Universidad Autónoma
Metropolitana

UNIDAD CUAJIMALPA

GREEN-VRP

Proyecto Terminal

QUE PRESENTA:
NOMBRE DEL ALUMNO

Licenciatura en ...
Departamento de Matemáticas Aplicadas e Ingeniería

División de Ciencias Naturales e Ingeniería

Asesor y Responsable de la tesis:
NOMBRE DEL ASESOR(ES)

Mes y año (de finalización)

Índice general

Índice de figuras

Capítulo 1

Resumen

Capítulo 2

Introducción

Capítulo 3

Conocimientos preliminares

3.1. Problema de Optimización

La optimización es una rama de las matemáticas aplicadas que se enfoca en el desarrollo de principios y métodos para resolver problemas cuantitativos en diversas disciplinas como la física, biología, ingeniería o economía, así como en cualquier campo donde sea necesaria la toma de decisiones dentro de un conjunto de opciones, con el objetivo de encontrar la mejor o una de las mejores soluciones de manera eficiente.

En términos formales, consideramos una función $f : S \to \mathbb{R}$, donde $S \subseteq \mathbb{R}^n$. Denominaremos a f como función objetivo, y a S lo llamaremos conjunto factible o conjunto de soluciones posibles.

3.2. Tipos de Optimización

Dentro de la optimización, existen dos tipos principales: la optimización discreta y la optimización continua.

La optimización discreta se aplica cuando el dominio S de la función objetivo es un conjunto discreto. En este contexto, el conjunto de soluciones posibles es finito o numerablemente infinito. La optimización discreta a menudo involucra la búsqueda de soluciones en combinaciones específicas y puede involucrar problemas como la programación lineal entera o los problemas de asignación. Las soluciones óptimas pueden ser difíciles de encontrar debido a la naturaleza combinatoria del problema.

Por otro lado, la optimización continua se refiere a situaciones en las que el conjunto S de la función objetivo es un conjunto continuo. En este caso, el dominio es un intervalo o un subconjunto de $\mathbb{R}^n$ que no es discreto. Aquí se buscan soluciones que maximizan o minimizan la función objetivo sobre un espacio continuo, y los métodos comunes incluyen la programación lineal, la programación no lineal y el cálculo de variaciones. La optimización continua suele implicar el uso de técnicas de cálculo y análisis matemático.

3.3. Heurística y Metaheurística

La palabra "heurística" proviene del término griego "euriskein", que significa "encontrar". En el contexto de la optimización, se refiere a técnicas diseñadas para mejorar o resolver problemas que, de otra manera, no tendrían una solución eficiente. Los algoritmos heurísticos ayudan a encontrar soluciones aproximadas para problemas cuyas soluciones exactas no son factibles en tiempos polinomiales. Muchos algoritmos de inteligencia artificial son heurísticos o se basan en sus principios para resolver problemas.

El prefijo "meta" otorga un sentido de mayor nivel a la palabra heurística, lo que permite abordar problemas de mayor complejidad mediante soluciones factibles.

Tanto las heurísticas como las metaheurísticas se pueden utilizar de manera intercambiable para resolver problemas de optimización combinatoria.

3.4. Complejidad P, NP, NP-completo y NP-difÃcil

3.4.1. IntroducciÃ³*nalaComplejidadAlgorÃtmica*

En computaciÃ³*n, unalgoritmoesunprocedimientoqueresuelveunproblemapasoapaso, generandounasa*
odeentrada, locualnosayudaaevaluarsueficiencia.

Los algoritmos se clasifican en distintas clases de complejidad dependiendo del algoritmo mÃ¡s eficiente conocido para resolverlos: P, NP, NP-completo y NP-difÃcil. La clase P contiene problemas que pueden ser resueltos de forma eficiente, mientras que NP abarca aquellos cuya soluciÃ³*npuedeverificarserÃ!'pidamente.LasclasesNP*
completoyNP−difÃcilincluyenproblemasdemayorcomplejidad; resolverunodeestosdemaneraeficien
alaresoluciÃ³neficientedecualquierproblemaenNP.EstonosllevaalproblemafundamentalenteorÃ-
adelacomputaciÃ³n : PvsNP.

3.4.2. Complejidad P

En la teorÃa de la complejidad, nos enfocamos principalmente en problemas de decisiÃ³*n, aquellosdondelarespuestaesunsimple" sÃ" o" no" .LaclasePestÃ!'compuestaporproblemas*
on, *puederesolverloenuntiempoquecrecealosumocomounpolinomiode*n.

Ejemplo: Algoritmo de Ordenamiento por SelecciÃ³*n*

El algoritmo de ordenamiento por selecciÃ³*nesunmÃ©todosimpleperoineficienteparaordenarunalista*
sticas.

Ejemplo Paso a Paso Consideremos la lista (3 1 6 8 2). El proceso de ordenamiento serÃa:

1. (3 1 6 8 2) - Estado inicial
2. (1 3 6 8 2) - Se intercambia 1 (mÃnimo) con 3
3. (1 2 6 8 3) - Se intercambia 2 con 3

4. (1 2 3 8 6) - Se intercambia 3 con 6
5. (1 2 3 6 8) - Se intercambia 6 con 8
6. (1 2 3 6 8) - La lista estÃ¡ ordenada

AnÃ¡lisis de Complejidad NÃºmero de Comparaciones: Para una lista de n elementos:

- Primera pasada: $n-1$ comparaciones
- Segunda pasada: $n-2$ comparaciones
- $\vdots$

- Ãšltima pasada: 1 comparaciÃ³n Total de comparaciones: $(n-1)+(n-2)+\cdots+2+1=\frac{n(n-1)}{2}$

El algoritmo de ordenamiento por selecciÃ³$n, aunquesimpledeentendereimplementar, noesefic$
$asdondelasimplicidaddelalgoritmopuedesermÃ!`simportantequesueficiencia.$

3.4.3. Clase de complejidad NP

La clase NP abarca problemas donde una soluciÃ³$nafirmativa("sÃ")puedeserverificadaentie$

Un ejemplo clÃ¡sico es el problema de la suma de subconjuntos: dado un conjunto de n nÃºmeros, se busca si existe un subconjunto cuya suma sea S. Para verificar una soluciÃ³$nafirmativa, bastarÃaconquenosdieranelsubconjuntoencuestiÃ^3n, yent$
$amossumarsuselementosparacomprobarquelasumaescorrecta.$

Al igual que en P, los problemas en NP pueden incluir aquellos que ya mencionamos, pero con una importante distinciÃ³$n: aunquepodemosverificarsolucionesentiempopol$
$adelacomplejidadessitodoslosproblemasenNPpuedensertambiÃ©nresueltosentiempopolino$

Ejemplo de problema NP: El Problema del Agente Viajero

El Problema del Agente Viajero es uno de los ejemplos mÃ¡s claros para entender la complejidad NP, dado que es un problema que no se resuelve en tiempo polinomial, buscando encontrar una trayectoria que visita todas las n ciudades exactamente una vez, sin repetir ninguna.

Datos del problema:

- Un entero $n > 0$, que representa el nÃºmero de ciudades.
- Una matriz de distancias (d_{ij}) de dimensiÃ³nn $\times n$, donde d_{ij} es un entero mayor o igual a cero que representa la distancia entre la ciudad i y la ciudad j.

RepresentaciÃ³$nmatem$Ã!'$tica$: Una permutaciÃ³$n\pi$ cÃclica representa un recorrido, donde $\pi(j)$ se interpreta como la ciudad despuÃ©s de la ciudad j, para $j = 1, 2, \ldots, n$.

Costo del recorrido: El costo del recorrido se calcula mediante la siguiente fÃ³$rmula$:

$$\sum_{j=1}^{n} d_{j,\pi(j)}$$

Donde:

- j recorre todas las ciudades de 1 a n
- $d_{j,\pi(j)}$ representa la distancia entre la ciudad j y la ciudad que le sigue en el recorrido segÃºn la permutaciÃ³$n\pi$

CaracterÃsticas importantes:

1. La matriz de distancias es simÃ©trica, es decir, $d_{ij} = d_{ji}$ para todo i, j.
2. Las distancias son no negativas: $d_{ij} \geq 0$ para todo i, j.
3. La diagonal de la matriz de distancias (d_{ii}) generalmente se define como 0, ya que representa la distancia de una ciudad a sÃ misma.

Complejidad: El Problema del Agente Viajero pertenece a la clase de problemas NP-difÃcil, lo que significa que no se conoce un algoritmo eficiente (de tiempo polinomial) para resolverlo de manera Ã³*ptimaparatodosloscasos.*

Esta formulaciÃ³*nmatemÃ!'ticaproporcionaunabasesÃ³lidaparaelanÃ!'lisisydesarrollodealg*

3.4.4. Reducciones y las clases NP-completo y NP-difÃcil

Para profundizar en las clases de NP-completo y NP-difÃcil, es esencial introducir el concepto de reducciÃ³*n.UnareducciÃ³nentreproblemassignificaquesipodemosresolve*

DefiniciÃ³*ndeReducciÃ³nPolinomial*

Dado un problema A y un problema B, diremos que existe una reducciÃ³*npolinomialdeAaBsici*

Esto nos lleva a la clase de complejidad NP-difÃcil, que estÃ¡ compuesta por problemas a los cuales cualquier problema en NP puede ser reducido. NP-completo, por otro lado, incluye aquellos problemas que son tanto NP-difÃciles como pertenecientes a NP. Si alguna vez encontrÃ¡ramos un algoritmo eficiente para resolver un problema NP-completo, entonces podrÃamos resolver todos los problemas en NP de manera eficiente.

3.4.5. ConclusiÃ³*n*

Las clases de complejidad P, NP, NP-completo y NP-difÃcil son fundamentales para comprender la eficiencia de los algoritmos. El problema abierto mÃ¡s importante de la teorÃa de la computaciÃ³*n, PvsNP, siguesiendounapreguntasinrespuesta :*
Â?'PodemosresolvertodoslosproblemasquepodemosverificareficientementeResolverestapr
aimplicacionesprofundasnosoloenlateorÃa, sinoenmuchasÃ!'reasprÃ!'cticasdelacomputaciÃ

Capítulo 4

Desarrollo del proyecto

Capítulo 5

Conclusiones y trabajo futuro

Bibliografía