$\xrightarrow{X_1=1} \{1,[0,1],[0,1]\} \xrightarrow{X_3=0} \{1,[0,1],0\}$$

$$\{1,[0,1],[0,1]\} \xrightarrow{X_3=1} \{1,[0,1],1\} \xrightarrow{X_2=0} \mathbf{\{1,\ 0,\ 1\}}$$

A. Chain CSP 구현

- n개의 variable을 가지는 Chain CSP를 구현해보자
- **create_chain_csp()** 함수를 구현

1. CSP 객체에 정의한 변수들을 추가한다
2. CSP 객체에 변수들 사이의 factor (or constraint) 함수들을 추가한다
   - Python lambda 함수를 사용하여 factor 함수를 정의하자

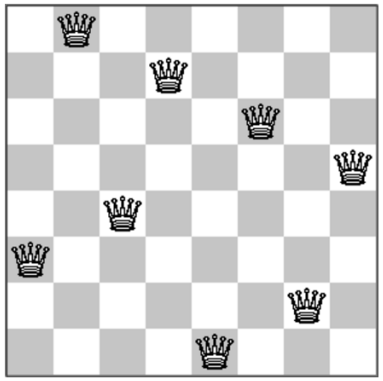
- 먼저, util.py에 구현된 CSP 클래스와 주요 내부 함수들을 살펴보자
  - add_variable(var, domain)
  - add_unary_factor(var, factor_func)
  - add_binary_factor(var1, var2, factor_func)

B. n-Queens CSP 구현

- n개의 queen을 가지는 n-Queens CSP를 구현해보자
- **create_nqueens_csp()** 함수를 구현

- n × n 보드에서 n개의 queen을 아래의 제한조건을 만족시키도록 배치
  - 어떤 두 queen도 같은 행에 있으면 안됨
  - 어떤 두 queen도 같은 열에 있으면 안됨
  - 어떤 두 queen도 대각선 방향으로 놓여있으면 안됨

- 힌트
- 각 queen의 위치를 나타내는 n개의 variable을 정의하자
- Queen들 위치 간의 binary factor를 추가하자



Backtrack 탐색 시간 개선

- **Dynamic Ordering**
  - Assign하는 variable의 순서를 동적으로 조절하는 방법
  - MCV (most constrained variable)
  - LCV (least constrained variable)

- **Arc Consistency**
  - 현재 assignment에서 arc consistent하지 않은 value들을 모두 제거하여 후보군을 줄이는 방법
  - AC-3 algorithm

C. MCV Heuristic 구현

- MCV (Most constrained variable) heuristic을 구현해보자
- BacktrackingSearch 클래스 내 **get_unassigned_variable()** 함수를 구현

- **get_unassigned_variable()**
  - 현재까지 assign되지 않은 variable 중에서 다음으로 assign을 수행할 variable을 선택하는 함수
  - **with MCV** – 남아있는 domain value의 수가 가장 적은 variable을 다음 variable로 선택하는 전략

- 먼저 backtrack() 함수 구현을 살펴보고, 어떤 식으로 동작하는지 확인
- **MCV**를 사용하는 경우 backtrack() 호출 횟수는 어떻게 되는가?

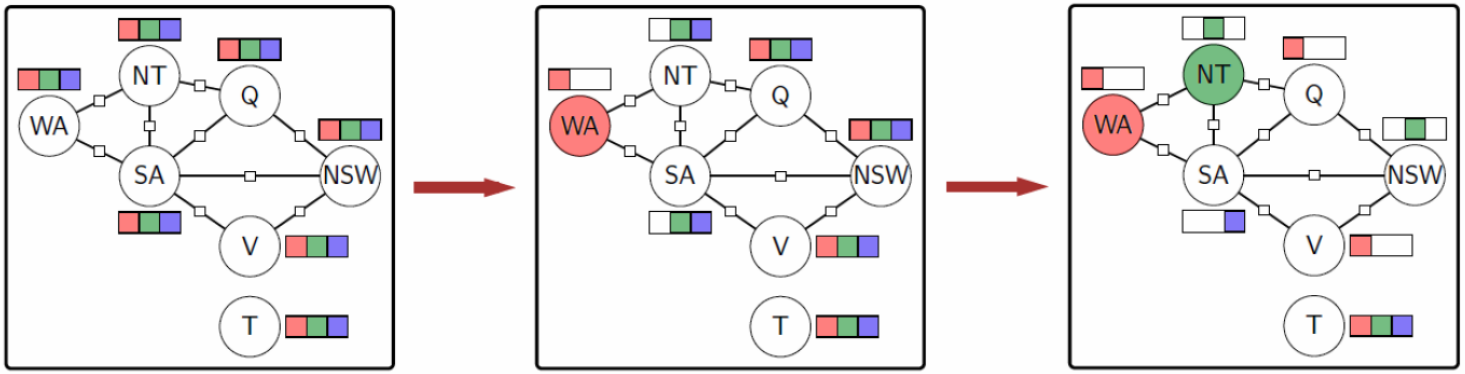
AC-3 Algorithm

- 각 variable에 value가 assign 될 때마다 알고리즘을 수행하여 후보군 축소
- Variable X_j가 assign된 경우,

Algorithm: AC-3

- Add X_j to set.
- While set is non-empty:
  - Remove any X_j from set.
  - For all neighbors X_i of X_j:
    - Enforce arc consistency on X_i w.r.t. X_j.
    - If Domain_i changed, add X_i to set.

- Ex. Map coloring problem



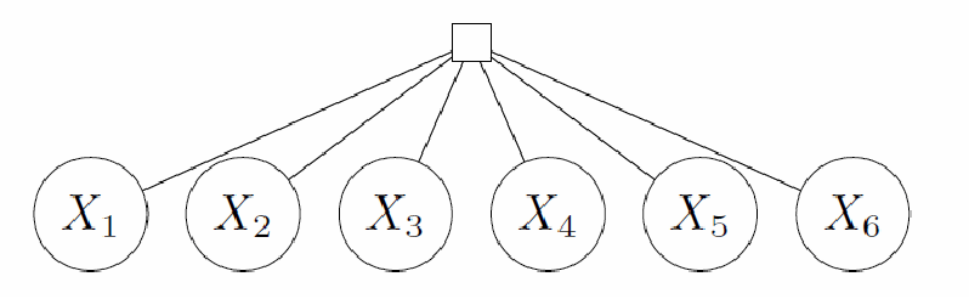
D. AC-3 Algorithm 구현

- AC-3 algorithm을 구현해보자
- BacktrackingSearch 클래스 내 **arc_consistency_check()** 함수를 구현

- 먼저 backtrack() 함수 구현을 살펴보고, 어떤 식으로 동작하는지 확인
  - 각 variable이 assign된 이후 **arc_consistency_check()** 함수가 호출

- **AC-3**를 사용하는 경우 backtrack() 호출 횟수는 어떻게 되는가?

n-ary factor 변환

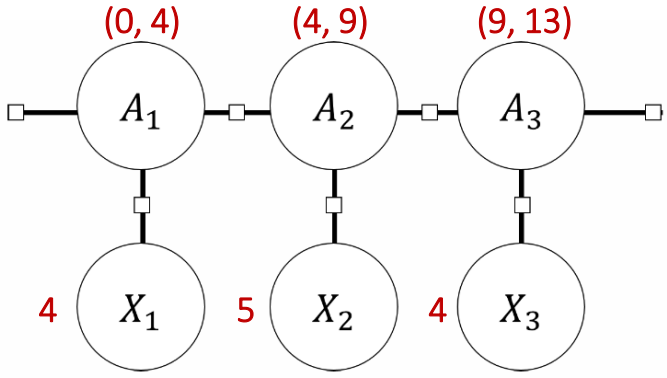
- **n-ary factor**
  - n개의 variable들과 연관된 factor
  - 

- 현재 탐색 알고리즘은 unary factor, binary factor만 처리 가능
- Unary factor, binary factor로 이루어진 factor graph로 변환 필요
- **Example**
  - 세 variable 값의 합이 13, 14, 15 중 하나여야 함
  - Variables : $X_1, X_2, X_3 \in \{1, 2, 3, 4, 5\}$
  - Ternary factor : $f = [X_1 + X_2 + X_3 = 13\ or\ 14\ or\ 15]$

n-ary factor 변환

- **새로운 variable A_1, A_2, A_3를 정의**
  - 각 variable A_i는 이전까지 variable 값들의 합 $\sum_{j=1}^{i} X_j$ 을 나타냄
  - 이전 variable과 관계를 나타내기 위해 pair를 값으로 가짐
  - Domain = $\{(m,n) \mid 0 \le m,n \le 15\}$ (m,n은 정수)
- **새로운 unary/binary factor를 정의**
  - $A_i[1] = A_i[0] + X_i$
  - $A_{i+1}[0] = A_i[1]$
- **새로운 factor graph**

One of possible assignment



E. Sum Variable 구현

- n개의 variable이 주어졌을 때, 모든 variable 값의 합을 나타내는 새로운 variable을 추가해보자
- **get_sum_variable()** 함수를 구현

- 테스트케이스 : $\boldsymbol{A + B + C = 12}\ or\ \boldsymbol{13}$

```
print "A = {0, 1, 2, 3}"
print "B = {0, 6, 7}"
print "C = {0, 5}"

csp = util.CSP()
csp.add_variable('A', [0, 1, 2, 3])
csp.add_variable('B', [0, 6, 7])
csp.add_variable('C', [0, 5])
sumVar = get_sum_variable(csp, 'sum-up-to-15', ['A', 'B', 'C'], 15)

csp.add_unary_factor(sumVar, lambda n: n in [12, 13])
sumSolver = BacktrackingSearch()
sumSolver.solve(csp)
```

Course Scheduling

- 자동으로 원하는 수강 시간표를 찾아주는 CSP를 정의해보자

- course.json: 모든 과목들에 대한 정보
- profile*.txt: 사용자가 원하는 수강 계획 (과목에 대한 제약 조건, 선호도 등)

```
# Unit limit per quarter. You can ignore this for the first
# few questions in problem 2.
minUnits 0
maxUnits 3

# These are the quarters that I need to fill out.
# It is assumed that the quarters are sorted in chronological order.
register Aut2015
register Win2016

# Courses I've already taken
taken CS106B
taken CS107

# Courses that I'm requesting
request CS229 or CS205A in Aut2015,Spr2016  # Can only be taken in Aut2015
request CS246 in Spr2016    # Unsatisfiable request
```

Course Scheduling

- 기본적으로 추가되어 있는 variable : **(req, quarter)**
- 해당 request에서 배정된 과목 id를 value로 가짐 (배정 안 된 경우 None)

- Register **Aut2015**, register **Win2016**
- req = "request **CS229** or **CS205A** in **Aut2015**, **Spr2016**"
  - **(req, Aut2015)** = '**CS229**' or '**CS205A**' or **None**
  - **(req, Win2016)** = '**CS229**' or '**CS205A**' or **None**

- 기본적인 constraints는 모두 구현되어 있음
- 추가적인 constraints들을 구현해보자
  - add_quarter_constraints()
  - add_unit_constraints()

F. Quarter constraints 추가하기

- 만약 요청된 과목을 수강할 경우, 해당 과목은 지정된 학기에 수강해야 함
- SchedulingCSPConstructor 안의 **add_quarter_constraints()** 함수를 구현

- 테스트 케이스: profile_f.txt
  - 3개의 optimal assignment (Weight = 1.0) 존재

- 힌트 : 불가능한 variable의 경우 None 값으로 assign되어야 함
- Ex. req = "request **CS245** in **Spr2016**"
  - **(req, Aut2015)** should be **None**
  - **(req, Win2016)** should be **None**
  - **(req, Aut2016)** should be **None**
  - **(req, Spr2016)** can be '**CS245**' or **None**

G. Unit constraints 추가하기

- 각 quarter에 할당된 과목들의 unit 합이 최소 unit값(minUnits)과 최대 unit값(maxUnits) 사이에 있어야 함
- SchedulingCSPConstructor 안의 **add_unit_constraints()** 함수를 구현

- 테스트 케이스: profile_g.txt
  - 15개의 optimal assignment (Weight = 1.0) 존재

- 힌트1 : 앞서 구현한 get_sum_variable() 함수를 활용하자
- 힌트2 : 새로운 **(courseID, quarter)** variable들을 추가하자
  - 해당 과목을 해당 quarter에 수강할 경우, 과목의 unit을 값으로 가짐
  - 해당 과목을 해당 quarter에 수강하지 않을 경우, 0을 값으로 가짐

G. Unit constraints 추가하기

- Ex. req1 = “request **CS148**”, CS148을 Aut2016에 배정할 경우
  - 기본 변수
    - (req1, Aut2016) = ‘**CS148**’
    - (req1, Win2017) = **None**
  - 추가 변수
    - (CS148, Aut2016) = **3**
    - (CS148, Win2017) = **0**
  - v1 = get_sum_variable(csp, ‘quarter_unit_sum’, [**(CS*, Aut2016)**], maxUnits)
  - v2 = get_sum_variable(csp, ‘quarter_unit_sum’, [**(CS*, Win2016)**], maxUnits)
  - v1과 v2가 minUnits와 maxUnits 사이의 값을 가지도록 unary factor를 추가