Techniki optymalizacji: opis zadania

Przemysław Wesołek

2007-11-08

Contents

1 Opis problemu

Celem zadania jest zaplanowania realizacji wszystkich zleceń opisanych w problemie, mając do użycia nieograniczoną flotę pojazdów o określonej charakterystyce.

Każde zlecenie polega na przewiezieniu pewnej ilości towaru od jednego klienta do drugiego. Każdy z klientów wymaga, aby obsługa miała miejsce w określonych godzinach. Dodatkowo, aby zlecenie uznać za zrealizowane, klient musi zostać odwiedzony w określoną w zleceniu liczbę dni. Każdego z tych dni musi nastąpić przewóz całości towaru przez dokładnie jeden samochód.

Spośród dopuszczalnych rozwiązań należy wskazać to, które ma jak najmniejszy koszt (patrz rozdział 1.3).

1.1 Definicja problemu

Dane są następujące obiekty:

- zbiór klientów i baz C wraz z odległościami pomiędzy nimi. Jeśli $c_1, c_2 \in C$ to $w_{12} = w_{21}$ jest odległością pomiędzy klientem c_1 a klientem c_2. Odległość jest symetryczna i spełnia nierówność trójkąta (tj. $\forall c_1, c_2, c_3 \in C, w_{12} + w_{23} \geq w_{13}$).
- zbiór typów produktów PT. Każdy typ produktu scharakteryzowany jest nazwą.
- zbiór zamówień. Każde zamówienie jest opisane poprzez:
  - jeden rodzaj produktu $pt \in PT$ i jego ilość (całkowitą) do przewiezienia,
  - punkt początkowy i końcowy (od którego klienta podjąć towar i do którego dostarczyć); każdy punkt pojawi się w dokładnie jednym zamówieniu, dokładnie jeden raz,
  - okna czasowe podjęcia i dostarczenia towaru (godziny, dokładność minutowa),
  - wymaganą liczbę wizyt w ciągu tygodnia (1-7).
- zbiór typów pojazdów VT. Każdy typ pojazdu opisany jest:
  - nazwą,
  - stałym kosztem użycia pojazdu,
  - kosztem wozokilometra (tj. kosztem przejazdu jednego kilometra),
  - stałą prędkością poruszania,
  - zbiorem możliwości. Możliwość to para (typ produktu, ilość). Ilość jest liczbą całkowitą. Opisuje ona jaka ilość danego produktu wypełnia całkowicie samochód.
  - bazą (elementem C); każdego dnia każdy pojazd określonego typu zaczyna i musi skończyć swoją trasę w bazie,
- zbiór pojazdów VE, dostępna jest dowolna liczba pojazdów dowolnego typu.

Celem jest znalezienie planu realizacji wszystkich zamówień. Plan układany jest na jeden tydzień (7 dni). Musi on spełniać następujące warunki:

- każdy klient jest odwiedzony dokładnie tyle razy, ile wizyt zadeklarował w zamówieniu, nie częściej niż raz dziennie,
- każdy samochód odwiedzający klienta dostarcza lub odbiera całość towaru wyspecyfikowanego w zamówieniu,
- w żadnym punkcie swojej trasy samochód nie może wieźć łącznie więcej towarów, niż jest to przewidziane w jego specyfikacji (patrz rozdział 1.2),
- odbiór towaru musi nastąpić przed jego dostarczeniem,
- przyjazd do punktu obsługi musi nastąpić nie później niż na koniec okna czasowego,
- przyjazd przed początkiem okna czasowego skutkuje oczekiwaniem do jego rozpoczęcia,

- każdego dnia samochód wyjeżdża z bazy najwcześniej o godzinie 0:00 i musi do niej powrócić najpóźniej o 23:59.

Zakłada się, że czas obsługi klienta jest nieistotny (zerowy).

1.2 Załadowanie samochodu

Możliwości samochodu są opisane poprzez typy towarów $i = 1, \ldots, I$, jakie może przewozić z przypisaną do nich ilością $a_i > 0$, która wypełnia cały pojazd.

Dopuszczalne są załadowania towarami tylko w takich ilościach $0 \leq c_i \leq a_i$, które nie przekraczają łącznie pojemności pojazdu:

$$\sum_{i=1}^{I} \frac{c_i}{a_i} \leq 1$$

Przykładowo, jeżeli samochód może przewozić 30 palet albo 2000 kartonów, to dopuszczalne są np. załadowania:

- 0 palet, 2000 kartonów (100% wypełnienia),
- 30 palet, 0 kartonów (100% wypełnienia),
- 10 palet, 1000 kartonów (83% wypełnienia),
- 15 palet, 1000 kartonów (100% wypełnienia).

Niedopuszczalne jest natomiast załadowanie 20 palet i 1500 kartonów (142% wypełnienia).

1.3 Koszt rozwiązania

Koszt rozwiązania jest sumą kosztów harmonogramów na każdy dzień. Koszt każdego dnia jest sumą:

- sumy stałych kosztów użycia po wszystkich używanych w tym dniu pojazdach,
- sumy kosztów przejechanych kilometrów po wszystkich zaplanowanych tego dnia trasach.

2 Specyfikacja formatu danych

```
# Regular expressions quick intro:
# - <a> is the non-terminal a
# - "abc" means "the literal string abc"
# - <a> = b means "a is defined as b"
# - between any two tokens spaces can be inserted
# - ( a ) means a
# - a b means "a and then b (with optional spaces
#   inbetween)"
# - a | b means "a or b"
# - a* means "a zero or more times"
# - a+ means "a one or more times"
# - a? means "a zero or one time"
# - [abc] means "any from characters a, b or c"
# - [a-b] means "any character in the range a to b"


######### Problem definition specification #########
# Additional context-dependent or semantic
# assumptions:
# - each of <ptype>, <vtype> and <order> must occur
#   at least once
# - all <ptype>-s occur before any <vtype>
# - all <vtype>-s occur before any <order>
# - distance between two customers is calculated as
#   planar Euclidean distance,
#   i.e. d^2 = (p1.x-p2.x)^2 + (p1.y-p2.y)^2
# - all data types (decimals, integers) will be
#   small enough to fit into standard Java types

<problem-definition> = "problem" <ws> <problem-name>
    <eol> (<empty> | <ptype> | <vtype> | <order>)*
<problem-name> = <name>

# Product type definition
# e.g. ptype karton1
<ptype> = "ptype" <ws> <ptype-name> <eol>
<ptype-name> = <name>

# Vehicle type definition
# e.g. vtype tir 1000 4.50 60.0 (europaleta:33,
#      karton1:1000) (16.47, 20.3)
<vtype> = "vtype" <ws> <vtype-name> <ws>
    <vtype-usage-cost> <ws> <vtype-kilometer-cost>
    <ws> <vtype-speed> <ws> <vtype-caps> <ws>
    <vtype-base> <eol>
<vtype-name> = <name>
<vtype-usage-cost> = <decimal>
<vtype-kilometer-cost> = <decimal>
<vtype-speed> = <decimal>
<vtype-caps> = "(" <vtype-cap> ("," <vtype-cap>)* ")"
<vtype-cap> = <ptype-name> ":" <integer>
<vtype-base> = <point>

# Order definition
# e.g. order europaleta:20 (16.47, 20.3) (18.10, 19)
#      08:00-23:59 12:00-17:00 5
<order> = "order" <ws> <order-product> <ws>
    <order-from> <ws> <order-to> <ws>
    <order-from-tw> <ws> <order-to-tw> <ws>
    <order-visits> <eol>
<order-product> = <ptype-name> ":" <integer>
<order-from> = <point>
<order-to> = <point>
<order-from-tw> = <time-window>
<order-to-tw> = <time-window>
<order-visits> = [1-7]


######### Result specification #########
# Additional context-dependent or semantic
# assumptions:
# - there must be exactly 7 different days
# - the result must fulfill all constraints specified
#   in problem description
# - problem specification will not allow for
#   ambiguous result file (e.g. each point will
#   appear only once in only one order)
# - all data types (decimals, integers) will be small
#   enough to fit into standard Java types

<result-definition> = "result" <ws> <problem-name>
    <eol> <days>
<days> = (<day> (<route> | <empty>)*)+
<day> = "day" <ws> <day-number> <eol>
<day-number> = [1-7]
<route> = <vtype-name> <ws> <vtype-base> <ws>
```

```
    <point>+ <ws> <vtype-base> <eol>

######### Common defintions #########
<empty> = <ws>? <eol>

<name> = [a-zA-Z_0-9.\-]+
<integer> = [0-9]+
<decimal> = <integer> ( "." [0-9]+)?
<point> = "(" <decimal> "," <decimal> ")"

<time-window> = <hour-minute> "-" <hour-minute>
<hour-minute> = <hour> ":" <minute>
<hour> = ([0-1] [0-9]) | ("2" [0-3])
<minute> = [0-5] [0-9]

<ws> = " "+
<eol> = "\n" | "\r\n"
```

2.1 Nazwy instancji

Używane nazwy instancji jednoznacznie wskazują na charakterystykę instancji, a dokładniej na wartości parametrów użytych do jej wygenerowania.

Każda instancja ma nazwę postaci

`klienci-rozklad-wizyty-produkty-pojazdy-okna`

gdzie:

`klienci` Liczbą klientów do obsłużenia. Dwukrotność liczby zleceń. Wartości od 20 do 100.

`rozklad` Rozkład klientów, C oznacza *clustered*, czyli rozkład nierównomierny, R oznacza *random*, czyli rozkład jednostajny. Klastry tworzone są wokół baz pojazdów, w liczbie równej liczbie typów pojazdów (`pojazdy`).

`wizyty` Oczekiwana liczba wizyt każdego z klientów. Wartość od 1 do 7.

`produkty` Liczba typów produktów przydzielanych do zamówień przez generator (wynikowa liczba typów produktów w instancji może być mniejsza). Wartość od 1 do 10.

`pojazdy` Liczba typów pojazdów przydzielanych do zamówień przez generator (wynikowa liczba typów pojazdów w instancji może być mniejsza). Wartość od 1 do 10.

`okna` Okna czasowe, a dokładniej ich ścisłość. S (*short*) oznacza wąski okna czasowe, M (*medium*) – średnie, a L (*large*) – szerokie. Szerokość okien ma najczęściej bezpośrednie przełożenie na średnią długość trasy.

3 Interfejsu systemu

Ponieważ programy uruchamiane są w warunkach konkursowych, muszą spełniać ścisłe wymagania dotyczące komunikacji ze środowiskiem, w którym są uruchamiane. Przede wszystkim, program jest kompilowany i uruchamiany w jednolity sposób, z użyciem narzędzi wspólnych dla wszystkich projektów.

3.1 Środowisko uruchomieniowe

Programy będą uruchamiane w środowisku Java 5 lub nowszym. Dostępny będzie jeden procesor. Dostępna pamięć na stertę to 512MB. Można tworzyć wątki (np. celem regularnego zapisu najlepszego rozwiązania na wyjście), jednak będą one uruchamiane w ramach jednego procesora.

3.2 Struktura katalogów

Na poziomie głównym repozytorium projektu istotny jest tylko jeden katalog, `code`. Żaden inny katalog nie będzie pobierany z repozytorium.

W katalogu `code` znaczenie mają następujące pliki i katalogi:

`src` Katalog z kodami źródłowymi. Będzie on podlegał kompilacji podczas konkursu. Dodatkowo, wszystkie pliki nie-Javowe będą skopiowane do katalogu docelowego.

`lib` Katalog z opcjonalnymi bibliotekami. Może być pusty. Sam katalog jak i wszystkie pliki z rozszerzeniem `.jar` będą umieszczone w ścieżce `classpath` zarówno podczas kompilacji, jak i uruchomienia programu.

`pl.put.two2007.to.contest.Run` Nazwa klasy (oczywiście umieszczonej w odpowiednim pliku), która będzie uruchamiana z linii poleceń.

3.3 Wczytywanie danych i zapis wyników

Po uruchomieniu program powinien wczytać jedną instancję ze standardowego wejścia. Będzie ono przekierowane, więc koniec danych będzie do wykrycia poprzez EOF.

Wszystkie wyniki powinny trafiać na standardowe wyjście. Można wypisywać wiele rozwiązań, system zewnętrzny sam wybierze najlepsze. Limit wypisywanych rozwiązań to 10 na minutę konkursu – jeśli na wyjściu pojawi się więcej rozwiązań, to system zignoruje wszystkie rozwiązania, które wychodzą poza limit całkowity (tzn. dla konkursu dziesięciominutowego limit rozwiązań wynosi 100).

Przed samym zakończeniem działania system wykona „przepchanie" wyjścia (*flush*), aby uniknąć problemu nie odczytania danych zalegających w buforze.

Program nie może w sposób jawny odwoływać się do jakichkolwiek zasobów w systemie: plików, sieci, innych procesów, itp. Można jedynie wczytywać dane, które znajdowały się w katalogu źródłowym.

4 Zmiany w dokumencie

2007-12-04

- dospecyfikowano czas wyjazdu i przyjazdu pojazdu z bazy
- dospecyfikowano konieczność realizację całości zamówienia

2007-11-08

- dostosowano opis do nowego kursu
- uproszczono specyfikację formatu
- dospecyfikowano środowisko

2007-01-12

- dodano opis nazewnictwa instancji

2006-10-27

- uproszczono format wejściowy (określono kolejność typów danych)
- podano specyfikę środowiska uruchomieniowego

2006-10-19

- włączono specyfikację formatu danych
- dodano informację o pojedynczej wizycie w każdym punkcie

2006-10-08

- zmieniono definicję kosztu na odległość pomiędzy klientami
- dopisano informację o zerowym czasie obsługi

2006-10-05

- poprawienie opisu wozokilometra
- uszczegółowienie liczby typów produktów w jednym zamówieniu
- dodanie opisu kosztu
- zastąpienie grafu zbiorem klientów i kosztów
- dopisanie koncepcji bazy samochodów
- uszczegółowienie okien czasowych
- dopisanie realizacji całości zlecenia w każdym zaplanowanym dniu

2006-09-27

- pierwsza wersja