mapf-gp-summary

nguyenngockhanh.pbc

March 2021

Contents

Chapter 1

Problem Decomposition

1.1 Problem Statement

In the map of a specific location in Singapore, there are multiple points of interest. Our goal is to navigate multiple robots to go around these points for the patrolling task where several objective values and constraints need to be satisfied. The central objective is to minimize the total cost of all tours and balance between those tours. Additionally, some conflicts must not occur, for example, two robots must not go from two opposite directions in the same road since it would cause collision in the real world.

We hence model the problem as a constrained optimization problem.

Problem 1 (Full Problem)**.** *Given a directed graph $G = (V, E)$ with corresponding cost function on each edge $c : E \to \mathbf{R}_+$ and a set of points of interest (POIs) $P \subseteq V$. Find K tours that visit V_P and minimize the maximum tour cost satisfy a set of predefined constraints.*

Let's begin tackle this problem by dropping all constraints and reduce the problem into a simpler version.

Problem 2 (Partial Problem)**.** *Given a directed graph $G_P = (V_P, E_P)$ with corresponding cost function on each edge $c_P : E_P \to \mathbf{R}_+$. Find K tours that visit V_P and minimize the maximum tour cost.*

Each edge in G_P from $p_1 \to p_2$ for $p_1, p_2 \in V_P$ is corresponding to the shortest path from p_1 to p_2. It is provable that if we drop all constraints, problem **??** and **??** are equivalent.

At the first glance, the problem **??** appears to be a graph partitioning problem where we need to partition the node set into multiple subset then allocate each agent to go around a particular subset and find the shortest tour in each subset. We have tried out many different graph partitioning techniques. In the experiment for randomly generated 2D points, the general techniques seemed to work well. However, for the real work case (Marina Bay and Botanic

Garden), the general techniques failed to partition the POIs set into multiple balanced tours. It suggests that the real networks exhibit a different geometry as compare to a random setup. Due to different in the objective function, we need to design an algorithm that specifically optimizes our objective.

1.2 Problem Decomposition

We decompose the problem into four steps.(**1. Partial Problem**) Reduce the instance of problem **??** without any constraints to an instance of problem **??**.(**2. Partial Solution**) Find an partial solution to problem **??**. (**3. Locally Optimized Partial Solution**) Locally optimize the partial solution. (**4. Constrained Solution**) Find the locally optimal solution that satisfies all constraints from the partial solution.

Chapter 2

Methods

2.1 Partial Problem

As discussed earlier, one can convert the problem **??** into a simpler version (problem **??**) by creating a new graph with nodes are the POIs and edges are the set of shortest paths between all pairs of POIs.

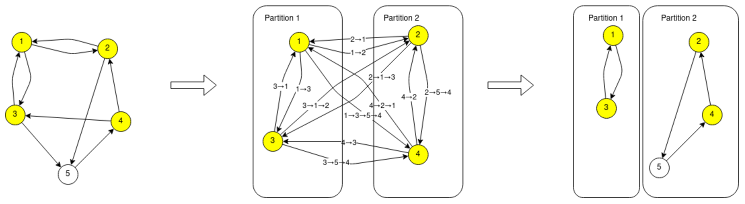

Figure 2.1: Reduction

2.2 Partial Solution

The problem **??** is different from other graph partitioning problems at the objective function. Unlike other problems, our objective function is not even polynomial-time computable (Travelling Salesman Problem). Isaac Vandermeulen, Roderich Groß, Andreas Kolling **?** have introduced a clever technique to obtain an approximation solution to the problem. In this section, we introduce a method to approximate the solution of problem **??** that fills some gaps in the original method.

2.2.1 Preliminaries

Consider the problem of minimizing a function $f : D \to \mathbb{R}$

In many scenarios, it is hard to find an optimal, or it is even also hard to compute the value of f. The method below was inspired from the work of Isaac Vandermeulen, Roderich Groß, Andreas Kolling **?**.

For a domain $X \subseteq D$ of the minimization problem, let $f_1 : X \to \mathbb{R}$ be the proxy function such that

$$(1) : f(x) = c(f_1(x)) + v(x)\ \forall x \in x$$

Where c is a monotonically increasing function and $v : X \to \mathbb{R}$ is function on X.

Let x^* and x_1^* be the optimal values for f and f_1 in the domain $X \subseteq D$. Let $t_{\max} = \max_{x \in X} v(x)$ and $t_{\min} = \min_{x \in X} v(x)$ be the maximum value and minimum value of v over the domain X.

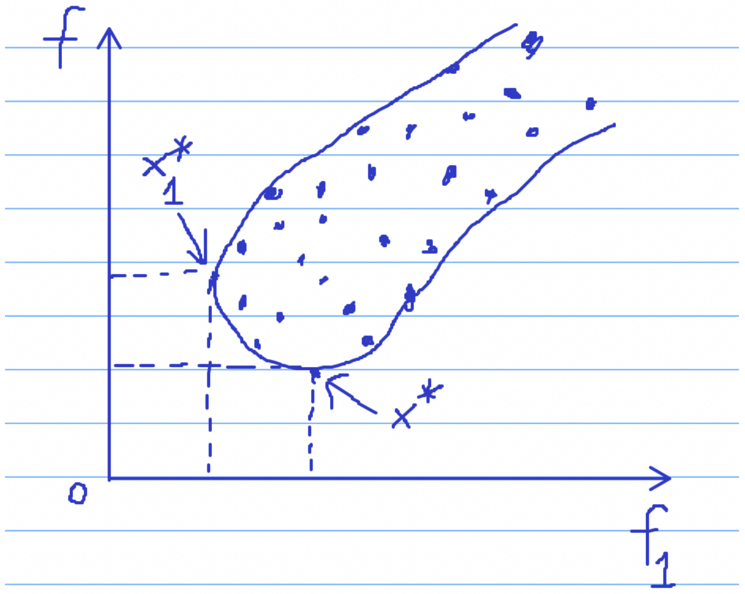

Figure 2.2: f_1 approximation

Theorem 1 (f_1 approximation)**.**

$$f(x_1^*) \leq f(x^*) + (t_{\max} - t_{\min})$$

The theorem **??** provides us a guarantee on how good the solution of f_1 in minimizing f. By finding an appropriate proxy function f_1, we can find an approximation on f.

2.2.2 Proxy Function

In **?**, in order to minimize the maximum tour cost, the authors introduced a proxy function f_1 to be the maximum of average tour cost of the K tours and a local search algorithm to find a local optimal. However, the function does only capture the maximum-length tour over the K tours that does not allow the local search to enhance to total tour cost and potentially traps the algorithm in a bad region in the search space. We introduce three proxy function as follow:

Given K partitions of the graph G_P, each partition is a disjoint subset of nodes in G_P (a subset of POIs). Denote $C_m^{(k)}$ to be the minimum tour cost of the k-th partition (TSP tour cost). Our objective function is to find a partitioning of K partition such that the objective is minimized:

$$f(.) = \max\{C_m^{(k)}\}_{k=1}^{K} = ||C_m||_\infty$$

Maximum of Average Tour Cost and Sum of Average Tour Cost

In **?**, denote $C_a^{(k)}$ to be the average tour cost if the k-th partition where the average tour cost in a graph is defined as expected value of a tour when the node ordering is chosen at uniformly random order all possible ordering. Let $V_P^{(k)}$ be the set of POIs in the k-th partition. $C_a^{(k)}$ can be calculated by:

$$C_a^{(k)} = \frac{1}{|V_P^{(k)}| - 1} \sum_{v_i \in V_P^{(k)}} \sum_{v_j \in V_P^{(k)} \setminus \{v_i\}} c_P((v_i, v_j))$$

Let $A_P^{(k)}$ be the adjacency of the sub-graph induced by $V_P^{(k)}$ with zeros on diagonal, then $C_a^{(k)}$ can also be calculated as the sum of all entries divided by $|V_P^{(k)}| - 1$.

$$C_a^{(k)} = \frac{\mathbb{1}^T A_P^{(k)} \mathbb{1}}{|V_P^{(k)}| - 1}$$

The author used maximum of average tour cost:

$$f_1(.) = O_0(.) = \max\{C_a^{(k)}\}_{k=1}^{K} = ||C_a||_\infty$$

Based on the same concept of average tour cost, we introduce the sum of average tour cost:

$$f_1(.) = O_8(.) = \sum_{k=1}^{K} C_a^{(k)} = ||C_a||_1$$

Another trivial extension is to interpolate $O_0(.)$ and $O_8(.)$:

$$f_1(.) = O_9(.) = \alpha||C_a||_\infty + \beta||C_a||_1$$

for $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$ Or,

$$f_1(.) = O_{10}(.) = \frac{1}{K}\sum_{k=1}^{K} \log C_a^{(k)}$$

Given a set of partitioning where $O_8(.)$ have the same objective value, $O_{10}(.)$ will find the most balanced partitioning [1]. Given a set of partitions where $O_0(.)$ have the same objective value, $O_{10}(.)$ will find the least cost partitioning [2]. [3]

By log concavity, $O_{10}(.)$ lies somewhere between $O_0(.)$ and $O_8(.)$

$$\frac{1}{K}\log \max\{C_a^{(k)}\}_{k=1}^{K} \le \frac{1}{K}\sum_{k=1}^{K} \log C_a^{(k)} \le -\log K + \log \sum_{k=1}^{K} C_a^{(k)}$$

Furthermore, this function encourage partitions being more balanced [4]

Other Handcrafted Tour Cost

Let A be the adjacency matrix of G_P defined by:

$$A_{ij} = c_P((v_i, v_j))$$

where $c_P((v_i, v_i)) = 0\ \forall v_i \in V_P$ (diagonal entries are all zero).

Let $x^{(k)} \in \{0,1\}^{|V_P|}$ be an indicator vector of k-th partition.

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

Since, K partitions are disjoint, we have an orthogonal constraint on $x^{(k)}$ such that $\forall k_1 \neq k_2$, $x^{(k_1)T}x^{(k_2)} = 0$

For the k-partition, the quadratic form $x^{(k)T}Ax^{(k)}$ equals the sum of all entries in the sub-graph adjacency matrix $A_P^{(k)}$ induced by $V_P^{(k)}$. We have:

$$C_a^{(k)} = \frac{x^{(k)T}Ax^{(k)}}{|V_P^{(k)}| - 1} = \frac{x^{(k)T}Ax^{(k)}}{x^{(k)T}x^{(k)} - 1}$$

We use the proxy function defined by:

$$f_1(.) = O_1(.) = \sum_{k=1}^{K} \frac{x^{(k)T}Ax^{(k)}}{x^{(k)T}x^{(k)}}$$

At this stage, we relax our 0-1 constraint to the indicator vector. Minimizing $O_1(.)$ with respect to the orthogonal constraint yields the K smallest eigenvalues. The minimal is at K corresponding eigenvectors.

[1] $O_{10}(.)$ converges at the middle of the simplex

[2] zero all other partitions except the maximum one

[3] hand-wavy argument

[4] A similar proof can be found in Appendix

The objective $O_1(.)$ is very close to the sum of average tour cost function. However, we are finding the partition such that the maximum tour cost should be minimized. We introduce a more complex proxy function as follow:

$$f_1(.) = O_7(.) = \sum_{k=1}^{K} \frac{x^{(k)T}(A - \alpha D)x^{(k)}}{x^{(k)T}x^{(k)} + \beta|V_P|} = \sum_{k=1}^{K} \frac{\sum_{i \in V_k} \sum_{j \in V_k} A_{ij}}{|V_P^{(k)}| + \beta|V_P|} - \alpha \frac{\sum_{i \in V_k} D_{ii}}{|V_P^{(k)}| + \beta|V_P|}$$

where α and β are two careful chosen non-negative parameters and D is the degree matrix of A

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. [5]

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

If $\alpha = 0$ and $\beta = 0$, we obtain $O_8(.)$. 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. This can be thought as representing each entry by the probability of the corresponding POIs belong to

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

The pair of constraints makes $x^{(k)}$ as a soft indicator. In the experiment, we changed the second constraint to be inequality.

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

Finally, the partition for each POI is taken as:

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

In the experiment, most of the time, entries of $x^{(k)}$ always converge to either 0 or 1.

In the experiment, minimizing $O_0(.)$ and $O_7(.)$ tends to reduce the maximum tour cost while minimizing $O_1(.)$ and $O_8(.)$ tends to reduce the sum of tour cost.

2.2.3 Optimizing Proxy Function

For $O_1(.)$ and $O_8(.)$, we use local search to obtain the local optimal partition. From the existing work, we use two operations defining neighbour in the search

[5]Detailed analysis in Appendix

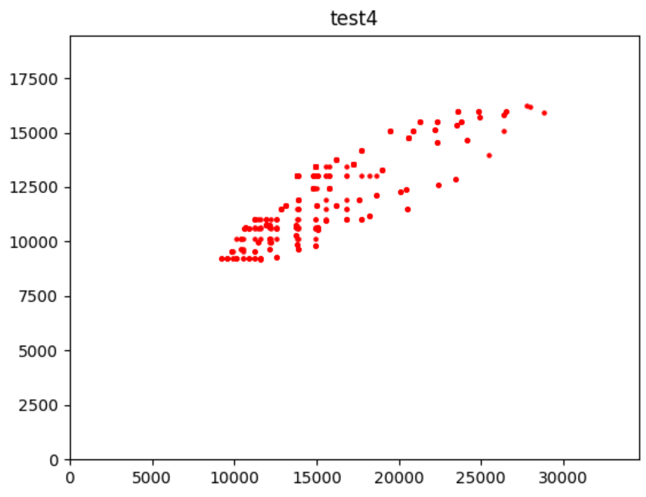

Figure 2.3: maximum of tour cost over $O_0(.)$ of all 5-partitions in a 15 nodes network

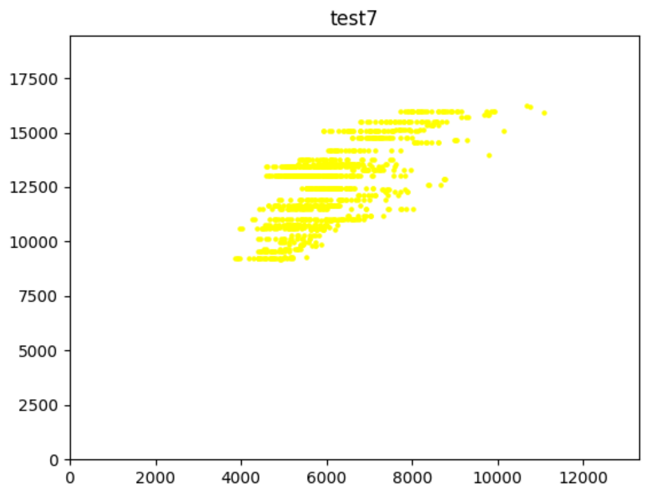

Figure 2.4: maximum of tour cost over $O_7(.)$ of all 5-partitions in a 15 nodes network

space: (**1. Transfer**) A POI is transferred from one partition to another partition. (**2. Swap**) A pair of POIs from two different partitions are swapped.

On the other hand, $O_1(.)$ can be solved using standard eigen-solver and $O_7(.)$

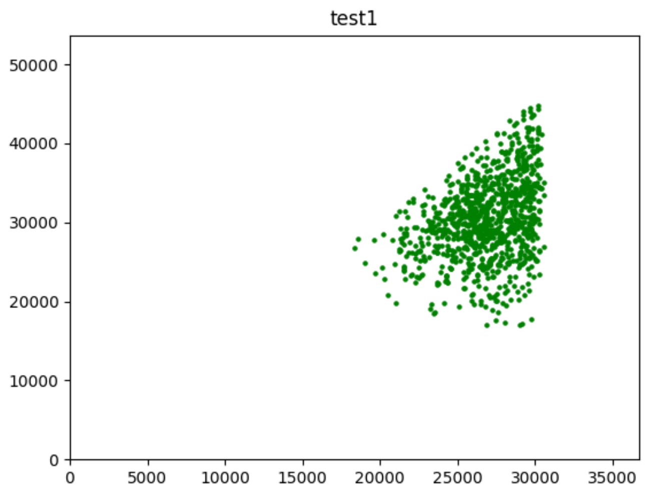

Figure 2.5: sum of tour cost over $O_1(.)$ of all 5-partitions in a 15 nodes network

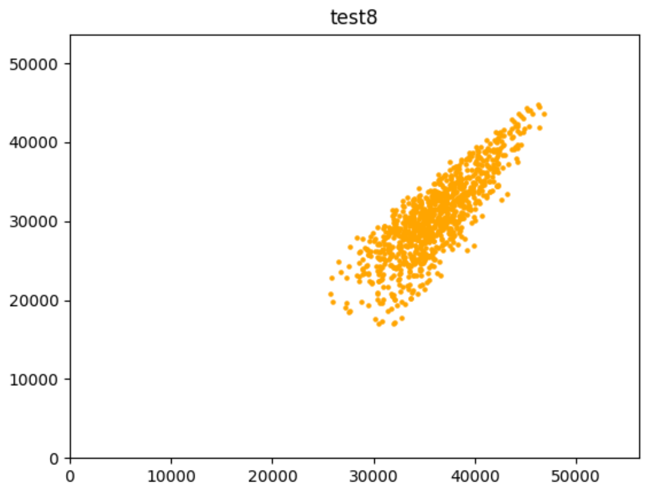

Figure 2.6: sum of tour cost over $O_8(.)$ of all 5-partitions in a 15 nodes network

can be solved using standard constrained optimization techniques.

2.3 Locally Optimized Partial Solution

Once we have a partial solution, we convert K partitions into K tours using standard TSP algorithm for each partition. In this stage, we define a new type of operation for our local search. **Transfer2** is the operation of transferring a node from the longest tour to another edge of a shorter tour. We perform the custom local search algorithm below to optimize both objectives maximum of tour cost and sum of tour cost.

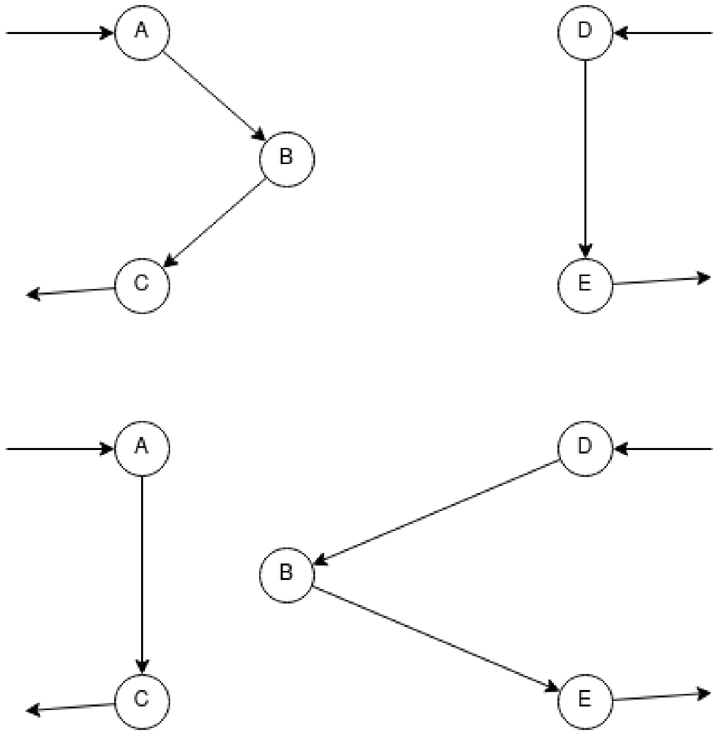

Figure 2.7: Transfer of B to $D \to E$

Algorithm 1 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 of tour cost:
            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 tour cost, update the current $K$ partitions
return current $K$ partitions
```

Performing this algorithm with strategy **best** guarantees to find a locally best partition. Equivalently, if a partial solution is returned by this algorithm, among its neighbourhood defined by the operation **Transfer2**, there is no other partial solution has a better maximum of tour cost and there is no other partial solution has a better sum of tour cost without scarifying the maximum of tour cost.

2.4 Constrained Solution

Our last step is to resolve all conflicts occurred in the partial solution. Remind that, the partial solution is just an order of POIs in each partition, it does not include any of actual paths.

2.4.1 Problem Statement

Problem 3 (Conflict Avoidance Problem)**.** *Given a set of POIs and a bijective mapping $b : V_P \to V_P$. Find a set of paths that does not yield any conflict.*

The each bijective mapping represents a valid partitioning on the graph

2.4.2 Naive Solution

A naive solution to the conflict avoidance problem of a pair of paths is to fix one path and find the shortest no-conflict path of the other. The strategy can

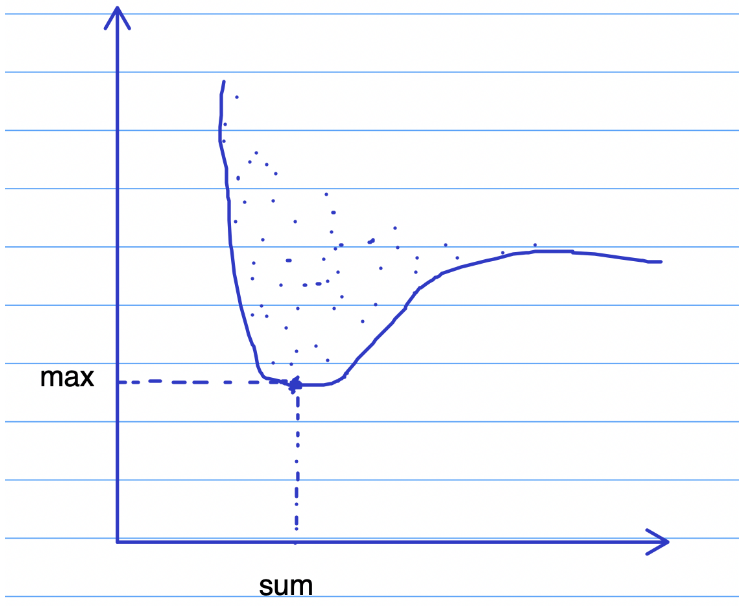

Figure 2.8: Multi-objective analogy

be used to solved the simple case as in figure **??**, we fix the blue path $1 \to 3$ and find a shortest path from $5 \to 6$ without inducing any conflict.

However, there is a long range effect that makes the problem harder. Consider the hard case in figure **??**, we fix the path from $1 \to 3$ and again find the shortest path from $5 \to 6$ without inducing any conflict. But this time, the only path causes the path $8 \to 1$ to be rescheduled.

In another case in figure **??**, we cannot pick any of the path from $1 \to 3$ or from $5 \to 6$ to be fixed while the solution still exists.

2.4.3 Conflict Based Search

Given a set of resources R and a set of k agents A. Each agent associates with a subset of resources by the cost function $c : A \times P(R) \to \mathbb{R}$. Our goal is to find such k disjoint subsets of R associate with k agents that minimize the total cost.

In conflict-based search, we define a conflict tree with each node has 3 properties: **constraint**, **assignment** and **cost**. **constraint** is a set of constraints, **assignment** is the least cost function $a : A \to P(R)$ that uniquely assigns each agent to a subset of resources that satisfies the **constraint** and **cost** is the cost of **assignment**.

The branching rules is defined as follow:

(1) if there is no conflict (all subsets are disjoint), the node is a terminal

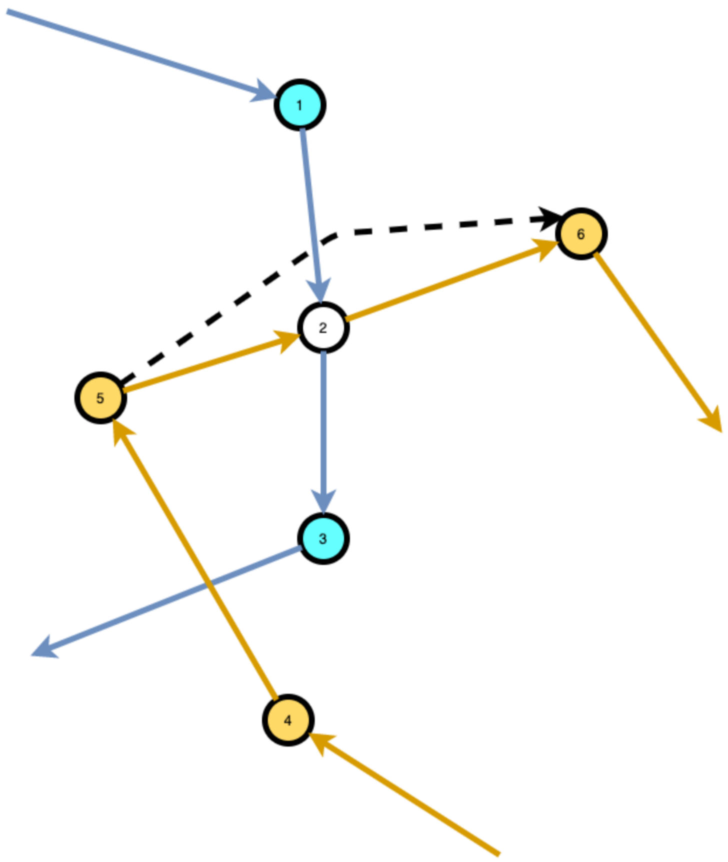

Figure 2.9: Naive Conflict

node.

(2) Choose a conflict (a resource in common of more than one agent), branch-out to $m+1$ child nodes where m is the number of agent taking that resource. In each of the first m child nodes, add a constraint assigning the respective agent to take the resource. In last child node, add a constraint preventing all agents to take the resource.

To elaborate more on the branching rule (2), consider a node with properties as follow:

constraint: $\{ c_1 \}$
assignment: $a_1 \rightarrow \{r_1, r_2\}, a_2 \rightarrow \{r_2, r_3\}, a_3 \rightarrow \{r_2, r_4\}$
cost: some real number.

Suppose we choose r_2 to branch-out, the first child node is:

constraint: $\{ c_1$, (assign r_2 to a_1) $\}$
assignment: least cost assignment satisfies the **constraint**
cost: some real number.

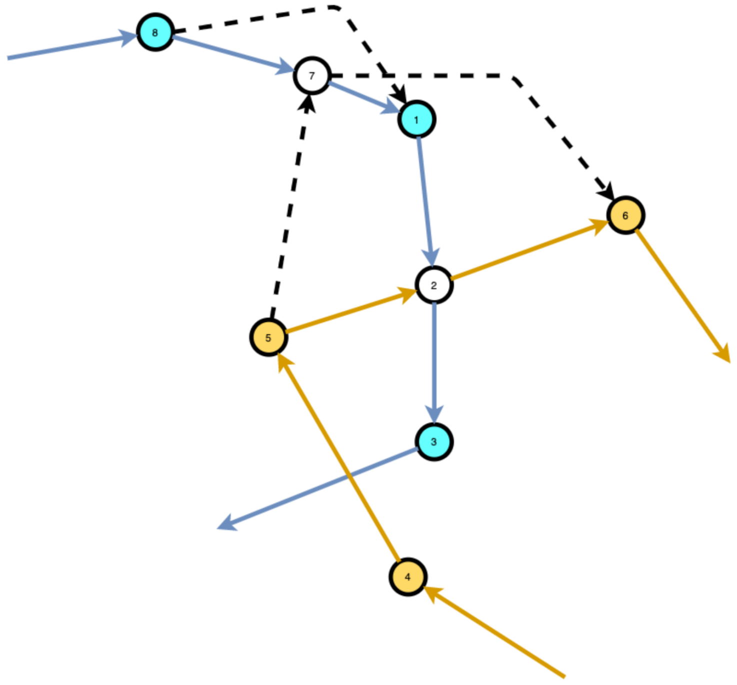

Figure 2.10: Harder Conflict

The next two child node is constructed in the same manner by replacing the constraint on a_1 to a_2 and a_3 respectively.

The last child node is:

constraint: { c_1, (do not assign r_2 to any of $\{a_1, a_2, a_3\}$) }
assignment: least cost assignment satisfies the **constraint**
cost: some real number.

By splitting the conflict tree node in this way, there is no duplicate node since $m+1$ child nodes split from a parent node are mutually exclusive. In the case of $m = 2$, the branching rules reduced to the method described in **?**.

Using best first search on the conflict tree guarantees to find the optimal assignment by the following two arguments:

Lemma 1 (Complete)**.** *Root node (empty* ***constraint****) permits the least cost terminal node if one exists.*

Lemma 2 (Lower bound)**.** *The cost of a conflict tree node is the lower bound of all terminal nodes it permits.*

Lemma **??** can be proved by verifying the conflict tree is limited depth and for every branch-out, the least cost terminal node belongs to either one of the branches.

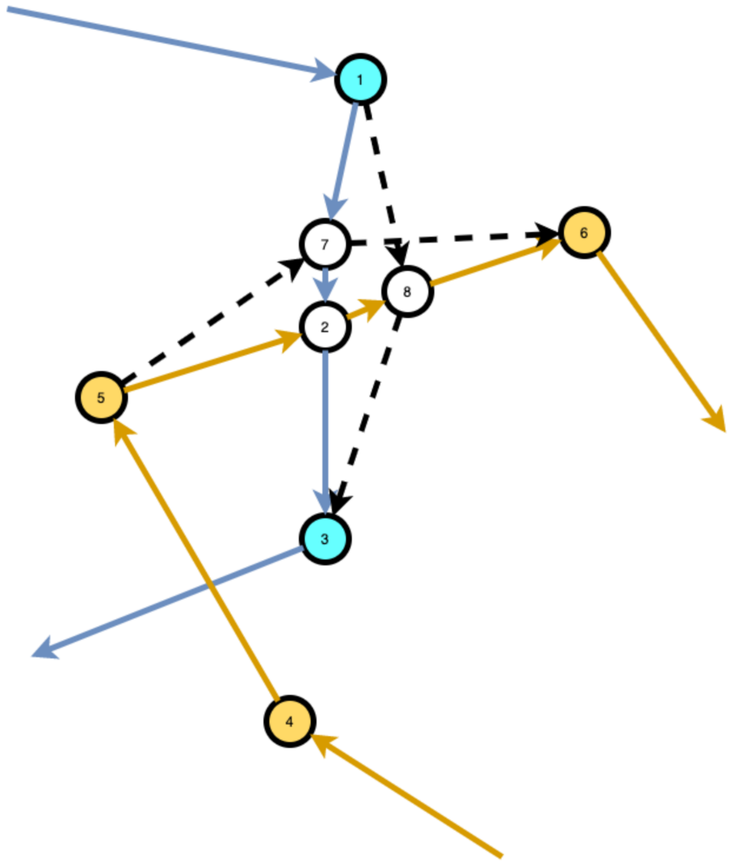

Figure 2.11: Hardest Conflict

Lemma **??** can be proved by contradiction since adding more constraints, the cost of a node cannot decrease.

Theorem 2 (Completeness of CBS)**.** *Conflict Based Search with best first search strategy guarantees to find the optimal assignment if one exists.*

In conflict avoidance, we consider each tour as an agent, a subset of resources corresponding to a possible plan for that tour. Conflict can be modelled as a resource shared by more than one agent (same node, same edge). We further impose two additional branch-out rules that makes the algorithm capable to explore its possible local advancement.

(1) **Transfer3**: the operation of transferring a node from a tour to another tour or itself.

(2) **Reverse**: reverse the direction of a tour.

Consider a series of POIs: $A \to B \to C \to D$. Let another tour occupied the path $C \to B$ (opposite direction). One possible strategy to is **reverse** the whole first tour, it becomes $D \to C \to B \to A$. Another strategy is the reschedule the first path to be $A \to C \to B \to D$, this rescheduled path can be obtained by **transfer3** of POI C to edge $A \to B$

Chapter 3

Appendix

3.1 Reduction

For an easier proof, consider these decision versions where triangle inequality satisfied.

Problem 4 (Full Problem Decision)**.** *Given a directed graph $G = (V, E)$ with corresponding cost function on each edge $c : E \to \mathbf{R}_+$ and a set of points of interest (POIs) $P \subseteq V$. Find K tours that visit V_P and the maximum tour cost does not exceed L satisfy a set of predefined constraints.*

Problem 5 (Partial Problem Decision)**.** *Given a directed graph $G_P = (V_P, E_P)$ with corresponding cost function on each edge $c_P : E_P \to \mathbf{R}_+$. Find K tours that visit V_P and the maximum tour cost does not exceed L.*

The problem **??*** (without any constraint) is equivalent to **??**.

Theorem 3 (Reduction)**.** *Problem* **??** ** (without any constraint) and problem* **??** *can be reduced from each other.*

Reduction 1 (problem **??*** $\to$ problem **??**)**.** *Given a 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.*

3.2 Proof of Reduction

3.2.1 problem ?? $\to$ 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 **??**

3.2.2 problem ??* $\to$ 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 **??***

3.3 Proof of Theorem ??

Consider the problem of minimizing a function $f : D \to \mathbb{R}$

In many scenarios, it is hard to find an optimal, or it is even also hard to compute the value of f. The method below was inspired from the work of Isaac Vandermeulen, Roderich Groß, Andreas Kolling **?**.

For a domain $X \subseteq D$ of the minimization problem, let $f_1 : X \to \mathbb{R}$ be the proxy function such that

$$(1) : f(x) = c(f_1(x)) + v(x) \ \forall x \in x$$

Where c is a monotonically increasing function and $v : X \to \mathbb{R}$ is function on X.

Let x^* and x_1^* be the optimal values for f and f_1 in the domain $X \subseteq D$. Let $t_{\max} = \max_{x \in X} v(x)$ and $t_{\min} = \min_{x \in X} v(x)$ be the maximum value and minimum value of v over the domain X. [1]

[1]Here, we used the maximum value and minimum value for v since in the original work **?**, the authors did not make it clear why x^* and x_1^* are independent from v and further more, their proof does not make a clear statement on the feasibility of the method to any problem but rather most problems.

Consider 3 inequalities:

$$(A) : f(x_1^*) \leq f(x^*) + t_{\max} - t_{\min}$$
$$(B) : f(x_1^*) \leq c(f_1(x_1^*)) + t_{\max}$$
$$(C) : f(x^*) \geq c(f_1(x^*)) + t_{\min}$$

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

By definition of $t_{\max}$, (B) holds,

$$f(x_1^*) \leq c(f_1(x_1^*)) + t_{\max} \leq c(f_1(x^*)) + t_{\max}$$

By definition of $t_{\min}$, (C) holds,

$$f(x^*) \geq c(f_1(x^*)) + t_{\min} = (c(f_1(x^*)) + t_{\max}) - (t_{\max} - t_{\min})$$

Hence,

$$(A) : f(x_1^*) \leq f(x^*) + (t_{\max} - t_{\min})$$

By choosing an appropriate proxy function f_1, we can approximate the solution of f.

In the derivation, we have set $t_{\max}$ and $t_{\min}$ be the maximum and minimum value of v in the domain of X. However, the bound can be even better if we have some methods to approximate the maximum and minimum value of v in a subset of X that contains both x^* and x_1^*

3.4 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

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

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 $\beta > 0$:

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

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

3.5 Theorem ??

Given the program:

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

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

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

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

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:

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

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

$$\textbf{Dual feasibility: } \mu_i \ge 0 \ \forall i$$

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

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, $\mu_{i1} \geq 0$, So

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

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

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

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

(1) deduces contradiction.