Index

Introduction

Objects

LP에 기반한 최적화 알고리즘을 시각화하여 certifying algorithm의 특징을 활용하는 동시에 사용자의 알고리즘에 대한 이해를 돕는 것이 목표이다.

Necessity

우리는 종종 자료구조나 알고리즘의 동작 과정이나 원리를 이해할 때 도면이나 애니메이션 등 시각적 요소의 도움을 많이 받는다. 영문 위키피디아에서 각종 정렬 알고리즘 페이지마다 저마다의 동작 과정을 애니메이션화하여 보여주는 것이나, VISUALGO 등의 사이트에서 각종 자료구조 및 알고리즘의 동작 과정을 입력에 대해 시각화하여 보여주는 것이 대표적인 예다.

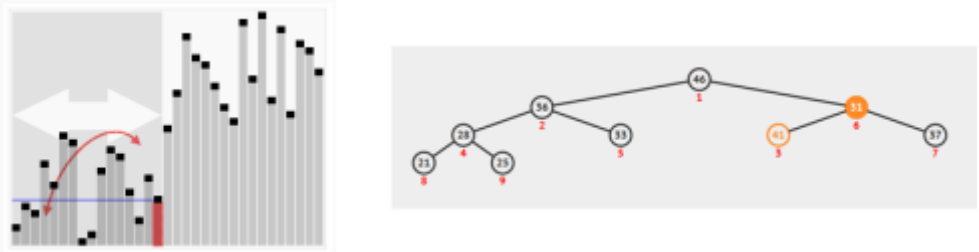

Figure 1 Wikipedia의 quick sort(좌), VISUALGO의 binary heap(우)

구현자는 이러한 결과물을 구현하기 위하여 어떻게 하면 보다 효과적으로 보는 이에게 알고리즘을 직관적으로 이해시킬지 고민해야 하며, 이는 구현자가 알고리즘에 대해 잘 이해하고 있어야 가능하다. 또한 매번 결과물을 더 나은 형태로 보완하는 과정이 여러 번 반복될 수 있으며 이 과정에서 구현자 또한 알고리즘에 대해 더 깊이 이해하는 순기능이 존재한다.

또한 우리가 관심을 가지는 알고리즘은 시각화한 동작 과정을 보는 것이 곧 이 알고리즘의 정당성도 동시에 직관적으로 이해시킬 수 있는 것들이다. 또한 시각화 결과물이 도출한 결과가 문제의 답과 일치한다는 점 역시 직관적으로 이해하기 쉽다. 따라서 사용자 또한 알고리즘의 동작 과정과 결과를 봄으로써 각각의 정당성을 쉽게 이해할 수 있다.

Backgrounds

수행하면 결과와 동시에 그 결과의 정당성을 증명할 수 있는 또다른 결과를 추출하는 알고리즘을 certifying algorithm이라고 한다. 우리는 이러한 certifying algorithm들을 타깃으로 하여 시각화한 결과가 곧바로 그 알고리즘이 정당함을 직관적으로 이해할 수 있게 하였다.

타깃으로 삼은 문제는 minimum spanning tree 문제와 perfect bipartite matching 문제이며, 알고리즘들은 primal-dual analysis에 기반하여 연구하고 시각화하였다.

Primal-dual Analysis

Primal-dual analysis는 primal LP와 dual LP의 variable들을 동시에 조작하여 optimal value에 도달하는 형태를 하고 있다. 이는 아래의 LP duality의 성질에 기반한다. 또한, 우리는 이러한 analysis가 적용 가능한 형태의 문제들을 연구하였다.

LP duality

모든 LP 문제에는 그 dual이 존재하며, primal LP와 dual LP 사이에는 weak duality와 strong duality가 성립한다. primal LP가 minimize 문제이고 dual LP가 maximize 문제라고 가정하자. 이때 primal LP의 모든 feasible한 objective function value는 항상 dual LP의 optimum보다 크거나 같다는 것이 weak duality이며, primal LP와 dual LP의 optimum이 일치한다는 것이 strong duality이다.

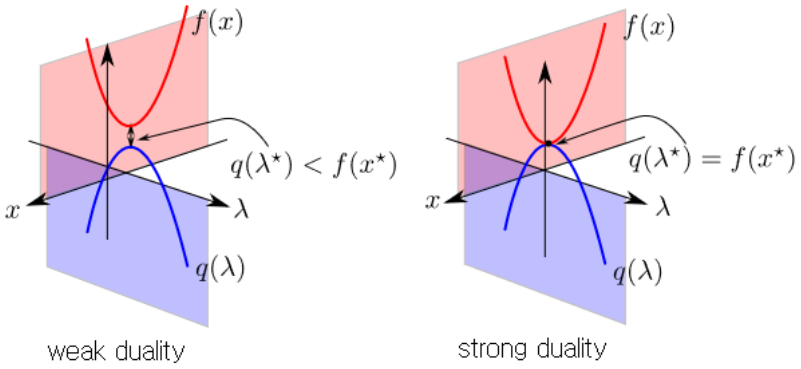

Figure 2 Weak duality(좌)와 strong duality(우)를 표현한 그림

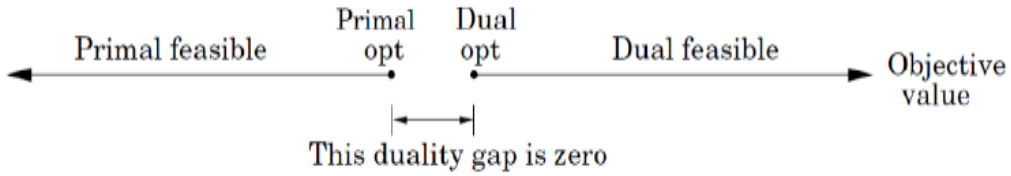

Figure 3 Strong duality(duality gap = 0)의 성립

Complementary slackness

Primal LP의 한 solution x^* 가 있을 때, 어떤 한 constraint와, 그에 대응하는 dual variable이 속한 constraint 둘 중 하나가 tight하다면 상대편 variable은 0이 아니라는 theorem이다. 이렇게 한쪽의 variable의 값들로 반대쪽 variable의 값들도 추론이 가능한 경우가 많다.

Primal-dual analysis

먼저 모든 variable의 값을 0으로 초기화한다. 이때 primal side는 infeasible하고 dual side는 feasible한 상태이다. 이제 dual variable 값들을 조건을 만족하면서 늘려가다가 primal side가 feasible해지는 순간 종료한다. Strong duality에 의해, primal과 dual이 모두 feasible한 지점이 optimum이므로 이 때의 objective function 값이 결과가 된다.

Figure 4 Primal-dual analysis의 시간 경과에 따른 동작 과정

보통 이러한 과정에는 보다 직관적인 설명을 위해 시간 개념을 의미하는 notation을 동반하기도 한다. 가장 단순한 형태에서는 dual variable의 일부가 그때까지 흐른 시간과 일치하기도 한다. 보는 이의 이해를 돕기 위해 시간과 동일하게 증가하는 값들을 따로 시각화하여 optimum을 표현하는 것도 좋은 방법이다. 알고리즘에 따라서 dual variable을 증가시키는 과정에서 complementary slackness에 의해 primal side의 constraint들이 tight해지는 여부를 알 수 있고, 이 때문에 primal side와 dual side를 동시에 관찰할 수 있게 된다.

또한 primal LP를 사람이 이해할 수 있도록 표현할 수 있다면 dual LP 역시 각 variable에 의미를 부여하여 표현할 수 있다. 예를 든다면 dual variable 각각이 어떤 event를 위해 지불하는 비용이라고 표현하고, 그 외 부등식들은 이때 지불한 값이 충분히 모여야만 event가 일어날 수 있다고 해석할 수 있다.

The Minimum Spanning Tree Problem

첫 번째 타깃은 최소 스패닝 트리를 구하는 문제이다. Primal-dual analysis를 적용할 경우 그 과정이 Kruskal's algorithm과 일치한다. 이 문제를 LP로 표현한 후 전술한 primal-dual analysis를 사용하여 답을 도출하고 시각화한다.

Primal & dual LP formulations

최소 스패닝 트리 문제를 LP의 형태로 나타낸다면 **Figure 5**와 같다. 어떠한 파티션 $\mathcal{P}$가 있을 때 간선 e의 양 끝점이 서로 다른 영역에 존재할 경우 $e \in \delta(\mathcal{P})$라 정의한다(즉 $\delta(\mathcal{P})$는 파티션 $\mathcal{P}$를 횡단하는 간선의 집합이다).

$$\begin{aligned} \textbf{min} \quad & \sum_e c_e x_e \\ \textbf{s.t.} \quad & \sum_{e \in \delta(\mathcal{P})} x_e \geq |\mathcal{P}| - 1 \quad \forall \mathcal{P} \\ & x_e \geq 0 \quad \forall e \end{aligned}$$

Figure 5 MST의 LP form

LP variable은 각 간선마다 존재하는 x_e이며, 이 variable의 값이 0이라면 해당 간선을 선택하지 않았다는 의미이며, 1이라면 선택했다는 의미다. c_e는 해당 간선의 cost이다. 이제 objective function은 선택한 간선들의 cost 합을 의미한다. n은 정점의 개수이다.

첫 번째 줄 부등식은 모든 형태의 파티션에 대해 횡단하는 간선이 파티션 크기-1개가 있어야 한다는 의미이다. 두 번째 줄 부등식은 본래의 식 $x_e \in \{0, 1\}$이 LP relaxation을 거친 것이다.

이제 이 LP의 dual을 찾으면 **Figure 6**과 같다.

$$\begin{aligned} \textbf{max} \quad & \sum_{\mathcal{P}} (|\mathcal{P}| - 1) y_{\mathcal{P}} \\ \textbf{s.t.} \quad & \sum_{\mathcal{P}: e \in \delta(\mathcal{P})} y_{\mathcal{P}} \leq c_e \quad \forall\, e \\ & y_{\mathcal{P}} \geq 0 \qquad\qquad\qquad \forall\, \mathcal{P} \end{aligned}$$

Figure 6 MST의 dual LP

Primal LP에서 모든 파티션 $\mathcal{P}$마다 부등식이 존재했으므로 각각 대응하는 dual variable $y_{\mathcal{P}}$이 발생한다. 이 변수에 의미를 부여한다면 각 파티션을 어떤 개체라 볼 때 이 개체가 MST를 연결하기 위해서 내는 비용이다. 비용이 충분히 모이면 컴포넌트들이 연결이 되고, 각 constraint들은 MST의 optimum보다 많은 비용을 내지는 않게 제어해 주고 있다.

첫 번째 줄 부등식이 이를 나타낸다. 간선 e는 자신이 횡단하는 파티션들로부터 비용을 받아서 연결될 수 있는데, 이때 받은 비용의 합이 본래 cost를 초과하면 안 된다. 즉 간선 e를 연결하는 데는 c_e의 비용이면 충분하므로, 그보다 많은 비용을 걷지는 못하게 하는 것이다.

Objective function의 경우 그냥 비용의 합이 아니라 각 변수에 $(|\mathcal{P}| - 1)$를 곱해 주는데, 이 식의 의미를 이해하는 데에 아래의 visualization 문단이 도움이 될 것이다.

Visualization

먼저 시각화하는 문제의 instance는 모두 2D Euclidean plane 위에 있다고 가정한다. 이는 각 정점을 점으로, 간선의 cost는 거리로 표현하기 쉽고, 보는 이 역시 이해하기가 쉽기 때문이다.

Dual side를 시각화하면 다음과 같은 형태가 된다.

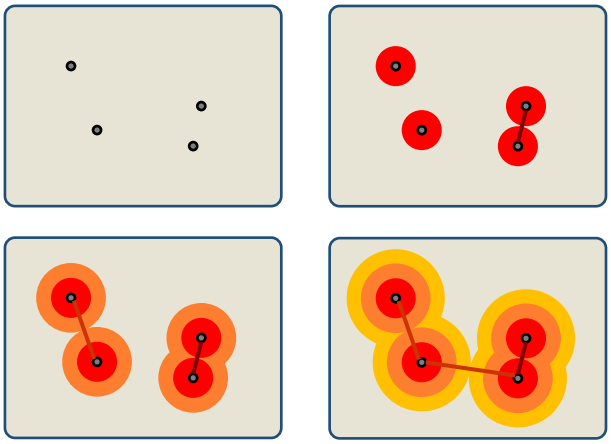

Figure 7 |V|=4인 한 예의 시각화

이러한 형태의 그림에서는 각 영역을 moat이라 부른다. Moat의 폭(width)은 안팎 영역의 임의의 두 점의 최단 거리로 생각하는데, 이 문제의 경우 moat의 형태 자체가 어느 곳에서나 일정한 폭을 가지고 있다. 또한 moat은 서로 떨어져 있는 두 개 이상의 영역을 합쳐서 칭할 수도 있다. 위 그림에서는 색으로 각각의 moat을 구분한다.

이 문제의 경우 정점마다 에워싼 원이 존재하는데, 처음엔 모든 원의 지름은 0이다. 동시에 모든 원의 지름을 증가시키다가 어떤 두 원이 만나게 되면 그 두 원을 연결한다. 이때 두 정점은 한 컴포넌트로 연결된다. 다음 스텝부터는 컴포넌트 단위로 지름을 증가시키며, 역시 두 컴포넌트가 접하게 되는 순간 접한 원 두 개를 연결함으로써 두 컴포넌트를 연결한다. 이 과정을 n-1번 반복하면 결과적으로 파티션 $\mathcal{P}$에 속하는 정점들은 $y_{\mathcal{P}}$ 값만큼 moat의 폭을 추가로 확장하는 형태가 되며, 정점들의 연결 관계가 곧 MST가 된다. 이때 원이 겹쳐서 moat의 형태를 이루게 되는데, 하나의 파티션에는 하나의 moat이 대응하게 되는 것이다.

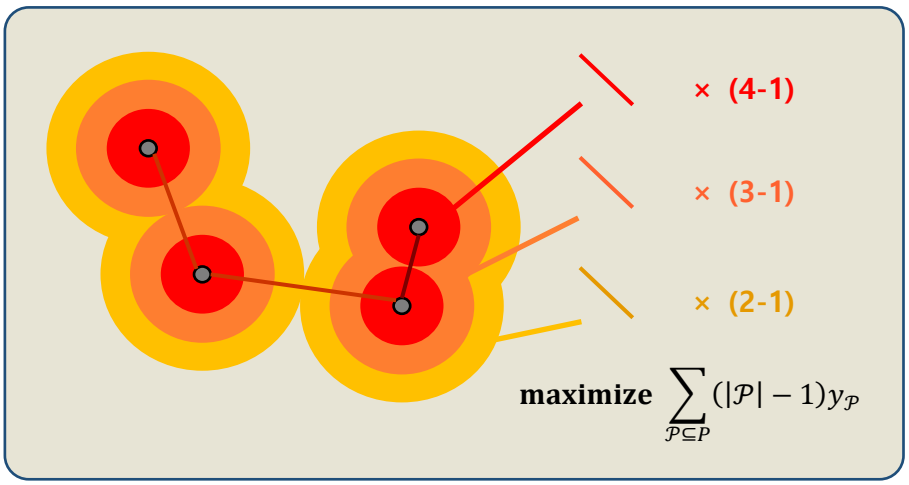

Figure 8 시각화 결과와 objective function의 관계

여기서 objective function의 의미를 파악할 수 있다. MST는 빨간색 moat을 6번, 주황색 moat을 4번, 노란색 moat을 2번 지나는 것을 알 수 있는데, 이를 일반화하면 파티션 $\mathcal{P}$의 크기가 k일 때 MST가 이 파티션에 해당하는 영역을 2(k-1)번 지나게 된다. $y_{\mathcal{P}}$ 을 해당 파티션의 width의 2배라 생각한다면, 각 파티션이 MST에 지불한 값은 $(|\mathcal{P}|-1)y_{\mathcal{P}}$ 로 일치하게 된다.

또한 시각화 과정 중에 연결되는 간선들의 순서를 보면, 이 시각화가 Kruskal's Algorithm 에 의한 MST 임을 직관적으로 알 수 있게 된다.

The Perfect Bipartite Matching Problem

두 번째 문제는 크기가 같은 두 그룹의 정점들을 complete 하게 매칭시키되, 그 간선 cost 의 총합이 최소가 되게 하는 문제이다. 역시 문제를 LP 로 표현한 후 primal-dual analysis 를 적용하여 답을 도출하고 시각화하였는데, 이 과정이 헝가리안 메소드와 유사하다.

Primal & dual LP formulations

이분 매칭의 완전 부합 문제를 LP 의 형태로 나타낸다면 **Figure 9** 와 같다. 크기가 동일한 두 정점 그룹 U, V가 있을 때, 각 그룹에 속하는 정점 원소를 u, v로 표현하였다.

$$
\begin{aligned}
\textbf{min} \quad & \sum_{u,v} c_{uv} x_{uv} \\
\textbf{s.t.} \quad & \sum_{v} x_{uv} = 1 \quad \forall\, u \\
& \sum_{u} x_{uv} = 1 \quad \forall\, v \\
& x_{uv} \geq 0 \quad \forall\, u, v
\end{aligned}
$$

Figure 9 Bipartite Matching 의 LP form

LP variable u 와 v 는 서로 다른 두 그룹에 속해 있는 점들이다. c_{uv} 는 해당 간선의 cost이다. x_{uv} 은 각 간선의 matching 여부이며, 이는 원래 0 또는 1이다. 0은 해당 간선을 선택하지 않았다는 의미이며, 1은 선택했다는 의미다. U 그룹의 점은 V 그룹의 단 하나의 점과만 만나기에 특정한 점 u에 대해 모든 x_{uv} 의 합은 1이 되게 된다. 이는 V 그룹의 점

에서도 통용이 된다. 이러한 cost와 matching 여부의 곱을 모두 더하게 되면 perfect bipartite matching의 최종 cost값을 알 수 있게 된다.

이제 이 LP의 dual을 찾으면 **Figure 10**과 같다.

$$\max \sum_u y_u + \sum_v y_v$$
$$\text{s.t.} \quad y_u + y_v \le c_{uv} \quad \forall u, v$$

Figure 10 Bipartite Matching 의 dual LP

Primal LP에서 모든 간선 마다 부등식이 존재했으므로 각각 대응하는 dual variable 이 발생한다. 이 변수는 U그룹의 한 점에 대해 V그룹의 모든 점과 간선이 연결이 되어있고, 이는 U그룹의 모든 점에게 동일하게 적용 된다고 가정할 때, 이 점들이 누군가와 하나의 matching을 이루기 위해서 내는 비용이다. 비용이 충분히 모이면 간선들이 연결이 되고, 각 constraint들은 간선 사이의 cost보다 많은 비용을 내지는 않게 제어해 주고 있다.

위의 부등식이 이를 나타낸다. 점 u의 cost인 y_u와 점 v의 cost인 y_v의 합은 점 사이에 존재하는 본래 cost를 초과하면 안 된다. 즉 점 u, v를 연결하는 데는 c_{uv}의 비용이면 충분하므로, 그보다 많은 비용을 걷지는 못하게 하는 것이다.

이러한 조건을 만족한 상태로 각 점들의 cost 들을 더하면 bipartite matching 의 optimal cost 를 구할 수 있게 되는 것이다.

Visualization

먼저 시각화하는 문제의 instance 는 모두 2D Euclidean plane 위에 있다고 가정한다. 이는 각 정점을 점으로, 간선의 cost 는 거리로 표현하기 쉽고, 보는 이 역시 이해하기가 쉽기 때문이다.

MST 문제보다는 시각화에 좀 더 난이도가 있는데, 그 이유는 dual variable 들의 값이 음수가 될 수 있기 때문이다.

Dual side를 시각화하면 다음과 같은 형태가 된다.

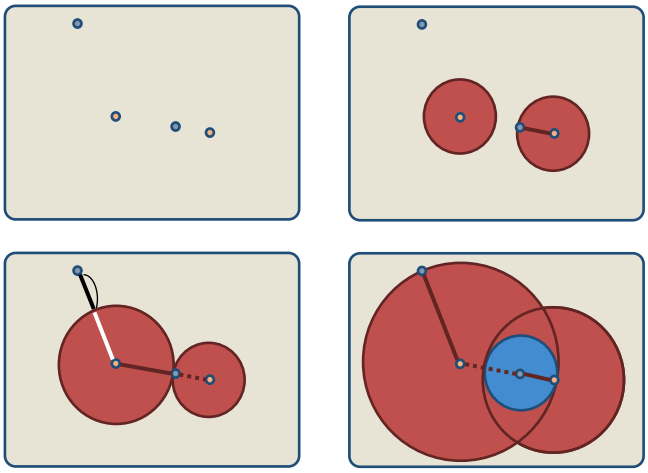

Figure 11 $\mathbf{U} = 2$, $\mathbf{V} = 2$ 인 한 예의 시각화

이번에도 각 영역을 moat이라 표현하는데, MST 문제보다는 단순한 형태로 각 정점을 중심으로 하는 원의 형태로 나타나며, width 또한 반지름으로 생각할 수 있다. 또한 MST 문제와는 다르게 여기서는 같은 색의 moat끼리는 영향을 주지 않고, 반지름이 음수값을 갖게 되는 moat도 존재한다.

처음엔 모든 원의 지름은 0이다. 동시에 U 그룹의 모든 원의 지름을 증가시키다가 어떤 원이 V 그룹의 점과 만나게 되면 이 원은 더 이상 커지지 않게 되고, 두 정점은 임시적으로 매칭을 이루게 된다. 다음 스텝에서는 U 그룹의 나머지 원들이 지름을 증가시키며, 역시 한 원이 V 그룹의 점과 만나게 되는 순간 반지름이 더 이상 커지지 않게 되고, 만약 만난 V 그룹의 점이 매칭을 이루지 않을 경우 서로 매칭을 이루게 된다. 그렇지 않을 경우 tight한 간선만 하나 생기게 된다. 이 과정을 U 그룹의 모든 점들이 tight 해질 때까지 반복하게 된다. 이 과정이 끝나면, U 그룹의 모든 점들은 일부 tight한 V 그룹의 점들과 함께 tree 형태를 이루게 되고, U 그룹의 모든 점들은 tight하지만 매칭을 이루지 못하는 점도 생기게 된다.

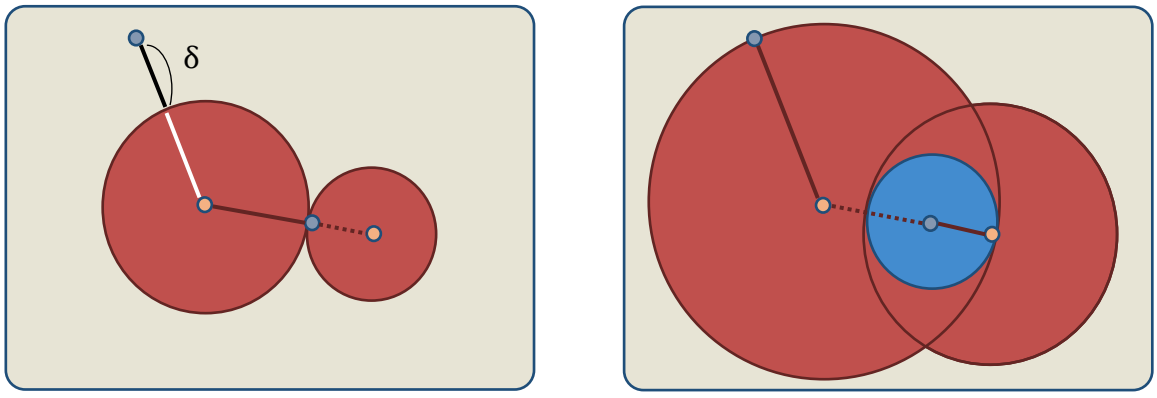

Figure 5 반지름이 음수인 원이 생기는 과정

이제부터 각 step마다 augmenting path를 찾아서 매칭 개수를 증가시키다가 그 수가 그룹

크기와 같아질 때 종료한다. Augmenting path는 매칭에 속한 간선과 속하지 않은 간선이 번갈아가면서 나타나며 양 끝점이 free vertex인 path인데, 이 path에 속한 모든 간선의 매칭 여부를 전부 뒤집는다면 매칭 개수가 1 늘어나게 된다.

누군가와 tight하지만 매칭엔 속하지 않은 U 그룹의 정점들과 아직 tight 하지 않은 V 그룹의 점 사이에 추가로 증가 가능한 cost들을 비교하여 가장 작은 값을 찾고 이를 δ라고 하자. U 그룹의 모든 원들은 δ만큼 반지름을 증가 시키고, tight한 V 그룹의 모든 원들은 이미 tight한 점들 사이의 cost를 유지하기 위해 δ만큼 반지름을 감소시켜 음수 값을 반지름으로 갖는 파란 원을 만든다. 이 과정을 통해 새롭게 tight해진 두 점은 매칭을 이루게 되는데, 이를 위해서는 기존에 매칭에 속해 있던 정점의 매칭 상대는 다른 정점을 찾아 매칭되어야 한다. 이러한 과정에 반복되어 tree안에 새로운 matching set이 생성된다. 이 과정은 U 그룹의 모든 점들이 매칭될 때까지 진행되고, 끝난 후 이 매칭에 속한 간선의 cost 합이 곧 perfect bipartite matching의 최소 cost가 된다.

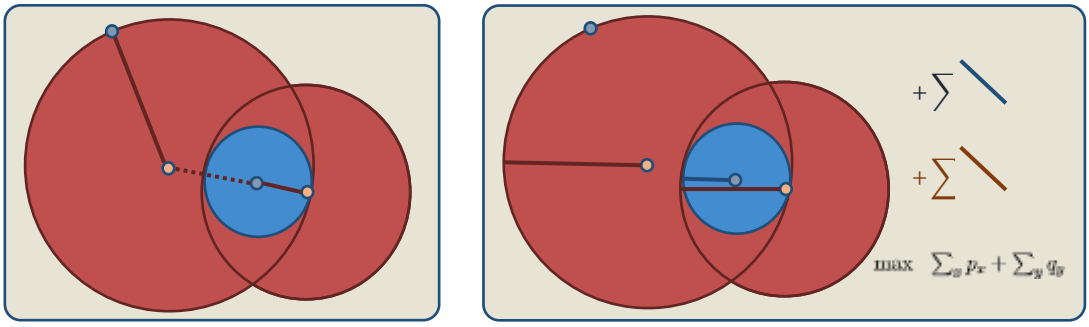

Figure 6 시각화 결과와 objective function 의 관계

결과적으로 반지름이 양수인 빨간 원들과 반지름이 음수인 파란 원들이 생기게 되는데, 여기서 objective function의 의미를 파악할 수 있다. 각 점들의 cost, 즉, 각 원들의 반지름의 길이인, y_u, y_v의 합을 더하면 매칭된 간선들의 길이의 합과 정확히 일치하게 된다.

Results

Unity 엔진을 사용하여 두 문제의 시각화를 구현하였고 성공적으로 build 및 export하여 결과물을 stand-alone한 실행 프로그램으로 추출하였다.

User Experience & Performance

UX의 측면 또한 고려하였다. 사용자가 정점 개수를 선택할 수 있고, 원하는 문제 instance를 얻도록 재설정하는 등의 버튼을 구현하여 UI를 지원한다. MST 문제의 경우를

예로 들자면 처음에는 사용자가 원하는 배치가 나올 때까지 정점 위치를 reset하는 버튼이 있으며, 원하는 배치가 나왔다면 재생 버튼으로 매 step을 관찰할 수 있다.

정점 개수가 수십~수백 개에 달하는 문제에 대해서도 무리없이 동작하는 것을 확인하였다. 아래의 **Figure 12**는 정점 개수가 50개인 instance를 test한 것으로, 별다른 오류나 성능 저하 없이 결과에 도달하는 것을 확인하였다.

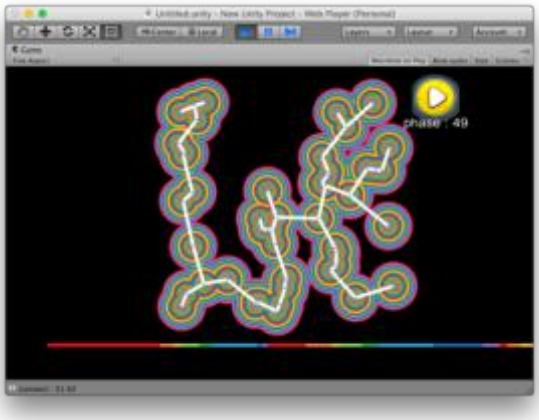

Figure 13 정점 개수가 50개인 MST 문제의 작동 사진

이렇게 충분히 큰 instance에 대해서도 잘 돌아가고 instance 크기의 제한이 없으나, 문제의 크기가 너무 크면 시각화 결과물 역시 복잡하여 이해 또한 까다로울 수 있으므로 작은 크기의 instance들에 대해서 사용하는 것을 권장한다.

Screenshots

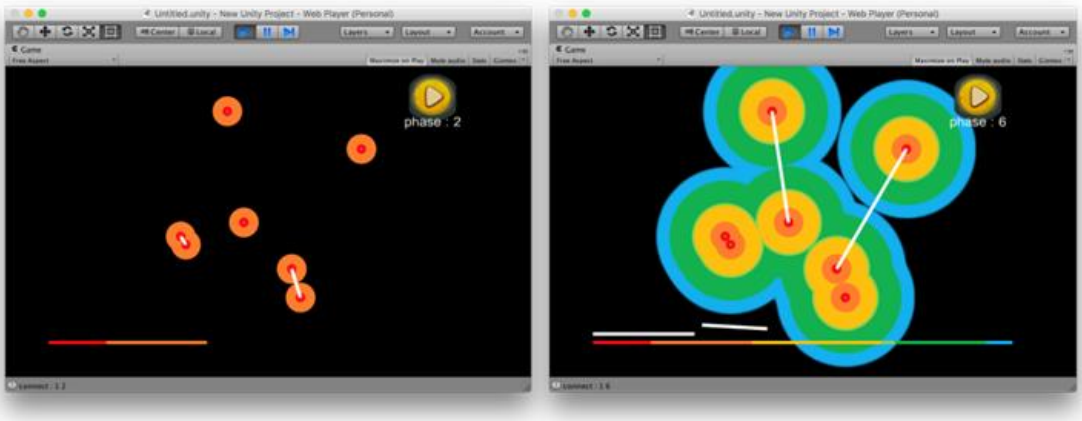

Figure 14 Minimum Spanning Tree 구현물

MST 문제의 경우 간선들을 옮겨서 dual의 objective function 값과 같다는 것을 보여준다. 이는 알고리즘 완료 후에 사용자가 한층 더 쉽게 결과의 정당성을 확인할 수 있는 장치이다.

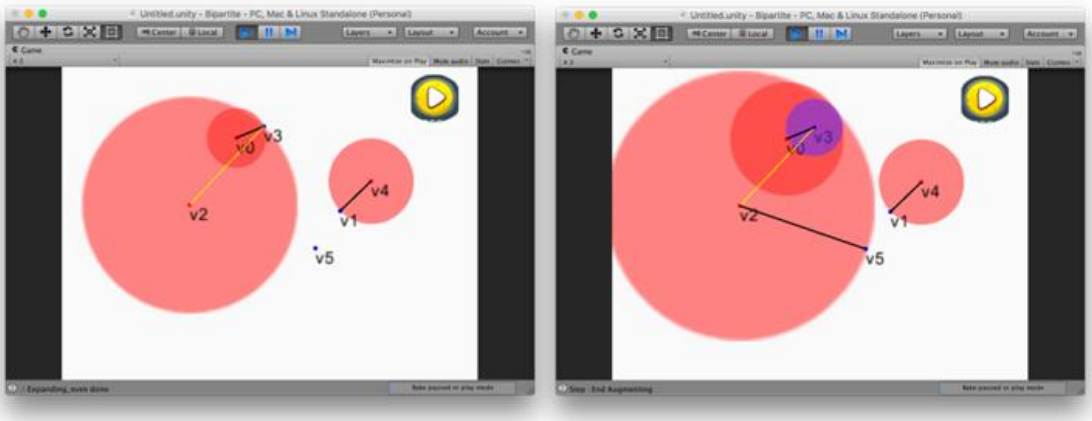

Figure 15 Bipartite Matching 구현물

Bipartite matching 문제의 예이다. 여기서 검정색 선은 실제로 이루어진 매칭 간선이며, 노란색 선은 tight하기는 하지만 매칭은 되지 못한 두 정점을 잇는 형태이다. 매 step마다 현재 tree에 속해 있는 정점들의 원이 동시에 늘어나면서 새로운 매칭을 찾게 된다. 물론 항상 파란 원은 다른 빨간 원에 내접한 상태를 유지한다.

Schedule

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
관련 연구 조사	→	→	→	→										
연구제안서 발표	→	→	→	→										
목표 알고리즘 선정 및 연구				→	→	→	→	→	→	→	→	→		
중간 발표 1						→	→	→						
알고리즘 시각화 착수						→	→	→	→	→	→	→		
중간 발표 2										→	→	→		
결과물 테스트 및 개선								→	→	→	→	→	→	
최종 발표												→	→	
시연 및 최종 보고서 작성													→	→

References

- M. Jünger and W. Pulleyblank, Geometric Duality and Combinatorial Optimization, 1992

- Kleinberg and Tardos. Cornel University, Fall 2010, Algorithms Lecture Notes: Primal-

dual min-cost bipartite matching

(https://www.google.co.kr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjWuPjc5OHQAhXCjZQKHVF9CMgQFggaMAA&url=http%3A%2F%2Fwww.cs.cornell.edu%2Fcourses%2Fcs6820%2F2010fa%2Fhandouts%2Fmatchings.pdf&usg=AFQjCNHu8e5eJwQsz1j8YXv4tQO7C8riCw&sig2=HkH0v4Od53JrhzJUJbeOmg)

- Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi and Vijay V. Vazirani. Greedy Facility Location Algorithms Analyzed Using Dual Fitting with Factor-Revealing LP

- Figure 1: https://en.wikipedia.org/wiki/Quicksort

- Figure 2: http://www.onmyphd.com/?p=duality.theory

- Figure 3: Seongbong Tang. Yonsei University, Fall 2015, Algorithm Analysis Lecture Notes: Linear Programming

- Unity reference - https://docs.unity3d.com/ScriptReference/