On A Capacitated Multivehicle Routing Problem

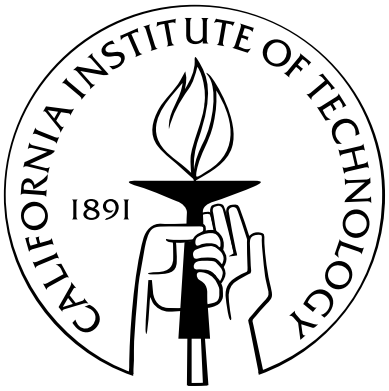
Thesis by

Xiaojie Gao

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2008

(Defended August 3, 2007)

Acknowledgements

First and foremost, it is my great pleasure to thank my advisor Professor Leonard J. Schulman. This thesis would never have existed without his help, support, inspiration, and guidance. To him, I offer my most sincere gratitude.

I wish to thank my fellow members of the theory group for their valuable discussions and helpful suggestions for my work.

In addition, my thanks go to Professor K. Mani Chandy, Professor Richard M. Murray, and Professor Chris Umans for helpful discussions. They provided me an education, formally or informally, that will be an invaluable resource in my future career.

I am particularly indebted to my whole family for their love, encouragement, and support, especially my parents who have always been there to offer guidance for me. I owe a lot to my dear husband Chih-Kai Ko, who have helped me in writing on this work and providing constructive suggestions.

I offer my deep thanks to all of my friends, both in the United States and in China, who have helped me in many aspects of daily life and study.

Abstract

The Vehicle Routing Problem (VRP) is a discrete optimization problem with high industrial relevance and high computational complexity. The problem has been extensively studied since it was introduced by Dantzig and Ramser. In a VRP, we are given a number of customers with known delivery requirements and locations (assumed to be vertices in a network). A fleet of vehicles with limited capacity is available. The objective is to design routes and customer assignments to minimize the total time or distance traveled to serve the demands. Because of its practical significance, this problem has been widely studied.

In this thesis, we present a version of the VRP motivated by mobile sensor networks which we call the Capacitated Multivehicle Routing Problem (CMVRP). In our framework, there are multiple geographically disperse vehicles each equipped with a limited energy supply. The vehicle consumes energy as it moves around and it also consumes energy while serving jobs. This situation models a network of mobile sensors where locomotion and computation all drain the limited capacity battery onboard. Our objective is to determine the minimum amount of energy required to serve all jobs, which takes into account both the service requirement and the travel overhead. We present a constant factor approximation algorithm. Furthermore, we study the on-line problem where job demands arrive sequentially and present a distributed algorithm that serves all jobs using only a constant factor more energy than the off-line solution.

Contents

Chapter 1

Introduction

First introduced by Dantzig and Ramser [5], the Vehicle Routing Problem (VRP) is a combinatorial optimization problem with applications to diverse areas such as resource allocation, load balancing, and sensor networks. In a VRP, we are given a number of customers with known delivery requirements and locations (assumed to be vertices in a network). A fleet of vehicles with limited capacity is available. The objective is to design routes and customer assignments to minimize the total time or distance traveled to serve the demands. Because of its practical significance, this problem has been widely studied [12, 17]. Unfortunately, like many combinatorial optimization problems, exact solutions of VRPs are often computationally intensive [7]. For the sake of efficiency, one must resort to approximation methods and heuristics which work well in practice [4, 19, 20, 9, 12, 8, 21].

In this thesis, we present a version on the VRP motivated by mobile sensor networks which we call the Capacitated Multivehicle Routing Problem (CMVRP). In our framework, there are multiple vehicles each equipped with a limited energy supply. The vehicle consumes energy as it moves around and it also consumes energy while serving jobs. This situation models a network of mobile sensors where locomotion and computation both drain the limited capacity battery onboard. Our objective is to determine the minimum amount of energy required to serve all jobs. We present a constant factor approximation algorithm. Furthermore, we also study the on-line problem where job demands arrive sequentially and present a distributed algorithm that serves all jobs using only a constant factor more energy than the off-line solution.

Before we present our formal problem definition, a review is in order.

1.1 Review

Let $G = (V, E)$ be an undirected weighted graph where vertices in V represent cities and edges in E represent roadways between pairs of cities. Associated with each road $e \in E$ is a non-negative weight $a(e)$ that denotes the distance between the cities (or sometimes the road toll). Let $\Omega \subseteq V$ denote the set of depots. Each depot $x \in \Omega$ initially contains $m(x)$ vehicles. Let $C \subseteq V$ denote the set of customers, each $x \in C$ has demand $d(x) \geq 0$. One can imagine the demand as the number of units of a good that a customer requires. The general goal is to design an optimal set of routes and/or schedules for each vehicle in order to satisfy all customer demands.

There are numerous variations on this VRP model. We shall briefly discuss a few of them and refer the reader to excellent surveys of Bodin and Golden [1] and Laporte [12] for further details.

- *(Original) Vehicle Routing Problem:* Multiple vehicles are dispatched from a central depot to serve customers. Each vehicle travels with unit speed. The goal is to minimize the time required to reach all customers. See [5]. If we replace the time requirement with a minimum total distance requirement, then we have the classic Traveling Salesman Problem (TSP).

- *Capacitated Vehicle Routing Problem (CVRP):* Multiple vehicles, which are labeled $1, 2, 3, \cdots, m$, are dispatched from a central depot to serve customers. Vehicle $i \in \{1, \ldots, m\}$ has service capacity $w(i)$ and each customer $x \in C$ has demand $d(x)$. The objective is to find the shortest vehicle routes (minimizing the total length of all vehicle routes) so that all customer demands are satisfied. See [14]. One can also consider finding the fastest routes (minimizing the longest route over all vehicle routes). See [13]. Sometimes, vehicles are re-stocked at depot locations and the problem becomes a pickup and delivery problem [11].

- *Vehicle Routing Problem with Time Windows (VRPTW):* Same basic framework as CVRP with the additional requirement that each customer must have all her jobs served within a given time window. See [2, 3, 16].

In most of the existing VRP literature, all vehicles originate from a central depot. In our version, we have many geographically disperse depots as well as customers. Our energy objective takes into account both the customer service requirement and the travel overhead, i.e., the vehicle capacity need be at least customer service cost plus travel overhead.

1.2 Motivation

One motivation of our work stems from the Smart Dust project [18].

Smart Dust is a hypothetical network of tiny wireless micro-electromechanical systems (MEMS) sensors, robots, or devices, installed with wireless communication capabilities, that can detect anything from light and temperature to vibrations. A typical application scenario is the scattering of hundreds of these sensors around a building to monitor temperature or humidity; or around the seabed to monitor seismic activity. In a military setting, they can act as remote sensors to track enemy movements, detect poisonous gas or radioactivity.

In our model, in which the sensors (robots or vehicles) have modest mobility, we not only provide coverage, like in Smart Dust, but also increase the robustness and longevity of the network. If one micro-robot dies, the rest of them can shift and cover for the missing micro-robot, and the task can still be completed.

Although our work is entirely at the theoretical level, it has foreseeable applications as current robotics researchers are already working on such tiny mobile sensors. For instance, Pister's "Smart Dust with Legs":

> *"It's a startling idea: Swarms of ant-size robots burrowing through the rubble of a building after an earthquake searching for survivors or crawling onto the hull of a spacecraft to repair damage in-flight. But perhaps the*

most amazing thing about Pister's dream is that it's not as far off as one might think. Already Pister and his graduate students have built simple solar-powered microrobots just 8.5 millimeters long and less than 4 millimeters wide."

(Quoted from http://www.coe.berkeley.edu/labnotes/0903/pister.html.)

The development of networking protocols for mobile Smart Dust represents a significant challenge [10].

1.3 Problem Statement

Let $G = (V, E)$ be the ℓ-dimensional grid $\mathbb{Z}^\ell$ (assume ℓ is a constant), let $C = \Omega = V$. That is, at each city $x \in V$, there is one customer and one depot. Inside each depot is a vehicle with capacity W. The traveling cost between any two adjacent points in V is 1 unit of energy.

Consider a sequence of k service requests (jobs) arriving at positions (cities) $x_1, x_2, \cdots, x_k \in \mathbb{Z}^\ell$ at times $t_1, t_2, \cdots, t_k$ where $t_1 < t_2 < \cdots < t_k$. Assume that each service request requires 1 unit of energy to process and denote by $d(x)$ the total service demand at position $x \in \mathbb{Z}^\ell$:

$$d(x) = \sum_{i=1}^{k} \mathbf{I}(x, x_i),$$

where the indicator function $\mathbf{I}(x, y)$ is 1 if $x = y$ and 0 otherwise.

We further distinguish between these two different scenarios:

- *Off-line:* The demand function $d(\cdot)$ and arrival sequence are known to all vehicles at the beginning.

- *On-line:* The demand function $d(\cdot)$ and arrival sequence are *not* known to all vehicles at the beginning. Since we need to process the request immediately (or with a small constant delay), we impose the additional requirement that for all

$i \in \{1, \ldots, k\}$ and for all $x \in \mathbb{Z}^\ell$, there exists a vehicle with non-zero energy within a constant distance of x at time t_i.

Our objective is to determine the minimal W such that all service requests can be satisfied. We let W_{off} and W_{on} denote the minimal W in the off-line and on-line cases respectively.

1.4 Main Contributions

Define $N_r(T)$ to be the neighborhood of radius r around T, where $T \subseteq \mathbb{Z}^\ell$ and $N_r(x)$ to be the neighborhood of radius r around x, where $x \in \mathbb{Z}^\ell$. In symbols,

$$N_r(T) = \{y \ : \ \exists \ x \in T \text{ such that } \|x - y\| \leq r\}$$

and

$$N_r(x) = \{y \ : \|x - y\| \leq r\},$$

where $\|x - y\|$ is the Manhattan distance[1] between x and y.

Given a nonempty subset $T \subseteq \mathbb{Z}^\ell$, let ω_T denote the solution of the equation:

$$\omega_T \cdot |N_{\omega_T}(T)| = \sum_{x \in T} d(x), \tag{1.1}$$

where $|N_{\omega_T}(T)|$ is the cardinality of set $N_{\omega_T}(T)$. Since the left hand side is strictly increasing in ω_T, a solution of (1.1) always exists and is unique.

Theorem 1.4.1 $W_{off} = \Theta\left(\max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T\right)$.

Theorem 1.4.2 $W_{on} = \Theta\left(W_{off}\right) = \Theta\left(\max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T\right)$.

We design an algorithm to achieve (up to a constant factor) the optimal W_{off}. We also present a strategy for moving vehicles for the on-line case that fulfills the job requirements.

[1] Some authors refer to this as the rectilinear distance, L_1 distance, or city block distance.

The rest of the thesis is organized as follows: Chapters 2 and 3 will discuss the off-line and on-line cases respectively. In Chapter 4, we extend the problem to allow a number of vehicles breaking down. In Chapter 5, we explore the possibility of energy transfers between vehicles. Finally, we conclude with some future research directions in Chapter 6.

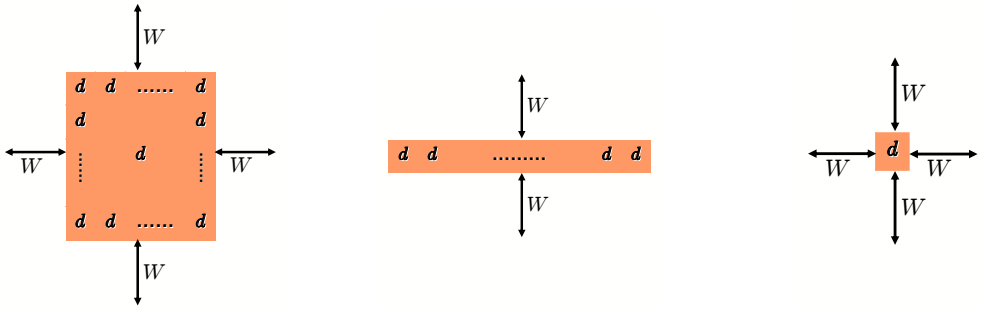
Chapter 2

Off-line Case

In this chapter, we focus on the off-line case of our Capacitated Multivehicle Routing Problem (CMVRP). As illustration, three special examples are given in Section 2.1. The characterization of optimal off-line performance (i.e. proof of Theorem 1.4.1) is provided in Section 2.2. A linear-time approximation algorithm to compute W_{off} is given in Section 2.3.

2.1 Examples

In this section, we will use three simple but practical 2-dimensional examples to give some intuition for the problem we are going to solve.



(a) Square (b) Line (c) Point

Figure 2.1: Examples of three special cases. (a) The demand is d at any point in a square and 0 at any point outside the square. (b) The demand is d at any point on a line and 0 at any other point. (c) The demand is d at a single point and 0 at any other point.

2.1.1 Example 1: All Demands Are in a Square

In this example, the demand is d at any point in a square T of size $a \times a$ and 0 at any point outside the square, which is shown in Figure 2.1(a).

Since the traveling cost between any two adjacent points is 1 unit of energy, only those vehicles within distance W could get into the area T. Therefore, $W \times (2W + a)^2 \geq d \times a^2$. Let W_1 be the solution to $W \times (2W + a)^2 = d \times a^2$, where W is the variable. Then $W \geq W_1$. When a approaches infinity, W approaches d.

2.1.2 Example 2: All Demands Are on a Line

In this example, the demand is d at any point on a line L and 0 at any other point, which is shown in Figure 2.1(b). It is a reasonable and practical model when using the mobile vehicles to detect the traffic flow on the highway.

Since the traveling cost between any two adjacent points is 1 unit of energy and only those vehicles within distance W could get onto the line L, we have $W \times (2W + 1) \geq d$. Let W_2 be the solution to $W \times (2W + 1) = d$, where W is the variable, then $W \geq W_2$. If each vehicle has a capacity $W = 2W_2$, there is a way to serve all the demands: Any vehicle in the neighborhood of radius W_2 around L, i.e. $N_{W_2}(L)$, moves to its nearest point on the line L. The traveling cost for each vehicle is at most W_2 units of energy and the remaining energy could be used to serve the demands. This is shown in Figure 2.2. Therefore, $W^2 \sim d$.

2.1.3 Example 3: All Demands Are in a Single Point

In this example, the demand is d at a single point p and 0 at any other point, which is shown in Figure 2.1(c). It is a reasonable model when using the mobile vehicles to detect the earthquake.

Only those vehicles within distance W could get into p. We have $W \times (2W+1)^2 \geq d$. Let W_3 be the solution to $W \times (2W + 1)^2 = d$, where W is the variable, then $W \geq W_3$. If each vehicle has a capacity $W = 3W_3$, there is a way to serve all the demands: Any vehicle in the square of size $(2W_3 + 1) \times (2W_3 + 1)$, whose center is

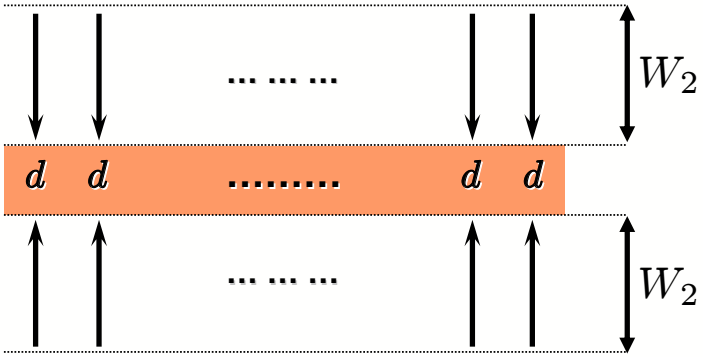


Figure 2.2: Any vehicle in the neighborhood of radius W_2 around the line L, i.e. $N_{W_2}(L)$, moves to its nearest point on the line.

at point p, moves to p. The traveling cost for each vehicle is at most $2W_3$ units of energy and the remaining energy is enough to serve the demands. This is shown in Figure 2.3. Therefore, $W^3 \sim d$.

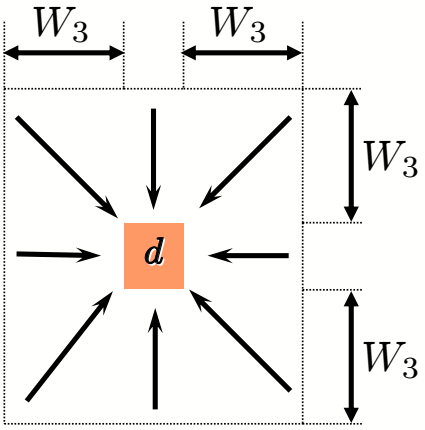


Figure 2.3: Any vehicle in the square of size $(2W_3 + 1) \times (2W_3 + 1)$, whose center is at point p, moves to p.

2.2 Characterization of Optimal Off-line Performance

Before diving into the proof of Theorem 1.4.1, it is instructive to first study a simpler problem: Let us ignore the energy expenditure due to locomotion. That is, we shall only take into account the customer service requirement and neglect the travel

overhead. But we confine each vehicle to within a local neighborhood of a specified radius $r > 0$ around its starting point. Later in the section, we will set the radius r to be the same as the vehicle capacity.

The minimal capacity required for resolving the supply-demand transports with a specified radius parameter r is the solution of the following linear program (LP):

$$\begin{aligned} &\text{minimize} \quad \omega \\ &\text{s.t.} \quad \begin{cases} \sum\limits_{j \in N_r(i)} f_{ij} \le \omega, & \forall\ i \in \mathbb{Z}^\ell \\ \sum\limits_{i \in N_r(j)} f_{ij} \ge d(j), & \forall\ j \in \mathbb{Z}^\ell \\ f_{ij} \ge 0, & \forall\ i, j \in \mathbb{Z}^\ell \ \text{ and } \ \|i - j\| \le r, \end{cases} \end{aligned} \tag{2.1}$$

where $F = \{f_{ij}\}$ denotes the set of "flows." Each flow f_{ij} represents the amount of energy transported by the vehicle at position i to position j.

Notice that (2.1) is not the classical LP used in the classical "Transportation Problem" [15] in two important aspects:

- In the Transportation Problem, both the supply distribution (how much energy is in each vehicle) and demand distribution (how much energy is needed at each position) are known a priori. The goal is to find the minimal cost (i.e. the Earthmover Distance [15]) to transform one distribution into the other. In (2.1), the supply distribution is part of the linear program, which is to be solved. The method we use here is different from those methods used in the classical Transportation Problem.

- In the Transportation Problem, there is either no distance constraint or fixed distance constraint. In our case, the transport distance is bounded by a parameter r and later by the supply (vehicle capacity), which is to be solved.

Before trying to solve (2.1), we first give a lemma as follows.

Lemma 2.2.1 *Given $d(j) \geq 0$ for all $j \in \mathbb{Z}^\ell$, the linear programming*

$$\begin{array}{ll} \textit{maximize} & \displaystyle\sum_{j\in\mathbb{Z}^\ell} \left(d(j) \times \min_{i:\|i-j\|\leq r} \alpha_i \right) \\ \textit{s.t.} & \begin{cases} \displaystyle\sum_{i\in\mathbb{Z}^\ell} \alpha_i \leq 1, \\ \alpha_i \geq 0, \quad \forall\ i \in \mathbb{Z}^\ell \end{cases} \end{array} \tag{2.2}$$

is equivalent to

$$\begin{array}{ll} \textit{maximize} & \displaystyle\sum_{j} d(j) \sum_{T:N_r(j)\subseteq T} h(T) \\ \textit{s.t.} & \begin{cases} \displaystyle\sum_{T\subseteq\mathbb{Z}^\ell} h(T)|T| \leq 1 \\ h(T) \geq 0,\ \forall\ T \subseteq \mathbb{Z}^\ell \end{cases} \end{array} . \tag{2.3}$$

where h denotes a mapping from the set of subsets of $\mathbb{Z}^\ell$ to $\mathbb{R}$.

Proof : We will prove the lemma by two steps:

1. The solution of (2.2) is at most the solution of (2.3).

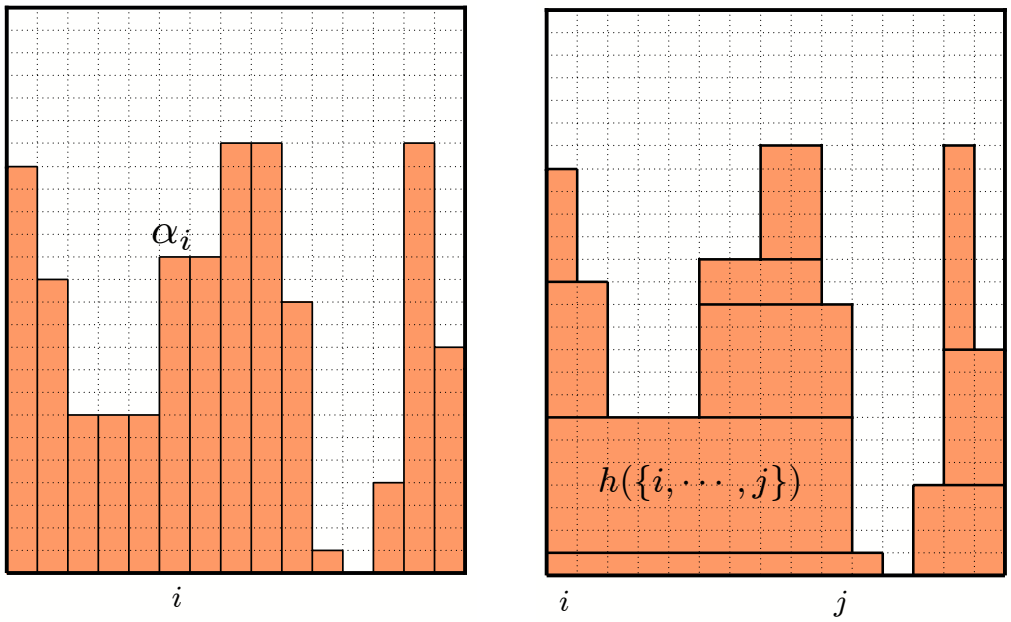
   For any $(\alpha_i)_{i\in\mathbb{Z}^\ell}$ that solves (2.2), we can define a mapping h from the set of subsets of $\mathbb{Z}^\ell$ to $\mathbb{R}$, that is, for any $T \subseteq \mathbb{Z}^\ell$,

$$h(T) = \begin{cases} \max\left\{0, \min_{i\in T} \alpha_i - \max_{i\in N_1(T)\setminus T} \alpha_i\right\} & \text{if } T \text{ is simply connected} \\ 0 & \text{otherwise} \end{cases} .$$

   Given the value of $(\alpha_i)_{i\in\mathbb{Z}^\ell}$ as seen in Figure 2.4(a), h (in Figure 2.4(b)) can be deduced by finding the maximal α_i's first and setting the value of those subsets to be the difference between the maximal value and the smaller value on the boundary; then reducing the value of $(\alpha_i)_{i\in\mathbb{Z}^\ell}$ on those subsets and repeating the first step. Figure 2.5 gives an example of the first a few steps of the process.

   Look at any two simply connected subsets T_1, T_2 of $\mathbb{Z}^\ell$, which are related as follows:

$$T_1 \cap T_2 \neq \emptyset,\ T_1 \setminus T_2 \neq \emptyset, \text{ and } T_1 \setminus T_2 \neq \emptyset.$$



(a) $(\alpha_i)_{i\in\mathbb{Z}^\ell}$ (b) h

Figure 2.4: An illustration of the relationship between $(\alpha_i)_{i\in\mathbb{Z}^\ell}$ and h in 1-dimensional space.

Suppose $h(T_1) \neq 0$, i.e., $\min\limits_{i\in T_1} \alpha_i > \max\limits_{i\in N_1(T_1)\setminus T_1} \alpha_i$. Let x be a vertex in set $T_1 \cap N_1(T_2)\setminus T_2$ and y be a vertex in set $T_2 \cap N_1(T_1)\setminus T_1$. Then

$$\max_{i\in N_1(T_2)\setminus T_2} \alpha_i \geq \alpha_x \geq \min_{i\in T_1} \alpha_i \geq \max_{i\in N_1(T_1)\setminus T_1} \alpha_i \geq \alpha_y \geq \min_{i\in T_2} \alpha_i.$$

So $h(T_2) = 0$. Therefore, for any two simply connected subsets T_1, T_2 of $\mathbb{Z}^\ell$, if $h(T_1) \neq 0$ and $h(T_2) \neq 0$, then

$$T_1 \subset T_2 \text{ or } T_1 \supseteq T_2 \text{ or } T_1 \cap T_2 = \emptyset.$$

For any $i \in \mathbb{Z}^\ell$, there exist a sequence of sets $T_1, T_2, \cdots, T_\kappa$ such that

- $i \in T_1 \subset T_2 \subset \cdots \subset T_\kappa \subseteq \mathbb{Z}^\ell$,
- $h(T_j) \neq 0$ for any $j \in \{1, 2, \cdots, \kappa\}$,
- and for any other T such that $T \notin \{T_1, T_2, \cdots, T_\kappa\}$ and $i \in T$, $h(T) = 0$.

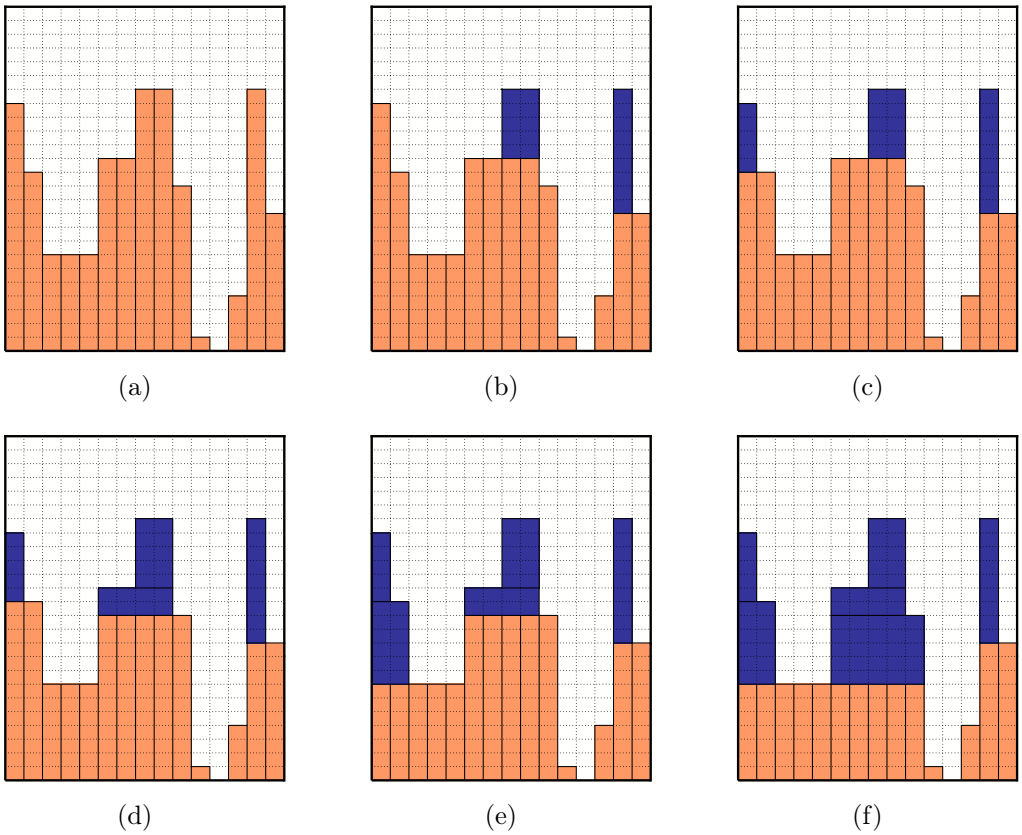


Figure 2.5: An example showing the first a few steps to deduce h from $(\alpha_i)_{i\in\mathbb{Z}^\ell}$.

From the definition of h, there exists an $x_j \in T_j \backslash T_{j-1}$ for each $j \in \{2, 3, \cdots, \kappa\}$ such that

$$h(T_j) = \begin{cases} \alpha_i - \alpha_{x_2}, & j = 1 \\ \alpha_{x_j} - \alpha_{x_{j+1}}, & j \in \{2, \cdots, \kappa - 1\} \\ \alpha_{x_\kappa}, & j = \kappa \end{cases}.$$

Thus,

$$\alpha_i = \sum_{j=1}^{\kappa} h(T_j) = \sum_{T: i \in T} h(T),$$

and

$$\sum_{T \subseteq \mathbb{Z}^\ell} h(T)|T| = \sum_{T \subseteq \mathbb{Z}^\ell} h(T) \sum_{i: i \in T} 1 = \sum_{i: i \in \mathbb{Z}^\ell} \sum_{T: i \in T} h(T) = \sum_{i: i \in \mathbb{Z}^\ell} \alpha_i \leq 1.$$

Given $j \in \mathbb{Z}^\ell$, let $x = \arg\min_{i: \|i-j\| \leq r} \alpha_i$. For any T such that $x \in T$ and $h(T) \neq 0$, if $N_r(j) \nsubseteq T$, then $\alpha_x > \alpha_y$ for any $y \in N_r(j) \cap N_1(T) \backslash T$, a contradiction with $\alpha_x = \min_{i: \|i-j\| \leq r} \alpha_i \leq \alpha_y$. Therefore,

$$\min_{i: \|i-j\| \leq r} \alpha_i = \sum_{T: N_r(j) \subseteq T} h(T) \text{ for any } j \in \mathbb{Z}^\ell.$$

2. The solution of (2.3) is at most the solution of (2.2).

   For any h that solves (2.3), we can define $\alpha_i = \sum_{T: i \in T} h(T)$ for all $i \in \mathbb{Z}^\ell$. From the definition, $\min_{i: \|i-j\| \leq r} \alpha_i \geq \sum_{T: N_r(i) \subseteq T} h(T)$ for any $j \in \mathbb{Z}^\ell$, and

$$\sum_{i \in \mathbb{Z}^\ell} \alpha_i = \sum_{i \in \mathbb{Z}^\ell} \sum_{T: i \in T} h(T) = \sum_{T \subseteq \mathbb{Z}^\ell} h(T) \sum_{i: i \in T} 1 = \sum_{T \subseteq \mathbb{Z}^\ell} h(T)|T| \leq 1.$$

□

We are now ready to solve (2.1):

Lemma 2.2.2 *Given $d(x) \geq 0$ for all $x \in \mathbb{Z}^\ell$ and $r > 0$. The value of LP (2.1) is $\max_{T: T \subseteq \mathbb{Z}^\ell} \frac{\sum_{x \in T} d(x)}{|N_r(T)|}$.*

Table 1 Linear programming Primal and Dual

Primal		**Dual**	
$\max \sum_{j=1}^{n} c_j x_j$		$\min \sum_{i=1}^{m} b_i y_i$	
s.t. $\sum_{j=1}^{n} a_{ij} x_j \le b_i$	$(i = 1, \cdots, m)$	s.t. $\sum_{i=1}^{m} a_{ij} y_i \ge c_j$	$(j = 1, \cdots, n)$
$x_j \ge 0$	$(j = 1, \cdots, n)$	$y_i \ge 0$	$(i = 1, \cdots, m)$

Proof : The dual of (2.1) is the following.

$$\begin{array}{ll} \text{maximize} & \sum\limits_{j \in \mathbb{Z}^\ell} \beta_j d(j) \\ \text{s.t.} & \begin{cases} \sum\limits_{i \in \mathbb{Z}^\ell} \alpha_i \le 1, & \\ \alpha_i \ge 0, & \forall\ i \in \mathbb{Z}^\ell \\ \beta_j \ge 0, & \forall\ j \in \mathbb{Z}^\ell \\ \beta_j \le \alpha_i, & \forall\ i, j \in \mathbb{Z}^\ell \ \text{ and } \ \|i - j\| \le r \end{cases} . \end{array} \tag{2.4}$$

Since $\beta_j \le \alpha_i$ for all $\|i - j\| \le r$, we know that $\beta_i \le \min\limits_{j:\|i-j\| \le r} \alpha_j$. We want to maximize $\sum\limits_{j \in \mathbb{Z}^\ell} \beta_j d(j)$ with $d(j) \ge 0$, which implies that $\beta_i = \min\limits_{j:\|i-j\| \le r} \alpha_j$ and (2.4) becomes the following LP:

$$\begin{array}{ll} \text{maximize} & \sum\limits_{j \in \mathbb{Z}^\ell} \left(d(j) \times \min\limits_{i:\|i-j\| \le r} \alpha_i \right) \\ \text{s.t.} & \begin{cases} \sum\limits_{i \in \mathbb{Z}^\ell} \alpha_i \le 1, & \\ \alpha_i \ge 0, & \forall\ i \in \mathbb{Z}^\ell \end{cases} . \end{array} \tag{2.5}$$

Let h denote a mapping from the set of subsets of $\mathbb{Z}^\ell$ to $\mathbb{R}$. By Lemma 2.2.1, the

LP (2.5) is equivalent to

$$\begin{aligned} \text{maximize} \quad & \sum_j d(j) \sum_{T:N_r(j)\subseteq T} h(T) \\ \text{s.t.} \quad & \begin{cases} \sum_{T\subseteq\mathbb{Z}^\ell} h(T)|T| \le 1 \\ h(T) \ge 0, \ \forall\ T \subseteq \mathbb{Z}^\ell \end{cases} . \end{aligned} \tag{2.6}$$

Since $\sum_i d(i) \sum_{T:N_r(i)\subseteq T} h(T) = \sum_{T\subseteq\mathbb{Z}^\ell} h(T) \sum_{i:N_r(i)\subseteq T} d(i)$, the dual of (2.6) is

$$\begin{aligned} \text{minimize} \quad & \omega \\ \text{s.t.} \quad & \omega \cdot |T| \ge \sum_{i:N_r(i)\subseteq T} d(i), \ \forall\ T \subseteq \mathbb{Z}^\ell. \end{aligned} \tag{2.7}$$

Therefore, the solution of (2.1) is

$$\omega = \max_{T:T\subseteq\mathbb{Z}^\ell} \frac{\sum_{i:N_r(i)\subseteq T} d(i)}{|T|} = \max_{T:T\subseteq\mathbb{Z}^\ell} \frac{\sum_{x\in T} d(x)}{|N_r(T)|}.$$

□

If we set the radius r to be ω, LP (2.1) becomes

$$\begin{aligned} \text{minimize} \quad & \omega \\ \text{s.t.} \quad & \begin{cases} \sum_{j\in N_\omega(i)} f_{ij} \le \omega, & \forall\ i \in \mathbb{Z}^\ell \\ \sum_{i\in N_\omega(j)} f_{ij} \ge d(j), & \forall\ j \in \mathbb{Z}^\ell \\ f_{ij} \ge 0, & \forall\ i,j \in \mathbb{Z}^\ell\ \&\ \|i-j\| \le \omega \end{cases} \end{aligned} \tag{2.8}$$

and the solution is as follows:

Lemma 2.2.3 *Given $d(x) \ge 0$ for all $x \in \mathbb{Z}^\ell$, the solution to program (2.8) is*

$$\max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T,$$

where ω_T is the solution of $\omega_T \cdot |N_{\omega_T}(T)| = \sum_{x\in T} d(x)$.

Proof : Let $\omega(r) = \max\limits_{T\subseteq\mathbb{Z}^\ell} \frac{\sum\limits_{x\in T} d(x)}{|N_r(T)|}$. For any $r_1 > r_2$, define

$$T_1 = \arg\max_{T\subseteq\mathbb{Z}^\ell} \frac{\sum\limits_{x\in T} d(x)}{|N_{r_1}(T)|}, \quad \text{and} \quad T_2 = \arg\max_{T\subseteq\mathbb{Z}^\ell} \frac{\sum\limits_{x\in T} d(x)}{|N_{r_2}(T)|}.$$

We see that $\omega(r)$ is a non-increasing function of r because

$$\begin{aligned}
\omega(r_1) - \omega(r_2) &= \max_{T\subseteq\mathbb{Z}^\ell} \frac{\sum_{x\in T} d(x)}{|N_{r_1}(T)|} - \max_{T\subseteq\mathbb{Z}^\ell} \frac{\sum_{x\in T} d(x)}{|N_{r_2}(T)|} \\
&= \frac{\sum_{x\in T_1} d(x)}{|N_{r_1}(T_1)|} - \frac{\sum_{x\in T_2} d(x)}{|N_{r_2}(T_2)|} \\
&\leq \frac{\sum_{x\in T_1} d(x)}{|N_{r_1}(T_1)|} - \frac{\sum_{x\in T_1} d(x)}{|N_{r_2}(T_1)|} \\
&\leq 0.
\end{aligned}$$

Together with Lemma 2.2.2, the solution to program (2.8) is equal to the unique solution of the equation

$$\omega = \max_{T\subseteq\mathbb{Z}^\ell} \frac{\sum\limits_{x\in T} d(x)}{|N_\omega(T)|}, \tag{2.9}$$

where ω is the variable. Let ω^* be the solution of (2.8) and (2.9).

For any $T' \subseteq \mathbb{Z}^\ell$,

$$\omega^* = \max_{T\subseteq\mathbb{Z}^\ell} \frac{\sum\limits_{x\in T} d(x)}{|N_{\omega^*}(T)|} \geq \frac{\sum\limits_{x\in T'} d(x)}{|N_{\omega^*}(T')|}. \tag{2.10}$$

Let ω' be the solution to the equation $\omega = \frac{\sum\limits_{x\in T'} d(x)}{|N_\omega(T')|}$. If $\omega' > \omega^*$, then

$$\frac{\sum\limits_{x\in T'} d(x)}{|N_{\omega^*}(T')|} \geq \frac{\sum\limits_{x\in T'} d(x)}{|N_{\omega'}(T')|} = \omega'. \tag{2.11}$$

Combining (2.10) with (2.11), we have $\omega^* \geq \omega'$, a contradiction with the assumption. Hence, $\omega^* \geq \omega'$. Therefore, $\omega^* = \max\limits_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$, where ω_T is the solution of $\omega_T \cdot |N_{\omega_T}(T)| = \sum_{x\in T} d(x)$. □

As used in the proof of Lemma 2.2.3, let $\omega^* = \max\limits_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$. Clearly, ω^* is a lower bound on W_{off}, because ω^* is the solution of program (2.8) which does not take into

account the cost of motion.

Corollary 2.2.4 $W_{off} \geq \omega^* = \max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$.

We are now ready for the upper bound on W_{off}.

Lemma 2.2.5 $W_{off} \leq (2 \cdot 3^\ell + \ell) \cdot \omega^* = (2 \cdot 3^\ell + \ell) \cdot \max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$.

Remark: *The primary interest in many applications is the plane ($\ell = 2$). In the plane, the upper bound is only a modest (and probably pessimistic) factor over the lower bound. However, for generality, we perform the analysis for general ℓ.*

Proof : In LP (2.8), for any position $x \in \mathbb{Z}^\ell$, only those vehicles within distance ω could move to x. According to Lemma 2.2.3, the amount of energy needed in any $\underbrace{\lceil\omega^*\rceil \times \lceil\omega^*\rceil \times \cdots \times \lceil\omega^*\rceil}_{\ell}$ ℓ-cube is at most $\omega^* \cdot (3\lceil\omega^*\rceil)^\ell$, excluding the travel overhead. We partition the grid $\mathbb{Z}^\ell$ into $\lceil\omega^*\rceil \times \lceil\omega^*\rceil \times \cdots \times \lceil\omega^*\rceil$ ℓ-cubes and provide each vehicle with $2 \cdot 3^\ell \cdot \omega^*$ units of energy to serve customer demands. Thus, every vehicle only needs to move and deliver energy inside its own $\lceil\omega^*\rceil \times \lceil\omega^*\rceil \times \cdots \times \lceil\omega^*\rceil$ ℓ-cube. In every such ℓ-cube $\mathbb{H}$, we have

$$\sum_{x\in\mathbb{H}} \left\lceil \frac{d(x) - 3^\ell \cdot \omega^*}{3^\ell \cdot \omega^*} \right\rceil \leq \sum_{x\in\mathbb{H}} \frac{d(x)}{3^\ell \cdot \omega^*} \leq \lceil\omega^*\rceil^\ell.$$

Let each vehicle use (leave) at most $3^\ell \cdot \omega^*$ energy first at its original position, and then move to a particular position and serve the requests there. The traveling overhead for any vehicle is at most $\ell \cdot \omega^*$. Therefore, at most $(2 \cdot 3^\ell + \ell) \cdot \omega^*$ energy is needed for each vehicle. □

Theorem 1.4.1 follows from Corollary 2.2.4 and Lemma 2.2.5. We also have the following two corollaries:

Corollary 2.2.6 *Let Γ be the set of all ℓ-cubes in $\mathbb{Z}^\ell$ and ω_T denote the solution of $\omega_T \cdot |N_{\omega_T}(T)| = \sum_{x\in T} d(x)$, then*

$$\max_{T:T\in\Gamma} \omega_T \leq W_{off} \leq (2 \cdot 3^\ell + \ell) \cdot \max_{T:T\in\Gamma} \omega_T.$$

Proof : Since $\Gamma \subseteq \{T : T \subseteq \mathbb{Z}^\ell\}$,

$$\max_{T:T\in\Gamma} \omega_T \leq \max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T \leq W_{\text{off}}.$$

$W_{\text{off}} \leq (2 \cdot 3^\ell + \ell) \cdot \max_{T:T\in\Gamma} \omega_T$ follows as in the proof of Lemma 2.2.5. □

Corollary 2.2.7 *Let Γ_ω be the set of all $\lceil\omega\rceil \times \lceil\omega\rceil \times \cdots \times \lceil\omega\rceil$ ℓ-cubes in $\mathbb{Z}^\ell$. Define* $\omega_c = \min\left\{\omega : \omega \cdot (3\lceil\omega\rceil)^\ell = \max_{T\in\Gamma_\omega} \sum_{x\in T} d(x)\right\}$.

$$\omega_c \leq W_{off} \leq (2 \cdot 3^\ell + \ell) \cdot \omega_c.$$

Proof : Assume that ω_c is achieved when $T = T_c$. Then,

$$\omega_c \cdot (3\lceil\omega_c\rceil)^\ell = \sum_{x\in T_c} d(x) = \omega_{T_c} \cdot |N_{\omega_{T_c}}(T_c)|.$$

So we must have $\omega_c \leq \omega_{T_c}$ because otherwise $\omega_c \cdot (3\lceil\omega_c\rceil)^\ell > \omega_{T_c} \cdot |N_{\omega_{T_c}}(T_c)|$. Therefore, $\omega_c \leq \max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$ and $\omega_c \leq W_{\text{off}}$. Finally, $W_{\text{off}} \leq (2 \cdot 3^\ell + \ell) \cdot \omega_c$ follows as in the proof of Lemma 2.2.5. □

The characterization using cubes is much simpler than T ranging over all subsets of $\mathbb{Z}^\ell$. The fact that we only need to examine cubes in $\mathbb{Z}^\ell$ is key to being able to provide an algorithm.

2.3 Approximation Algorithm to Compute W_{off}

For simplicity, we shall restrict our analysis to 2-dimensions ($\ell = 2$). The derivation for higher dimensions are straightforward extensions. We further assume that the graph is the $n \times n$ grid $\mathbb{Z}_n \times \mathbb{Z}_n$ where n is a power of 2 and the demand function is $d(x) \geq 0$ for every $x \in \mathbb{Z}_n^2$. Define

- The maximal demand is $D = \max_{x\in\mathbb{Z}_n^2} d(x)$.
- The average demand is $\hat{D} = \frac{\sum_{x\in\mathbb{Z}_n^2} d(x)}{n^2}$.

Algorithm 1 Approximation algorithm to compute W_{off}, running in linear time

```
1  if n ≤ D̂
2      return min{D, 2 · D̂ + ℓ · n};
3  if D ≤ 1
4      return D;
5  w ← 2, n′ ← n/w, d_1(i, j) ← d(i, j);
6  if w = n
7      return min{D, 2 · D̂ + ℓ · n};
8  for 1 ≤ i ≤ n′, 1 ≤ j ≤ n′
9      d_w(i, j) ← d_{w/2}(2i − 1, 2j − 1) + d_{w/2}(2i − 1, 2j) + d_{w/2}(2i, 2j − 1)
                  + d_{w/2}(2i, 2j);
10 if there exists a d_w(i, j) > w · (3w)^ℓ
11     w ← 2w, n′ ← n′/2;
12     goto 6;
13 else
14     return (2 · 3^ℓ + ℓ) · w.
```

Notice that W_{off} has the following properties:

Property 2.3.1 *$\hat{D} \leq W_{off} \leq D$.*

Proof : This is straightforward from the definition of $\hat{D}$ and D. □

Property 2.3.2 *If $D \leq 1$, then $W_{off} = D$.*

Proof : From 2.3.1, $W_{\text{off}} \leq D$, so $W_{\text{off}} \leq 1$, which means that the vehicles don't have enough energy to move. Therefore, $W_{\text{off}} = D$. □

Property 2.3.3 *If $n \leq \hat{D}$, then $W_{off} \leq 2 \cdot \hat{D} + \ell \cdot n$.*

Proof : Since $n \leq \hat{D} \leq W_{\text{off}}$, the whole grid does not need to be partitioned into small cubes and any vehicle can walk to any other point in the grid. The traveling overhead is upper bounded by $\ell \cdot n$. Following the same reason as in Lemma 2.2.5, at most $2 \cdot \hat{D} + \ell \cdot n$ energy is needed for each vehicle. □

Using Corollary 2.2.7, together with Properties 2.3.1, 2.3.2, and 2.3.3, a $2(2 \cdot 3^{\ell} + \ell)$-approximation linear-time algorithm to compute W_{off} is given in Algorithm 1.

Analysis of Algorithm 1: Steps 1 to 5 will be visited only once and need time $O(n^{\ell})$; Steps 6 and 7 will be visited at most $\log_2 n$ times and need time $O(1)$ for each

visiting; Steps 8 and 9 will be visited at most $\frac{n^\ell}{2^\ell} + \frac{n^\ell}{4^\ell} + \frac{n^\ell}{8^\ell} + \cdots + 1 \leq \frac{n^\ell}{2^\ell - 1}$ times and need processing time $O(2^\ell)$ for each visiting; Steps 10 to 12 need processing time $O(\frac{n^\ell}{2^\ell} + \frac{n^\ell}{4^\ell} + \frac{n^\ell}{8^\ell} + \cdots + 1) = O(\frac{n^\ell}{2^\ell - 1})$. Therefore, the time complexity of Algorithm 1 is $O(n^\ell)$. In terms of memory requirement, each d_w is an ℓ-dimensional array of size $(\frac{n}{w})^\ell$.

Chapter 3

On-line case

In this chapter, we examine the on-line case of our Capacitated Multivehicle Routing Problem (CMVRP). In Section 3.1, we review the concept of diffusing computation, a technique used in our on-line algorithm. We give a decentralized on-line strategy for vehicles to serve jobs in Section 3.2. Finally, the characterization of optimal on-line performance (i.e. proof of Theorem 1.4.2) is provided in Section 3.3.

3.1 Diffusing Computations

In the seminal work [6], Dijkstra and Scholten first introduced the concept of diffusing computations in a distributed system of processes. A computation is *diffusing* when nodes receive information from another node (predecessor), and send it to all or a subset of their neighbors (successors). In such computations, a single active initial node awakens other nodes to perform some computation. These awoken nodes can spread the computation to other nodes, which then spread the computation further, and so on.

Dijkstra and Scholten [6] also suggested an elegant algorithm for detecting the termination of an arbitrary diffusing computation in any network. That is, the process starting the computation is informed when it is completed. In their algorithm, a spanning tree of active nodes is constructed by starting with a single active node (the root of the tree and initiator of the diffusing computation) that gradually broadcasts queries to other nodes in the network, awakening idle nodes as queries are passed

along. Upon activation, a node becomes the child of the activating node, causing a transition from idle to active status. The active node is always part of the tree. If a node that is already in the tree receives a request for activation, it notifies the sender that the tree's topology need not change. A node can be removed from the tree by replying its parent when it is an idle leaf node (when replies have been received for all queries sent). Termination is detected when the tree contains only one idle node — the root node.

The algorithm assumes very little about the underlying graph which represents the network. Thus, it is suitable for application to a number of problems arising in distributed programming.

3.2 On-line Strategy

Unlike the off-line case where the demand distribution is known to all vehicles a priori, the on-line problem requires communication between the vehicles. Before proceeding further, we first state our assumptions about the communication model and the messaging protocol:

- The time interval between any two successive job arrivals is long enough to finish any computation and movement.
- This is a decentralized model: there is no central controller for the system.
- A vehicle can communicate with other vehicles by sending/receiving messages.
- When two vehicles are within a constant distance[1] of each other, they are said to be neighbors. Neighbors can communicate directly with one another without having messages relayed through intermediate vehicles.
- The underlying communication topology is connected. That is, any two vehicles can communicate with each other by having messages go through some intermediate neighboring vehicles.

[1] This could be any arbitrary constant number. We use 2 here.

- The communication links are bidirectional.
- The communication cost is negligible: we assume communication requires no energy. If a vehicle uses up all its energy, it is still able to communicate with its neighbors and relay messages.
- A vehicle has only local knowledge: it knows the identities and positions of its neighbors; it is ignorant of the identities of all other vehicles and of the general structure of the network.
- Every vehicle has an input buffer of unbounded length. If vehicle P sends a message to a neighbor vehicle Q, then the message gets appended at the end of the input buffer of Q after a finite, arbitrary delay. *The assumption of unbounded length buffers is for ease of exposition. It shall be apparent from the strategy that the input buffer length of Q can be bounded by the number of neighbors of Q.*
- Error free communication: Messages are not lost or altered during transmission.
- Synchronous communication: Messages sent from P to Q arrive at Q's input buffer in the order sent.
- Two messages arriving simultaneously at an input buffer are ordered arbitrarily and appended to the buffer. A process receives a message by removing one from its input buffer.

We further assume there is enough energy to follow the strategy and process all the jobs. In Section 3.3, we determine this required amount of energy for each vehicle.

As in Lemma 2.2.5, we partition $\mathbb{Z}^\ell$ into $\underbrace{\lceil \omega_c \rceil \times \lceil \omega_c \rceil \times \cdots \times \lceil \omega_c \rceil}_{\ell}$ ℓ-cubes (ω_c is defined in Corollary 2.2.7). The vertices of each cube are colored black or white according to:

$$Color(x) = \begin{cases} black & \text{if } \sum x_i \equiv 0 (mod\ 2) \\ white & \text{otherwise} \end{cases} .$$

So, in 2 dimensions the cubes are colored like a chessboard.

If $\lceil \omega_c \rceil$ is even, the number of black vertices and the number of white vertices are equal. If $\lceil \omega_c \rceil$ is odd, we assume (without loss of generality) there is 1 more black vertex in any ℓ-cube (if not, switch the colors of vertices in the cube). In this manner, each ℓ-cube can be further divided into pairs: each pair consists of two adjacent vertices: one black and one white; for odd $\lceil \omega_c \rceil$, there may be a single black vertex left unpaired.

3.2.1 Vehicle State

We characterize the state of each vehicle by the pair: (S_1, S_2), where S_1 represents the working state of the vehicle and S_2 represents the message-transfer state of the vehicle.

The working state of the vehicle, S_1, can be one of the following:

- ***Idle***. Initially, all the vehicles at white vertices are *idle*. An *idle* vehicle does not serve any job but waits for a message to move to the specified vertex. After such move, the vehicle becomes *active*.

- ***Active***. Initially, all the vehicles at black vertices are *active*. An *active* vehicle will serve the jobs arriving at vertices that belong to the same pair as the vehicle belongs to. Because the two vertices belonging to the same pair are adjacent, the vehicle need walk at most distance 1.

- ***Done***. When an *active* vehicle uses up its energy, its working state becomes *done*.

The message-transfer state of the vehicle, S_2, can be one of the following:

- ***Initiator***. When an *active* vehicle becomes *done*, there is no vehicle to serve the jobs arriving at the vertices in the same pair. A replacing candidate — an *idle* vehicle in the same ℓ-cube — needs to be found and sent a message with the *done* vehicle's vertex position. We use a diffusing computation [6] to find the

candidate *idle* vehicle. (In a centralized system, the candidate *idle* vehicle could be easily found and messaged.) The *done* vehicle starts a new computation and is the *initiator* of the diffusing computation.

- ***Waiting***. Initially, all the vehicles are *waiting* for signals to partake in a diffusing computation. When the computation finishes, vehicles change back to the *waiting* state.

- ***Searching***. When a *waiting* vehicle receives a signal of a diffusing computation, its state changes to *searching*. When it has finished all its computation, its state returns to *waiting* again.

The state transition diagram is shown in Figure 3.1. Since an *active* or *idle* vehicle can not be the initiator of a diffusing computation, the states (*active*, *initiator*) and (*idle*, *initiator*) are not valid states.

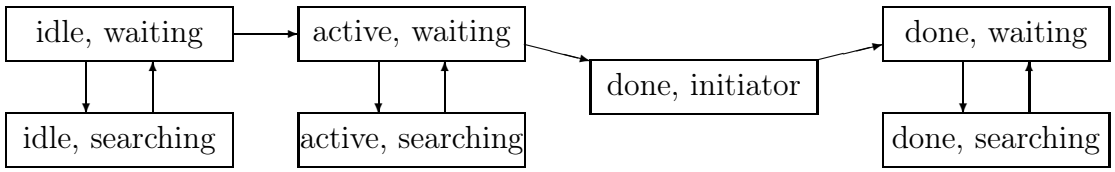


Figure 3.1: State Transition Diagram. States (*active*, *initiator*) and (*idle*, *initiator*) are not valid states.

3.2.2 The Overall Structure

Our scheme has two parts. The first part is the processing of the jobs. When a job arrives at a vertex, if there is an active vehicle at the vertex, then it serves the job; if not, there must be an active vehicle at the other vertex in the same pair and that vehicle will serve the job.

The second part is the diffusing computation to find a candidate *idle* vehicle to replace the *done* vehicle. The algorithm is based on the Dijkstra-Scholten algorithm [6] and works in two phases, both of which are initiated by the *done* vehicle. In phase I, the *done* vehicle initializes a diffusing computation. At the end of phase

I, the candidate will be found and the path from the initiator (*done* vehicle) to the candidate will have been identified. In phase II, a message will be transmitted along the path, which is identified in phase I, from the *done* vehicle to the candidate. Upon receiving the message, the candidate will move to the position of the *done* vehicle and become *active*.

In the remainder of the section, we focus on the algorithm for finding a candidate by diffusing computation and the algorithm for replacing the *done* vehicle.

3.2.3 Phase I Computation

3.2.3.1 Messages Used in Phase I

Phase I computation uses two kinds of messages:

- A ***query*** message associated with the pair $(init, p)$, where $init$ is the initiator of the computation and p is the identity of the vehicle sending the message. vehicle p sends a *query* message to all its neighbors in the same ℓ-cube. Intuitively, a *query* message signals that the vehicle is asking its neighbors if they are idle vehicles.

- A ***reply*** message to a *query* is a pair $(flag, p)$, where p is the identity of the vehicle sending the message and $flag$ is a boolean value. A vehicle p sends a *reply* message to vehicle q in response to a *query* message sent by q. Intuitively, a *reply* message with *flag=true* denotes that p has found an *idle* vehicle; a *reply* message with *flag=false* denotes that p has not found an *idle* vehicle.

3.2.3.2 Local Data Used By a Vehicle During Phase I

num This is the number of un-responded messages, that is, the number of messages sent by this vehicle for which no *reply* message has been received so far.

par The predecessor (parent) identity from which the first *query* message was received.

child The successor (child) identity from which the first *reply* message with *flag=true* was received.

init This is the initiator identity of the diffusing computation. Initially it is NULL. It is used to distinguish diffusing computations initiated by different vehicles and to avoid joining the same computation more than once. If we tag the diffusing computation with another sequence number k, we can also keep track of diffusing computations initiated at different times by the same vertex.

3.2.3.3 Phase I Algorithm Description

- When an *active* vehicle becomes *done*, its status changes from (*active*, *waiting*) to (*done*, *initiator*). It sends out a *query* message to all its neighbors. Its *par* is NULL. When it receives the first *reply* message with *flag=true*, it sets *child* to be the sender of the message. When all its messages have been responded, *i.e. num*=0, the vehicle changes its status to (*done*, *waiting*).

- When a *waiting* vehicle receives a *query* message with an *init* different from its current initiator identity, it records the sender of the query as its *par*, updates its *init* to be the *init* associated with the *query*. If it is an *idle* vehicle, send back a *reply* message with *flag=true*. If it is not an *idle* vehicle, change its status to *searching*, and send *query* messages to all its neighbors.

- When a non-*waiting* vehicle receives a *query* message, or a *waiting* vehicle receives a *query* message with an *init* same as its current initiator identity, send back a *reply* message with *flag=false* immediately.

- When a *searching* vehicle receives the first *reply* message with *flag=true*, set its *child* to be the sender of the message and send a *reply* message with *flag=true* to its *par*.

- When a *searching* vehicle has received *replies* from each of its neighbors, *i.e. num*=0, it changes status to *waiting*. If its *child* is NULL, it sends a *reply* with *flag=false* to its *par*.

When the *done* vehicle's status changes back to *waiting*, the diffusing computation has finished and a path to a candidate *idle* vehicle has been established. The details of the algorithm in Phase I are given in Algorithms 2.

3.2.4 Phase II Computation

Phase II employs only one kind of message: ***Move*** along with the destination associated with it. When the initiator of the diffusing computation changes its status to *waiting*, the computation has been finished. Initiator p sends out a *move*(location of p) message to its *child*, its *child* copies it to the next *child*, and the process continues until we reach an *idle* vehicle. The *idle* vehicle then moves to the location of p and changes its state to (*active*, *waiting*).

3.2.5 Discussion

In the previous sections, we have described a decentralized on-line strategy for vehicles to serve jobs. The strategy works under the condition that all the vehicles work well. But in practice, a *done* vehicle could fail to initialize a diffusing computation, an *active* vehicle could break down and become *dead* (it can't process jobs any more). We distinguish among these four different scenarios:

1. All the vehicles work well and no exception occurs.
2. There are *done* vehicles failing to initialize a diffusing computation, but no vehicles breaking down and becoming *dead*.
3. There are constant number of *active* vehicles breaking down and becoming *dead*.
4. There are large (more than constant) number of *active* vehicles breaking down and becoming *dead*.

The scenario 1 is the normal scenario and the protocol works well under this case.

When the second scenario occurs, an alternative initiator of the diffusing computation is needed, which could be one of the neighbors of the *done* vehicle. This

Algorithm 2 Phase I Algorithm.

When a vehicle p uses up its energy:

1. $p.s_1 \leftarrow$ *done*, $p.s_2 \leftarrow$ *initiator*;
2. $p.par \leftarrow$ NULL;
3. $p.init \leftarrow p$;
4. send $query(p, p)$ to all its neighbors;
5. $p.num \leftarrow$ the number of neighbors of p.

For a vehicle p upon receiving an query message (init, q):

1. If $p.s_2 = waiting$ & $p.init \neq init$
2. $p.par \leftarrow q$;
3. $p.init \leftarrow init$;
4. $p.child \leftarrow NULL$;
5. If $p.s_1 = idle$
6. send a *reply* message (*true*,*p*) to q;
7. else
8. $p.s_2 \leftarrow searching$;
9. send *query* messages $(init, p)$ to all its neighbors;
10. $p.num \leftarrow$ the number of neighbors of p;
11. else
12. send a *reply* message (*false*,*p*) to q.

For a vehicle p upon receiving an reply message (flag, q):

1. $p.num \leftarrow p.num$ - 1;
2. If *flag*=*true* and this is the first *reply* message with *flag*=*true* it receives
3. p.$child \leftarrow q$;
4. send a *reply* message (*true*, p) to $p.par$;
5. If $p.num$=0
6. $p.s_2 \leftarrow waiting$;
7. If $p.child$=NULL
8. send a *reply* message (*false*, p) to $p.par$.

could be easily implemented by adding a pointer, "*monitoring*," to each *active* vehicle, which points to one of its active neighbors. Any *active* vehicle is pointed by a unique "*monitoring*" pointer. All the pointers form a loop.

A vehicle sends out "*existing*" messages periodically to notify its neighbors that it is still there. If a vehicle has not received the "*existing*" messages from the neighbor its "*monitoring*" pointer pointing to for a certain time, it decides that the neighbor is *done* and initializes a diffusing computation for the neighbor. The replacing vehicle for the *done* vehicle needs to update its "*monitoring*" pointer to maintain the loop.

When the misfortune of scenario 3 occurs, since there are only constant number of *active* vehicles breaking down and the overall energy requirement changes very slightly, the energy constraint is not a concern. We can use the same way used in the second scenario to initialize a diffusing computation.

The scenario 4 is different from the other scenarios, which will be discussed in Chapter 4.

3.3 Proof of Theorem 1.4.2

Lemma 3.3.1 *$W_{on} \leq (4 \cdot 3^\ell + \ell) \cdot \omega_c = (4 \cdot 3^\ell + \ell) \cdot \max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$, where ω_T denotes the solution of the equation $\omega_T \cdot |N_{\omega_T}(T)| = \sum_{x\in T} d(x)$.*

Proof : In order to satisfy the requirement that upon a job arrival, there exists an *active* vehicle in each black-white pair, we need to make sure that after all jobs have arrived, there still exists an *active* vehicle in each black-white pair. According to Corollary 2.2.7, the amount of energy needed in any $\underbrace{\lceil\omega_c\rceil \times \lceil\omega_c\rceil \times \cdots \times \lceil\omega_c\rceil}_{\ell}$ ℓ-cube is at most $\omega_c(3\lceil\omega_c\rceil)^\ell$, excluding the travel overhead. Since the processing of each job requires a walk of distance at most 1, at most $2 \cdot \omega_c(3\lceil\omega_c\rceil)^\ell$ units of energy is needed in each cube for job processing. If each vehicle is given $4 \cdot 3^\ell \cdot \omega_c$ units of energy for processing jobs, then after all jobs have arrived and been processed, there are at least

$$\frac{4 \cdot 3^\ell \cdot \omega_c \cdot \lceil\omega_c\rceil^\ell - 2 \cdot \omega_c(3\lceil\omega_c\rceil)^\ell}{4 \cdot 3^\ell \cdot \omega_c} = \frac{\lceil\omega_c\rceil^\ell}{2}$$

vehicles with energy left, which means there exists an active vehicle in each black-white pair. The distance between any two vertices in the same cube in at most $\ell \cdot \omega_c$, which implies that an *idle* vehicle needs to move a distance at most $\ell \cdot \omega_c$ to become *active*. Therefore, at most $(4 \cdot 3^\ell + \ell) \cdot \omega_c$ units of energy is needed for each vehicle for the on-line case. Since $\omega_c \leq \max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$ (see in the proof of Corollary 2.2.7), the lemma is proven. □

Clearly, $W_{\text{on}} \geq W_{\text{off}} \geq \max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$, where ω_T denotes the solution of $\omega_T \cdot |N_{\omega_T}(T)| = \sum_{x\in T} d(x)$. Combining this with Lemma 3.3.1, we have proven Theorem 1.4.2.

Chapter 4

Different Case Study: Broken Vehicles

In Chapter 2, we discussed the problem of minimizing the initial energy needed for each vehicle in the off-line case and showed that W_{off} is of the same order as the solution to program (2.8), which is

$$W_{\text{off}} = \Theta\left(\max_{T:T\subseteq\mathbb{Z}^{\ell}} \omega_T\right),$$

where ω_T is the solution of $\omega_T \cdot |N_{\omega_T}(T)| = \sum_{x\in T} d(x)$. Furthermore, we showed that W_{on} is of the same order as W_{off} in Chapter 3.

We now extend the problem to allow a large number of vehicles breaking down, which is the fourth scenario mentioned in Section 3.2.5.

For each vehicle i there is a "longevity" parameter $p_i (0 \leq p_i \leq 1)$. The vehicle i breaks at the time when a fraction p_i of its initial energy has been used. If $p_i = 0$, the vehicle breaks initially; if $p_i = 1$, the vehicle will not break.

In the off-line case, the demand function $d(\cdot)$, arrival sequence, and the "longevity" parameters are known at the beginning. Let $W_{\text{off-b}}$ denote the minimal W needed in the off-line case when a large number of vehicles breaking down is allowed.

Using Linear Programming, we get a lower bound on $W_{\text{off-b}}$ in Section 4.1. An example is given in Section 4.2. It turns out that $W_{\text{off-b}}$ is not of the same order as the lower bound gotten in Section 4.1. This is different from the case in Chapter 2.

4.1 Lower Bound on $W_{\text{off-b}}$

In this section, we use the same method as in Section 2.2 to get a lower bound on $W_{\text{off-b}}$.

Theorem 4.1.1 *$W_{\text{off-b}} \geq \max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$, where ω_T is the solution of $\omega_T \cdot \sum_{i\in N_{p_i\cdot\omega_T}(T)} p_i = \sum_{i\in T} d(i)$.* □

Proof : Straightforwardly, the solution to the following programming (4.1) is a lower bound on $W_{\text{off-b}}$.

$$\begin{aligned} &\text{minimize} \quad \omega \\ &\text{s.t.} \quad \begin{cases} \sum_{j\in N_{p_i\cdot\omega}(i)} f_{ij} \leq p_i\cdot\omega, & \forall\ i\in\mathbb{Z}^\ell \\ \sum_{i\in N_{p_i\cdot\omega}(j)} f_{ij} \geq d(j), & \forall\ j\in\mathbb{Z}^\ell \\ f_{ij}\geq 0, & \forall\ i,j\in\mathbb{Z}^\ell\ \&\ \|i-j\|\leq p_i\cdot\omega, \end{cases} \end{aligned} \tag{4.1}$$

where $F=\{f_{ij}\}$ denotes the set of "flows." Each flow f_{ij} represents the amount of energy transported by the vehicle at position i to position j.

Since we are using the same method as in Section 2.2 to solve (4.1), we will just give an outline of the proof in the following.

We first study the following linear program (LP):

$$\begin{aligned} &\text{minimize} \quad \omega \\ &\text{s.t.} \quad \begin{cases} \sum_{j\in N_{p_i\cdot r}(i)} f_{ij} \leq p_i\cdot\omega, & \forall\ i\in\mathbb{Z}^\ell \\ \sum_{i\in N_{p_i\cdot r}(j)} f_{ij} \geq d(j), & \forall\ j\in\mathbb{Z}^\ell \\ f_{ij}\geq 0, & \forall\ i,j\in\mathbb{Z}^\ell\ \text{ and }\ \|i-j\|\leq p_i\cdot r \end{cases} \end{aligned} \tag{4.2}$$

The dual of (4.2) is

$$\begin{aligned} \text{maximize} \quad & \sum_{j\in\mathbb{Z}^\ell} \beta_j d(j) \\ \text{s.t.} \quad & \begin{cases} \sum\limits_{i\in\mathbb{Z}^\ell} p_i\cdot\alpha_i \le 1, & \\ \alpha_i \ge 0, & \forall\ i\in\mathbb{Z}^\ell \\ \beta_j \ge 0, & \forall\ j\in\mathbb{Z}^\ell \\ \beta_j \le \alpha_i, & \forall\ i,j\in\mathbb{Z}^\ell \ \text{ and } \ \|i-j\|\le p_i\cdot r \end{cases} \end{aligned} \tag{4.3}$$

which simplifies to the LP:

$$\begin{aligned} \text{maximize} \quad & \sum_{j\in\mathbb{Z}^\ell} \left(d(j) \times \min_{i:\|i-j\|\le p_i\cdot r} \alpha_i \right) \\ \text{s.t.} \quad & \begin{cases} \sum\limits_{i\in\mathbb{Z}^\ell} p_i\cdot\alpha_i \le 1, & \\ \alpha_i \ge 0, & \forall\ i\in\mathbb{Z}^\ell \end{cases} \end{aligned} \tag{4.4}$$

Let h denote a mapping from the set of subsets of $\mathbb{Z}^\ell$ to $\mathbb{R}$, then (4.4) is equivalent to

$$\begin{aligned} \text{maximize} \quad & \sum_{j} \left(d(j) \sum_{T:\{i:\|i-j\|\le p_i\cdot r\}\subseteq T} h(T) \right) \\ \text{s.t.} \quad & \begin{cases} \sum\limits_{T\subseteq\mathbb{Z}^\ell} \left(h(T) \sum\limits_{i\in T} p_i \right) \le 1 \\ h(T)\ge 0,\ \forall\ T\subseteq\mathbb{Z}^\ell \end{cases} \end{aligned} \tag{4.5}$$

The dual of (4.5) is

$$\begin{aligned} \text{minimize} \quad & \omega \\ \text{s.t.} \quad & \omega\cdot\sum_{i\in T} p_i \ge \sum_{j:\{i:\|i-j\|\le p_i\cdot r\}\subseteq T} d(j),\ \forall\ T\subseteq\mathbb{Z}^\ell \end{aligned} \tag{4.6}$$

Therefore, the solution of (4.2) is

$$\omega = \max_{T:T\subseteq\mathbb{Z}^\ell} \frac{\sum\limits_{j:\{i:\|i-j\|\le p_i\cdot r\}\subseteq T} d(j)}{\sum\limits_{i\in T} p_i} = \max_{T:T\subseteq\mathbb{Z}^\ell} \frac{\sum\limits_{i\in T} d(i)}{\sum\limits_{i\in N_{p_i\cdot r}(T)} p_i}.$$

Let $\omega(r) = \max_{T:T\subseteq\mathbb{Z}^\ell} \frac{\sum_{i\in T} d(i)}{\sum_{i\in N_{p_i\cdot r}(T)} p_i}$. We see that $\omega(r)$ is a non-increasing function of r. Therefore, the solution to the program (4.1) is the same as the solution of the equation $r = \omega(r)$, which is $\max_{T:T\subseteq\mathbb{Z}^\ell} \omega_T$. □

4.2 An Example of a Large $W_{\text{off-b}}$

In this section, we use an example to illustrate the difference between $W_{\text{off-b}}$ and the solution to (4.1).

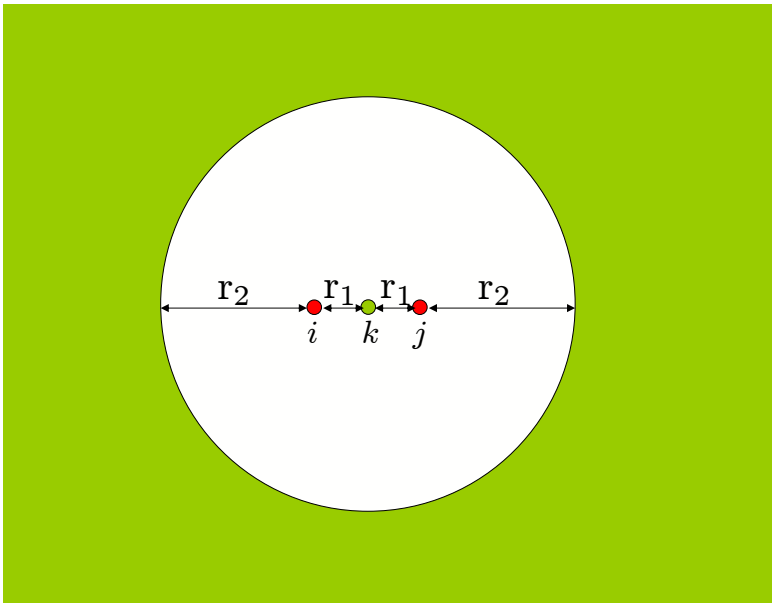


Figure 4.1: A scenario illustration. Every vehicle x outside the circle (green position) has $p_x = 1$; the vehicle at k (green point in the circle) has $p_k = 1$; any other vehicle x has $p_x = 0$. The demands at i and j (red points) are $d(i) = d(j) = r_1$; the demand at any other position is 0.

Consider the scenario shown in Figure 4.1. The distance between i and k is $D(i,k) = r_1$. The distance between j and k is also $D(j,k) = r_1$. $D(i,j) = 2r_1$. The distance from i or j to the boundary of the circle is $r_2 \gg r_1$. Every vehicle x outside the circle has $p_x = 1$; the vehicle at the green position k in the circle has $p_k = 1$; other vehicles have $p_x = 0$. The demands at i and j (red points) are $d(i) = d(j) = r_1$; the demand at any other position is 0. The service requests arrive at positions i and j alternatively, i.e., the sequence of requests is $i, j, i, j, \cdots, i, j$.

Under the scenario of Figure 4.1, the solution to programming (4.1) is $2r_1$, which is achieved when $f_{ki} = r_1, f_{kj} = r_1$.

Assume $W_{\text{off-b}}$ is of order r_1. Since $r_2 \gg r_1$, the non-broken vehicles outside the circle can't move to positions i or j. The only remaining non-broken vehicle is k. The vehicle k needs to walk back and forth between i and j because the service requests arrive at positions i and j alternatively. In order to serve all the requests, the travel distance of k is $r_1 + (2r_1 - 1) \cdot 2r_1$, which is not of order r_1. A contradiction. Therefore, $W_{\text{off-b}}$ is not of order r_1 and $W_{\text{off-b}} = \omega(r_1)$.

From this example, we see that, when the vehicle breaking down is allowed, the job arriving order matters and the bound given in Section 4.1 is a weak lower bound, which is not tight up to a constant factor.

Chapter 5

Inter-Vehicle Energy Transfers

In Chapters 2 and 3, we discussed the problem of minimizing the initial energy needed for each vehicle in the off-line and on-line cases respectively. We now extend the problem to allow inter-vehicle energy transfers. That is, vehicle $\mathcal{A}$ can transfer energy to vehicle $\mathcal{B}$ when $\mathcal{A}$ and $\mathcal{B}$ are at the same location. We consider two methods of accounting for energy transfers:

- Fixed cost: Charge a_1 units of energy per transfer, no matter how much energy is transferred, or
- Variable cost: Charge $a_2 \ll 1$ units of energy per unit of energy transferred.

It turns out that, no matter which accounting method is used, under the same job load, the minimal vehicle capacity W needed with energy transfer allowed is of the same order as that without energy transfer. We let $W_{\text{trans-off}}$ and $W_{\text{trans-on}}$ denote the minimal W in the off-line and on-line energy-transfer cases respectively. Clearly,

$$W_{\text{trans-off}} \leq W_{\text{trans-on}}, \quad W_{\text{trans-off}} \leq W_{\text{off}}, \quad \text{and} \quad W_{\text{trans-on}} \leq W_{\text{on}},$$

so it is enough to show that $W_{\text{trans-off}} = \Theta(W_{\text{off}})$. This, together with Theorem 1.4.2, shows that all four W's ($W_{\text{trans-off}}$, $W_{\text{trans-on}}$, W_{off}, and W_{on}) are of the same order. In the following, we will first show that $W_{\text{trans-off}} = \Theta(W_{\text{off}})$. Then we provide a brief discussion on how high capacity tanks can help performance.

5.1 $W_{\text{trans-off}} = \Theta(W_{\text{off}})$

In this section, the analysis in the proof of Theorem 5.1.1 gives a lower bound on $W_{\text{trans-off}}$. It works for both energy-transfer accounting methods: the fixed cost scenario where each transfer costs a_1 units regardless of amount transferred, and the variable cost scenario where the transfer costs a_2 units per unit of energy transferred. Once again, for simplicity, we only study the two-dimensional ($\ell = 2$) case.

Theorem 5.1.1 *$W_{\text{trans-off}} = \Theta(W_{\text{off}})$.*

Proof : Since there is at most $W_{\text{trans-off}}$ units of energy in each vehicle, the minimum traveling cost per unit energy per distance is $\frac{1}{W_{\text{trans-off}}}$[1]. Moving $W_{\text{trans-off}}$ units of energy from a position i to another position j will cost energy at least

$$\begin{aligned} & W_{\text{trans-off}} \times \frac{1}{W_{\text{trans-off}}} + W_{\text{trans-off}} \left(1 - \frac{1}{W_{\text{trans-off}}}\right) \times \frac{1}{W_{\text{trans-off}}} \\ & + \cdots + W_{\text{trans-off}} \left(1 - \frac{1}{W_{\text{trans-off}}}\right)^{\|i-j\|-1} \times \frac{1}{W_{\text{trans-off}}} \\ = \; & W_{\text{trans-off}} \times \left(1 - \left(1 - \frac{1}{W_{\text{trans-off}}}\right)^{\|i-j\|}\right), \end{aligned}$$

and so the left energy amount is at most $W_{\text{trans-off}} \times \left(1 - \frac{1}{W_{\text{trans-off}}}\right)^{\|i-j\|}$, no matter how many transfers of energy occur, nor which transfer accounting method is used.

Therefore, when moving $W_{\text{trans-off}}$ units of energy from position i into an $s \times s$ square $T \subseteq \mathbb{Z}^2$, the amount of remaining energy, which is initially from i, is at most

$$W_{\text{trans-off}} \times \left(1 - \frac{1}{W_{\text{trans-off}}}\right)^{D(i,T)},$$

where $D(i,T)$ is the Manhattan distance from the point i to the square T.

The amount of energy initially in the square T is $W_{\text{trans-off}} \times s^2$. Since

$$|\{i : D(i,T) = r\}| = 4s + 4(r-1),$$

[1]The traveling cost is 1 unit of energy per unit of distance.

the total amount of energy that can be moved into the square T is at most

$$\begin{aligned}
& W_{\text{trans-off}} \times s^2 + \sum_{r=1}^{\infty} W_{\text{trans-off}} \times (1 - \frac{1}{W_{\text{trans-off}}})^r \times (4s + 4(r-1)) \\
= \; & W_{\text{trans-off}} \times \left(s^2 + 4 \sum_{r=1}^{\infty} (1 - \frac{1}{W_{\text{trans-off}}})^r \times (r + s - 1) \right) \\
= \; & W_{\text{trans-off}} \times \left(s^2 + 4 \sum_{r=1}^{\infty} r(1 - \frac{1}{W_{\text{trans-off}}})^r + 4(s-1) \sum_{r=1}^{\infty} (1 - \frac{1}{W_{\text{trans-off}}})^r \right) \\
= \; & W_{\text{trans-off}} \times \left(s^2 + 4(W_{\text{trans-off}} - 1) W_{\text{trans-off}} + 4(s-1)(W_{\text{trans-off}} - 1) \right) \\
= \; & W_{\text{trans-off}} \times \left(s^2 + 4W_{\text{trans-off}}^2 + 4sW_{\text{trans-off}} - 8W_{\text{trans-off}} - 4s + 4 \right),
\end{aligned}$$

which should be at least the total demand in T. That is,

$$W_{\text{trans-off}} \times \left(s^2 + 4W_{\text{trans-off}}^2 + 4sW_{\text{trans-off}} - 8W_{\text{trans-off}} - 4s + 4 \right) \geq \sum_{i \in T} d(i).$$

Since

$$\begin{aligned}
& |N_{W_{\text{trans-off}}}(T)| \\
= \; & \Theta \left(s^2 + W_{\text{trans-off}}^2 \right) \\
= \; & \Theta \left(s^2 + 4W_{\text{trans-off}}^2 + 4sW_{\text{trans-off}} - 8W_{\text{trans-off}} - 4s + 4 \right),
\end{aligned}$$

we have that $W_{\text{trans-off}} = \Omega(\omega_T)$, where ω_T denotes the solution of $\omega_T \cdot |N_{\omega_T}(T)| = \sum_{x \in T} d(x)$. Let Γ be the set of all squares in $\mathbb{Z}^2$. Then $W_{\text{trans-off}} = \Omega \left(\max_{T \in \Gamma} \omega_T \right)$.

By Corollary 2.2.6, $W_{\text{off}} = \Theta \left(\max_{T \in \Gamma} \omega_T \right)$. Since $W_{\text{trans-off}} \leq W_{\text{off}}$, we have that $W_{\text{trans-off}} = \Theta \left(\max_{T \in \Gamma} \omega_T \right) = \Theta(W_{\text{off}})$. □

Corollary 5.1.2 *$W_{\text{trans-off}}$, $W_{\text{trans-on}}$, W_{off}, and W_{on} are $\Theta \left(\max_T \omega_T \right)$.*

5.2 High Capacity Tanks

Often in practical applications, the vehicle tanks might not be full initially. Let W denote the initial energy and $C > W$ denote the vehicle tank capacity. When energy transfer is not allowed, the minimal W is the same as discussed in Chapters 2 and 3 regardless of C. However, when energy transfer is allowed, large capacity tanks can improve performance. We present an interesting example where $W_{\text{trans}-\text{off}} = \Theta\left(\text{avg}_{\text{x}}\text{d(x)}\right)$.

5.2.1 An example of $W_{\text{trans}-\text{off}}$ with $C = \infty$

For this illustration, one dimension suffices. Consider a line segment of length N (vertices numbered 1 through N). When the vehicle capacities are infinite, we let vehicle-1 travel to N while collecting energy from vehicles $2, 3, \ldots, N-1$ along the way. At N, it exchanges energy with vehicle N such that vehicle-N has exactly the amount of energy required to process the jobs at its position. After the exchange, vehicle-1 walks back to its original position while distributing energy to vehicles $N-1, \ldots, 3, 2$ according to the demand required at each position. In total, the number of energy transfers is $2N - 3$ and the distance traveled is $2N - 2$.

- In the fixed cost scenario where each transfer costs a_1 units regardless of amount transferred, the total energy needed is

$$E_{total} = a_1 \cdot (2N - 3) + (2N - 2) + \sum_x d(x).$$

  Then

$$\begin{aligned} W_{\text{trans}-\text{off}} &= \frac{a_1 \cdot (2N - 3) + (2N - 2) + \sum_x d(x)}{N} \\ &= 2a_1 + 2 + \frac{\sum_x d(x) - 3a_1 - 2}{N}. \end{aligned}$$

- In the variable cost scenario where the transfer costs a_2 units per unit of energy

transferred, the total energy needed is

$$E_{total} = a_2 \cdot W_{\text{trans-off}} \cdot (2N-3) + (2N-2) + \sum_x d(x).$$

Then

$$\begin{aligned} & W_{\text{trans-off}} \cdot N = a_2 \cdot W_{\text{trans-off}} \cdot (2N-3) + (2N-2) + \sum_x d(x) \\ \Rightarrow \quad & W_{\text{trans-off}} = \frac{2N - 2 + \sum_x d(x)}{N - 2a_2 N + 3a_2}. \end{aligned}$$

Therefore, $W_{\text{trans-off}} = \Theta\left(\sum_x d(x)/N\right) = \Theta\left(\text{avg}_\text{x}\text{d(x)}\right)$, no matter which energy-transfer accounting method is used.

From this example, we see that there is significant performance difference between the $C = W$ and $C \neq W$ in the off-line case.

Chapter 6

Conclusions and Future Works

The key concepts/contributions of the thesis is as follows:

- The minimal energy needed in the off-line case is of the same order as that of the on-line case.
- We proposed an algorithm and a strategy for the Capacitated Multivehicle Routing Problem.

We note that the proposed algorithm represents only an initial prototype. We expect that this thesis will provide an incentive for future work on this problem. There are also several immediate research directions that can be pursued:

- The example from Section 5.2.1 shows that when inter-vehicle energy transfer is allowed and vehicles posses (non-full) large-capacity tanks, there can be significant energy savings. The questions of how much energy could be saved in general remains open.
- One can explore further the precise relationship between W_{off} and W_{on}, trying to tighten the constant factor, which is exponential in ℓ. In particular, it would be nice to show that the exponential dependence on ℓ is unnecessary.
- We have only discussed the case where the underlining graph is a grid. It would be nice to have results for graphs in general.

Ongoing advances in technology suggests that electronic devices will become smaller, smarter, more mobile, and ubiquitous. We hope to see the applications of these devices benefit from the theoretical research of CMVRP. The rich applications of such mobile devices will surely yield many interesting theoretical research problems as well.

Bibliography

[1] L. Bodin and B. Golden. Classification in vehicle routing and scheduling. *Networks*, 11(2):97–108, 1981.

[2] J. Bramel and D. Simchi-Levi. Probabilistic analyses and practical algorithms for the vehicle routing problem with time windows. *Operations Research*, 44(3):501–509, 1996.

[3] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[4] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[5] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[6] E. W. Dijkstra and C. S. Scholten. Termination detection for diffusing computations. *Inf. Proc. Letters*, 11(1):1–4, 1980.

[7] M. L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42(4):626–642, 1994.

[8] M. Gendreau, F. Guertin, J. Y. Potvin, and E. Taillard. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390, 1999.

[9] B. E. Gillet and L. R. Miller. A heuristic algorithm for the vehicle - dispatch problem. *Operations Research*, 22(2):340–349, 1974.

[10] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". In *International Conference on Mobile Computing and Networking (MOBICOM)*, pages 271–278, 1999.

[11] N. Katoh and T. Yano. An approximation algorithm for the pickup and delivery vehicle routing problem on trees. *Discrete Appl. Math.*, 154(16):2335–2349, 2006.

[12] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European J. Oper. Res.*, 59:345–358, 1992.

[13] T. Ralphs, J. Hartman, and M. Galati. Capacitated vehicle routing and some related problems. *Some CVRP Slides, Rutgers University*, 2001.

[14] T. Ralphs, L. Kopman, W. Pulleyblank, and L. Trotter. On the capacitated vehicle routing problem. *Accepted to Mathematical Programming*, 2001.

[15] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, 2000.

[16] M. M. Solomon. Algorithms for the vehicle routing problem with time windows. *Transportation Science*, 29(2):156–166, 1995.

[17] P. Toth and D. Vigo. An overview of vehicle routing problems. pages 1–26, 2001.

[18] B. Warneke, M. Last, B. Liebowitz, and K. S. J. Pister. Smart dust: Communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51, 2001.

[19] A. Wren. Computers in transport planning and operation. *Operational Research Quarterly*, 23(3):404–405, 1972.

[20] A. Wren and A. Holliday. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly*, 23(3):333–344, 1972.

[21] K. Zhu, K. Tan, and L. Lee. Heuristics for vehicle routing problem with time windows. In *Sixth International Symposium on Artificial Intelligence and Mathematics*, 2000.