mapf-gp

nguyenngockhanh.pbc

January 2021

Contents

Chapter 1

Introduction

Our central problem is defined as follow:

Problem 1 (Problem 1). *:*

Given a multi directed graph $G = (V, E)$ and a set of points of interest $V_P \subseteq V$.

Find K closed walks that cover V_P.

Objective: *Minimize the sum of walk length over all walks.*

Constraint 1: *Maximum walk length does not exceed L*

Constraint 2: *No overlap between any pair of walks.*

Where a walk is defined as a sequence of edges (not necessary distinct). A closed walk is defined as a walk such that the starting node and the ending node are identical.

In order to solve the problem **??**, we introduce an equivalent problem as follow:

Problem 2 (Problem 2). *:*

Given a directed graph $G_P = (V_P, E_P)$ with triangle equality.

Find K cycles that cover V_P.

Objective: *Minimize the sum of cycle length over all cycles.*

Constraint 1: *Maximum cycle length does not exceed L*

Where a cycle is defined as a closed walk that all nodes are distinct.

Theorem 1 (Theorem 1). *:*

Problem **??** ** (without constraint 2) and problem* **??** *can be reduced from each other.*

Problem **??*** and problem **??** are equivalent. That means we can use the results of problem **??** to solve problem **??*** [1]. The reduction is as follow:

[1] Informal proof in Appendix

Reduction 1 (problem **??*** → problem **??**). *:*

Given a multi directed graph $G = (V, E)$ and a set of points of interest $V_P \subseteq V$. Let $G_P = (V_P, E_P)$ be a directed graph such that each edge $(v_i, v_j) \in E_P$ is the shortest path in G.

Using reduction **??**, we can construct an instance of problem **??** that is equivalent to problem **??***. After solving G_P, we map each of the edges in the solution of G_P by its corresponding shortest path in G.

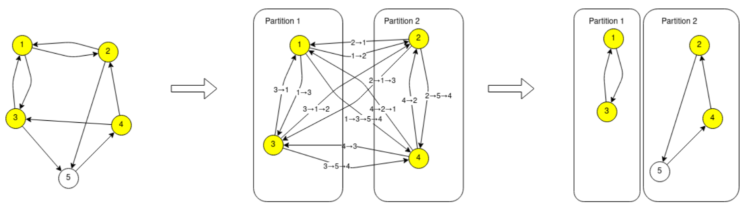

Figure 1.1: Reduction

Chapter 2

Pipeline

We consider a variant of problem **??** as follow:

Problem 3 (Problem 3)**.** *:*
Given a directed graph $G_P = (V_P, E_P)$ with triangle equality.
Find K cycles that cover V_P.
Objective: *Minimize the maximum of cycle length over all cycles.*

The algorithm pipeline is as follow:

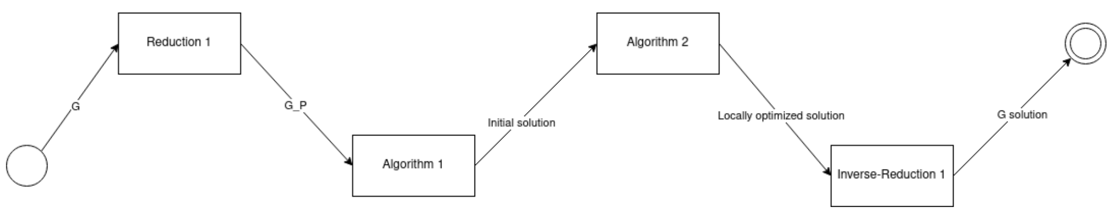

Figure 2.1: Pipeline

Reduction 1 converts a graph G and a set of POIs V_P to graph G_P. After that, Algorithm 1 and Algorithm 2 find a approximate-solution for problem **??**. Finally, Inverse-Reduction 1 converts the approximate-solution back to an approximate-solution for problem 1.

Chapter 3

Preliminaries

3.1 Lower bound

A lower bound on problem **??** can be derived using a minimum assignment problem.

Problem 4 (Minimum assignment)**.** *:*

Given a set of m agents and n tasks. The cost matrix $M \in \mathbb{R}_+^{m \times n}$ is given such that each entry M_{ij} is the non-negative cost of the assignment agent i to task j. Each agent is assigned to at most one task and each task is assigned to at most one agent.

Find $min\{m, n\}$ assignments A ($A \subset [m] \times [n]$, $|A| = min\{m, n\}$) that minimize $\sum_{(i,j) \in A} M_{ij}$

Let $A \in \mathbb{R}_+^{|V_P| \times |V_P|}$ be the G_P adjacency matrix where each entry A_{ij} is the non-negative edge length from node i to node j. Define $A_{ii} = 0$. The lower bound on the total cost of k cycles can be obtained from the solution of problem **??** by treating A as the cost matrix.

3.2 Optimization Approximation

Consider the program of minimizing a function $f : X \to \mathbb{R}$. In many scenarios, it is hard to find an optimal or it is hard to compute the value of $f(x)$. Isaac Vandermeulen, Roderich Groß, Andreas Kolling **?** has introduced a method that find a function $f_1 : X \to \mathbb{R}$ such that $f(x) = c(f_1(x)) + v$ where c is a monotonically increasing function and v is a random variable.

Let some bounds on v as follows: $\alpha \in (0, 0.5)$ and $b_\alpha^-, b_\alpha^+ \in \mathbb{R}_+$

$$\mathbb{P}[-b_\alpha^- \leq v] = \mathbb{P}[v \leq +b_\alpha^+] = 1 - \alpha$$

Let x^* and x_1^* be the optimal values for f and f_1.

$$(A) : f(x_1^*) \leq f(x^*) + b_\alpha^- + b_\alpha^+$$

Theorem 2 (Approximation)**.** *:*

$$\mathbb{P}[A] \geq (1-\alpha)^2$$

The theorem states that if one can find f_1 with small variance on v, minimizing f_1 provides a good solution on f with high probability.

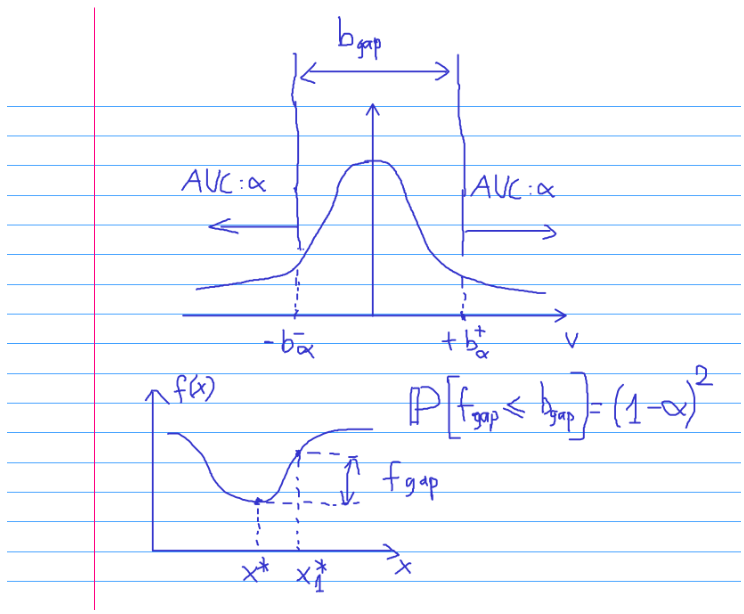

Figure 3.1: Optimization

In **?**, the authors defined average cycle length as follow:

Definition 1 (Average cycle length)**.**

$$C_a^{(k)} = \frac{\sum_{i \in V_k} \sum_{j \in V_k} A_{ij}}{|V_k| - 1}$$

Where V_k is the set of nodes in the partition k.

By choosing f_1 as maximum average cycle length, the authors approximated the solution of problem **??** by minimizing:

$$O_0 = max\{C_a^{(k)}\}_{k=1}^{K}$$

However, maximum average cycle length does not contain any information about the other $K-1$ smaller cycles. We have experimented with different f_1 objective functions.

From the pattern on each f_1 function to the f, we can predict that test4 and test7 tend to provide a smaller maximum cycle length and test1 and test 8 tend to provide a smaller total cycle length.

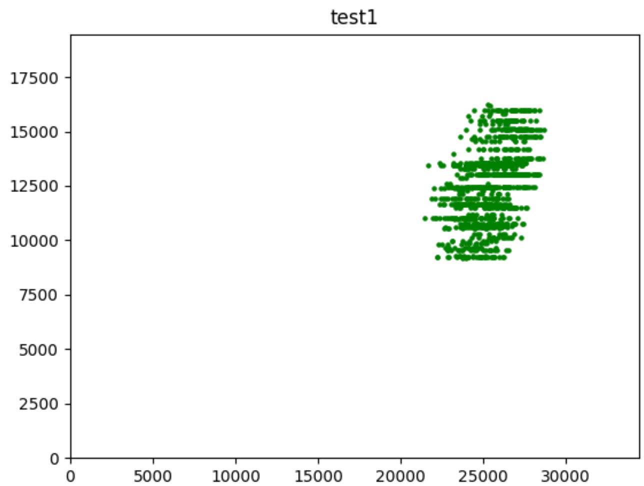

Figure 3.2: maximum tsp cycle length over test1 objective of all 5-partitions in a 15 nodes network

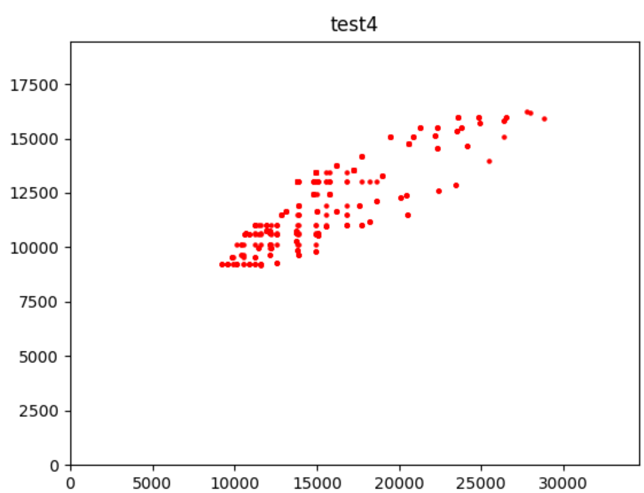

Figure 3.3: maximum tsp cycle length over test4 objective of all 5-partitions in a 15 nodes network

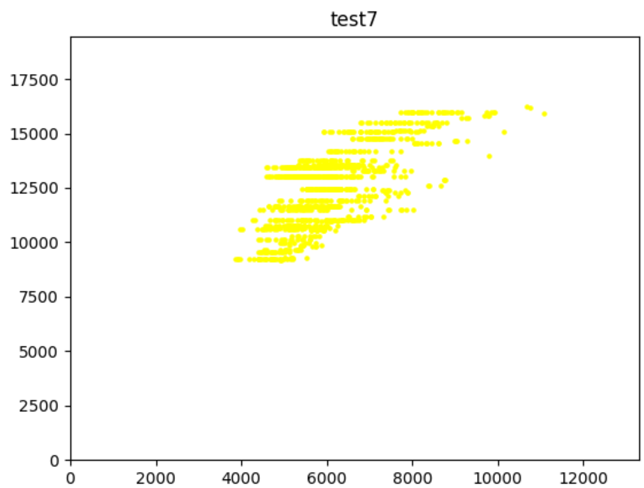

Figure 3.4: maximum tsp cycle length over test7 objective of all 5-partitions in a 15 nodes network

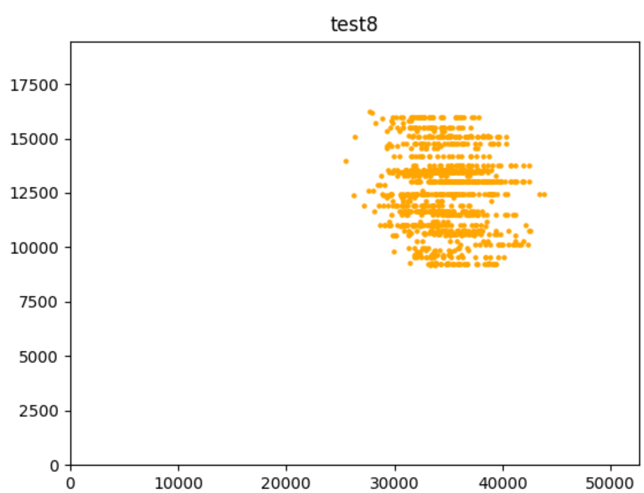

Figure 3.5: maximum tsp cycle length over test8 objective of all 5-partitions in a 15 nodes network

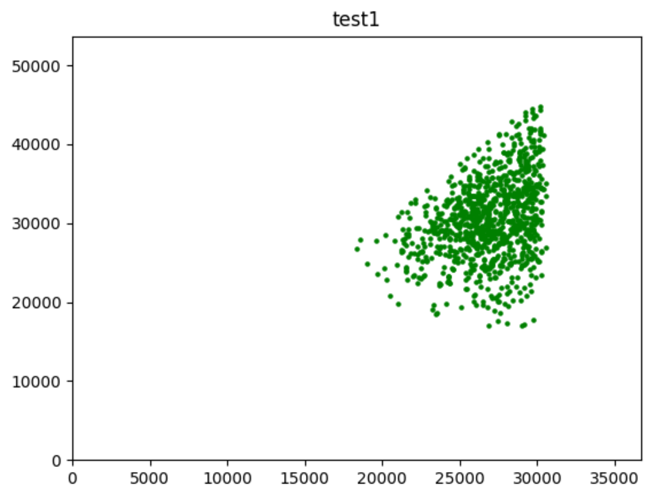

Figure 3.6: sum tsp cycle length over test1 objective of all 5-partitions in a 15 nodes network

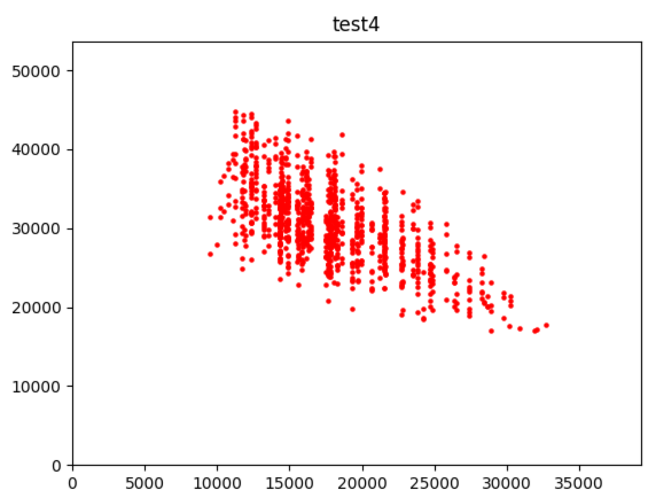

Figure 3.7: sum tsp cycle length over test4 objective of all 5-partitions in a 15 nodes network

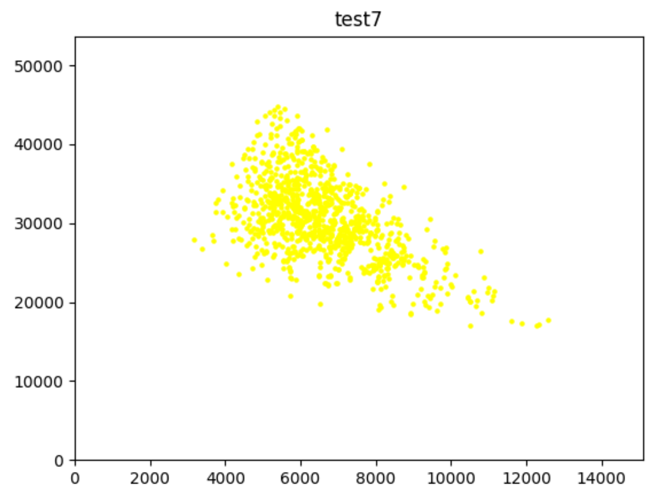

Figure 3.8: sum tsp cycle length over test7 objective of all 5-partitions in a 15 nodes network

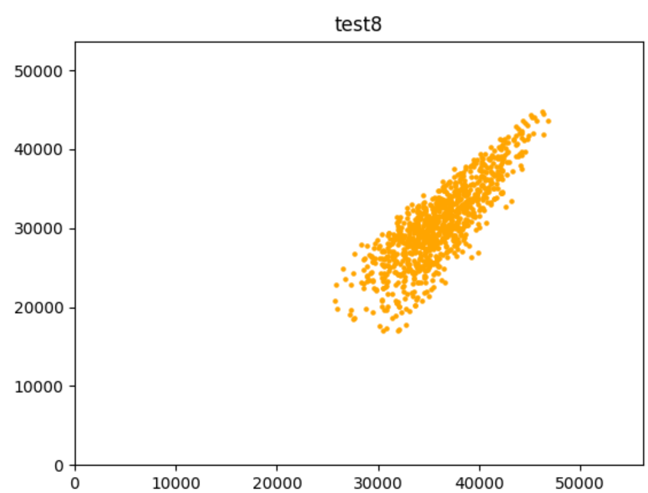

Figure 3.9: sum tsp cycle length over test8 objective of all 5-partitions in a 15 nodes network

Chapter 4

Methods

4.1 Algorithm 1

4.1.1 test1

Inspired by derivation of Ratio cut **?**, we introduced method test1.

Let A be symmetric, let D be the degree matrix of A, $L = D - A$ be the laplacian matrix. Let $x^{(1)}$, $x^{(2)}$, ..., $x^{(K)}$ be the indicator vector for each partition such that:

$$x_i^{(k)} = \{ \begin{array}{ll} 1 & \text{if } i \in V_k \\ 0 & \text{otherwise} \end{array}$$

Since K partitions are disjoint, we always have a set of K orthogonal vectors.

$$x^{(k_1)T} x^{(k_2)} = 0 \quad \forall k_1 \neq k_2$$

Ratio cut minimizes the sum of all cuts divided by its corresponding partition volume.

$$\sum_{k=1}^{K} \frac{x^{(k)T} L x^{(k)}}{x^{(k)T} x^{(k)}} = \sum_{k=1}^{K} \frac{cut(V_k, V \setminus V_k)}{|V_k|}$$

subject to $x^{(k)} \in \{0,1\}^{|V|}$ and $x^{(k_1)T} x^{(k_2)} = 0 \; \forall k_1 \neq k_2$

Furthermore, Ratio cut extends the domain the indicator vectors to real number.

$$\sum_{k=1}^{K} \frac{x^{(k)T} L x^{(k)}}{x^{(k)T} x^{(k)}}$$

subject to $x^{(k)} \in \mathbb{R}^{|V|}$ and $x^{(k_1)T} x^{(k_2)} = 0 \; \forall k_1 \neq k_2$

The problem of minimizing Rayleigh quotients with orthogonal constraints yields K-smallest eigen vectors.

In test1, we replaced laplacian matrix L by adjacency matrix A. The objective of the formulation is

$$O_1 = \sum_{k=1}^{K} \frac{x^{(k)T} A x^{(k)}}{x^{(k)T} x^{(k)}} = \sum_{k=1}^{K} \frac{\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} x_i^{(k)} x_j^{(k)} A_{ij}}{\sum_{i=1}^{|V|} x_i^{(k)2}} = \sum_{k=1}^{K} \frac{\sum_{i \in V_k} \sum_{j \in V_k} A_{ij}}{|V_k|} = \sum_{k=1}^{K} (1 - \frac{1}{|V_k|}) C_a^{(k)}$$

Which is approximately equal to the sum of average cycle length.

4.1.2 test7

test7 uses the same concept of indicator vector from test1. We modified the objective function as follow:

$$O_7 = \sum_{k=1}^{K} \frac{x^{(k)T} (A - \alpha D) x^{(k)}}{x^{(k)T} x^{(k)} + \beta |V|} = \sum_{k=1}^{K} \frac{\sum_{i \in V_k} \sum_{j \in V_k} A_{ij}}{|V_k| + \beta |V|} - \alpha \frac{\sum_{i \in V_k} D_{ii}}{|V_k| + \beta |V|}$$

Where α and β are two non-negative hyper-parameters.

Constant β makes the partitions more balanced. Let $\alpha = 0$, consider a graph that all edges have unit length. A positive constant β makes the unique minimum partition to be the balanced one. [1]

Constant α is intended to encourage large degree nodes to join the small partitions.

If $\alpha = 0$ and $\beta = 0$, we obtain the objective function of test1. If $\alpha \in \{0, 1\}$ and $\beta \to +\infty$, we obtain the objective function of MAX-CUT problem.

In the experiment, we have chosen $\alpha = 0$ and $\beta = 1$.

Instead of the orthogonal constraint, we imposed a pair of constraints for the indicator vectors.

$$x^{(k)} \succeq 0 \text{ and } \sum_{k=1}^{K} x^{(k)} = 1_{|V|}$$

The pair of constraints is a soft indicator of each node belonging to a partition. In the experiment, we changed the second constraint to be inequality.

$$x^{(k)} \succeq 0 \text{ and } \sum_{k=1}^{K} x^{(k)} \succeq 1_{|V|}$$

Finally, the partition for each node is taken as:

$$k_i = argmax_k \{x_i^{(k)}\}_{k=1}^{K}$$

[1] Detailed analysis in Appendix

4.1.3 Local Search

The authors **?** introduced two local search operations: $transfer$ and $swap$. $transfer$ is the operation that moves a node from one partition to another partition. $swap$ is the operation that exchanges two nodes from a pair of partitions.

Algorithm 1 Local Search

Input::
Initial K partitions
Strategy $\in$ { first, best }
Objective function
Output::
Final K partitions
Procedure::

```
While true:
    var candidate_set = []
    For operation in all possible operations:
        Calculate the gain of the operation
        If the gain is positive:
            Push the operation and gain to candidate set
            If Strategy is first:
                break
    If candidate_set is empty:
        break
    Pick the best candidate, update the current K partitions
return current K partitions
```

The algorithm **??** is capable to use with the two proposed operations to approximately find our solution.

test4

Strategy: first

$$O_4 = O_0 = max\{C_a^{(k)}\}_{k=1}^{K}$$

test8

Strategy: first

$$O_8 = \sum_{k=1}^{K} C_a^{(k)}$$

4.2 Algorithm 2

In algorithm 2, we firstly convert K partitions using standard TSP algorithm for each partition. In this stage, we used a different type of operation for cycles.

$Transfer2$ is the operation of transferring a node from the longest cycle to another edge of a shorter cycle.

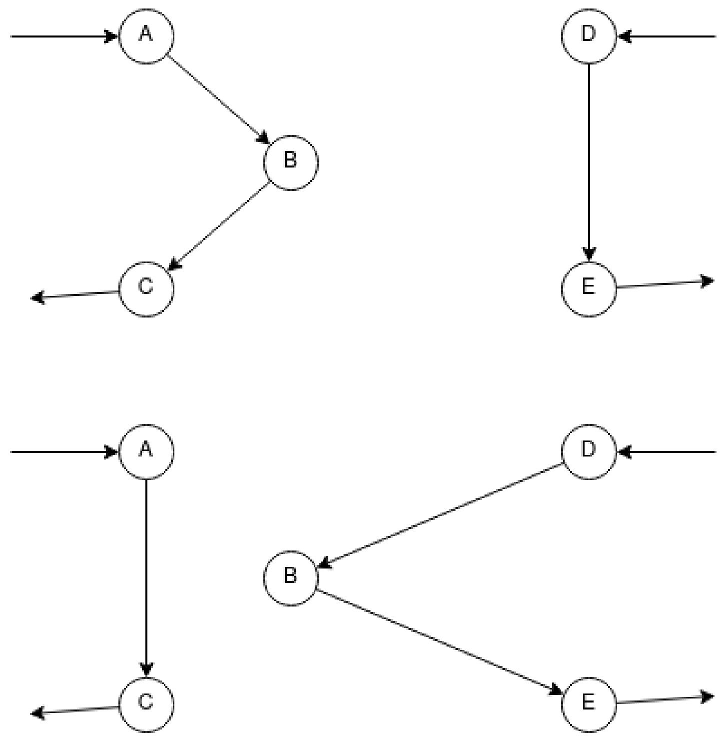

Figure 4.1: Transfer of B to $D \to E$

Algorithm 2 Local Search 2

Input::
Initial K cycles
Strategy $\in$ { first, best }
Output::
Final K cycles
Procedure::

```
While true:
    var candidate_set = []
    For operation in all possible operations:
        Calculate the gain of the operation
        If operation reduced maximum cycle length and did not induce any intersection:
            Push the operation and gain to candidate set
            If Strategy is first:
                break
    If candidate_set is empty:
        break
    Pick the candidate with smallest sum of cycle length, update the current K partitions
return current K partitions
```

Chapter 5

Appendix

5.1 Informal proof of theorem ??

5.1.1 problem ?? → problem ??*

$V_P \to V$: nodes of problem **??** become nodes of problem **??***
$V_P \to V_P$: nodes of problem **??** become POIs of problem **??***
$E_P \to E$: edges of problem **??** become edges of problem **??***
$L \to L$: remains L

If S_1 is the solution problem **??***, S_1 is also a valid assignment for problem **??** ($|S_1| \geq |S_2|$).

The claim is correct due to this procedure: let any B appears more than once in S_1, $A \to B \to C$ becomes $A \to C$.

If S_2 is the solution of problem **??**, S_2 is also a valid assignment for problem **??*** ($|S_2| \geq |S_1|$).

Hence, problem **??*** is at least as hard as problem **??**

5.1.2 problem ??* → problem ??

$V_P \to V_P$: POIs of problem **??*** become nodes of problem **??**
*: shortest paths of problem **??*** become edges of problem **??**
$L \to L$: remains L

If S_2 is the solution of problem **??**, S_2 is also a valid assignment for problem **??*** ($|S_2| \geq |S_1|$)

If S_1 is the solution of problem **??***, $S_1$is also a valid assignment for problem **??**

The claim is correct by:

If there is no node appears more than once, trivial.

Otherwise, let any B appears more than once in S_1, $|A \to B \to C| \geq |A \to C|$. If the inequality occurs, replacing $|A \to B \to C|$ by $|A \to C|$ yields a shorter solution that conflicts with S_1 is the solution of problem **??***. The equality occurs, we replace $|A \to B \to C|$ by $|A \to C|$ until there is no more node appear more than once.

Hence, problem **??** is at least as hard as problem **??***

5.2 Informal proof of theorem ??

Consider the program of minimizing a function $f : X \to \mathbb{R}$. In many scenarios, it is hard to find an optimal or it is even hard to compute the value of $f(x)$. Isaac Vandermeulen, Roderich Groß, Andreas Kolling **?** has introduced a method that find a function $f_1 : X \to \mathbb{R}$ such that $(1) f(x) = c(f_1(x)) + v$ where c is a monotonically increasing function and v is a random variable.

Let some bounds on v as follows: $\alpha \in (0, 0.5)$ and $b_\alpha^-, b_\alpha^+ \in \mathbb{R}_+$

$$\mathbb{P}[-b_\alpha^- \leq v] = \mathbb{P}[v \leq +b_\alpha^+] = 1 - \alpha$$

Let x^* and x_1^* be the optimal values for f and f_1.

Consider 3 random events:

$$(A) : f(x_1^*) \leq f(x^*) + b_\alpha^- + b_\alpha^+$$

$$(B) : f(x_1^*) \leq c(f_1(x_1^*)) + b_\alpha^+$$

$$(C) : f(x^*) \geq c(f_1(x^*)) - b_\alpha^-$$

Theorem 3 (Approximation)**.** *:*
$\mathbb{P}[A] \geq (1 - \alpha)^2$

We have $f(x^*) \leq f(x_1^*)$ and $f_1(x_1^*) \leq f_1(x^*)$. c is a monotonically increasing function, so $c(f_1(x_1^*)) \leq c(f_1(x^*))$.

If (B) holds,

$$f(x_1^*) \leq c(f_1(x_1^*)) + b_\alpha^+ \leq c(f_1(x^*)) + b_\alpha^+$$

If (C) also holds,

$$f(x_1^*) \leq c(f_1(x^*)) + b_\alpha^+ = (c(f_1(x^*)) - b_\alpha^-) + b_\alpha^- + b_\alpha^+$$

Hence,

$$(A) : f(x_1^*) \leq f(x^*) + b_\alpha^- + b_\alpha^+$$

(A) implies the upper-bound on how good the solution of f_1. By assuming v, x^* and x_1^* be independent, $\mathbb{P}[B \cap C] = \mathbb{P}[B] \times \mathbb{P}[C]$. Since $B \cap C \to A$, $\mathbb{P}[A] \geq \mathbb{P}[B] \times \mathbb{P}[C]$. From (1), $\mathbb{P}[B] = \mathbb{P}[C] = 1 - \alpha$. So, $\mathbb{P}[A] \geq (1 - \alpha)^2$

In conclusion, if one can the proxy f_1, a good solution for f can be obtained with high probability.

The error on this approximation depends on how good we can find a function f and a smoother c.

5.3 Analysis on constant β

Let $\alpha = 0$, $\beta > 0$, consider a graph that all edges have unit length. Objective function is

$$O_7 = \sum_{k=1}^{K} \frac{x^{(k)T}(A - \alpha D)x^{(k)}}{x^{(k)T}x^{(k)} + \beta|V|} = \sum_{k=1}^{K} \frac{\sum_{i \in V_k} \sum_{j \in V_k} A_{ij}}{|V_k| + \beta|V|} = \sum_{k=1}^{K} \frac{|V_k|(|V_k| - 1)}{|V_k| + \beta|V|}$$

Subject to the constraint

$$\sum_{k=1}^{K} |V_k| = |V| \text{ and } |V_k| > 0 \ \forall k$$

Let $x_k = \frac{|V_k|}{|V|}$ be a real variable, we have the problem of minimizing

$$f(x) = \sum_{k=1}^{K} \frac{x_k(x_k - 1)}{x_k + \beta|V|}$$

Subject to

$$c_0(x) = \sum_{k=1}^{K} x - 1 = 0 \text{ and } c_k(x) = -x_k \leq 0 \ \forall k$$

We have:

$$\frac{\partial f}{\partial x_k} = |V|(1 - \frac{\beta(\beta + 1/|V|)}{(x_k + \beta)^2})$$

$\frac{\partial f}{\partial x_k}$ has this property if $\beta > 0$:

$$(1)\text{: } \frac{\partial f}{\partial x_k} \text{ is a monotonically increasing function for all } x_k \geq 0$$

Theorem **??** deduces the unique minimum of O_7 at $|V_k| = |V|/K$

5.4 Theorem ??

Given the program:

$$\textbf{Minimize: } f(x) \textbf{ subject to: } c_0(x) = \sum_{i=1}^{n} x_i = 1 \text{ and } c_i(x) = -x_i \leq 0 \ \forall i$$

Such that $\frac{\partial f}{\partial x_i}$ has this property:

$$(1)\text{: } \frac{\partial f}{\partial x_i} \text{ is a monotonically increasing function for all } x_i \geq 0$$

$$\frac{\partial f}{\partial x_i}(x_1) < \frac{\partial f}{\partial x_i}(x_2) \ \forall\ 0 \le x_1 < x_2$$

Theorem 4 (Unique solution)**.** *Program has unique solution at* $x_i = \frac{1}{n}\ \forall i$

Lagrangian function is

$$L(x, \mu, \lambda) = f(x) + \sum_{i=1}^{n} \mu_i c_i(x) + \lambda c_0(x)$$

If $x*$ is a minimum, KKT conditions:

Stationary: $\frac{\partial L}{\partial x}(x*) = 0_n$

Primal feasibility: $c_0(x*) = 0$ and $c_i(x*) \le 0\ \forall i$

Dual feasibility: $\mu_i \ge 0\ \forall i$

Complementary slackness: $\mu_i c_i(x*) = 0\ \forall i$

We have:

$$\frac{\partial L}{\partial x_k} = \frac{\partial f}{\partial x_k} - \mu_k + \lambda$$

Consider 2 cases:

Case 1: all $x_i > 0$, due to complementary slackness, all $\mu_i = 0$. So that, all $\frac{\partial f}{\partial x_i}$ must be equal. (1) deduces that the unique solution satisfying KKT conditions is $x_i = \frac{1}{n}$

Case 2: some $x_i = 0$, let $x_{i1} = 0$

$$\frac{\partial f}{\partial x_i}(x_{i1}) - \mu_{i1} + \lambda = 0$$

$$\lambda = \mu_{i1} - \frac{\partial f}{\partial x_i}(0)$$

Since dual feasibility, $\mu_{i1} \ge 0$, So

$$\lambda \ge -\frac{\partial f}{\partial x_i}(0)$$

There is at least one $x_i > 0$, let $x_{i2} > 0$, due to complementary slackness, $\mu_{i2} = 0$, So

$$\frac{\partial f}{\partial x_i}(x_{i2}) + \lambda = 0$$

$$\frac{\partial f}{\partial x_i}(x_{i2}) = -\lambda \le \frac{\partial f}{\partial x_i}(0)$$

(1) deduces contradiction.