Problema de Ruteo de Vehículos

Solución mediante Recocido Simulado

Manuel Valenzuela Rendón
valenzuela@itesm.mx

1 de septiembre de 2016

Búsqueda local

Algoritmo 1: Búsqueda local básica

```
1  u ← u_0
2  f_u ← f(u)
3  mejor ← u
4  f_mejor ← f_u
5  repeat
6      Generar vecino v de u
7      f_v ← f(v)
8      if f_v es mejor que f_mejor then
9          mejor ← v
10         f_mejor ← f_v
11     if se acepta v then
12         u ← v
13 until cumplir criterio de terminación
```

Búsqueda local

- El criterio de terminación puede depender de la regla de aceptación, o puede ser estar relacionado con un número de vecinos generados si obtener mejora en el mejor encontrado.
- La aceptación depende de $f(v)$ y su relación con $f(u)$.
- Usualmente, el vecino v se crea haciendo una modificación pequeña al estado actual u mediante una *función de vecindad*. Esta función de vecindad depende del problema.
- A la forma en que se decide si se acepta el vecino se le llama *regla de aceptación*. En las siguientes secciones se describirán algunas formas de la regla de aceptación.

Búsqueda local

- El criterio de terminación puede depender de la regla de aceptación, o puede ser estar relacionado con un número de vecinos generados si obtener mejora en el mejor encontrado.
- La aceptación depende de $f(v)$ y su relación con $f(u)$.
- Usualmente, el vecino v se crea haciendo una modificación pequeña al estado actual u mediante una *función de vecindad*. Esta función de vecindad depende del problema.
- A la forma en que se decide si se acepta el vecino se le llama *regla de aceptación*. En las siguientes secciones se describirán algunas formas de la regla de aceptación.

Búsqueda local

- El criterio de terminación puede depender de la regla de aceptación, o puede ser estar relacionado con un número de vecinos generados si obtener mejora en el mejor encontrado.
- La aceptación depende de $f(v)$ y su relación con $f(u)$.
- Usualmente, el vecino v se crea haciendo una modificación pequeña al estado actual u mediante una *función de vecindad*. Esta función de vecindad depende del problema.
- A la forma en que se decide si se acepta el vecino se le llama *regla de aceptación*. En las siguientes secciones se describirán algunas formas de la regla de aceptación.

Búsqueda local

- El criterio de terminación puede depender de la regla de aceptación, o puede ser estar relacionado con un número de vecinos generados si obtener mejora en el mejor encontrado.
- La aceptación depende de $f(v)$ y su relación con $f(u)$.
- Usualmente, el vecino v se crea haciendo una modificación pequeña al estado actual u mediante una *función de vecindad*. Esta función de vecindad depende del problema.
- A la forma en que se decide si se acepta el vecino se le llama *regla de aceptación*. En las siguientes secciones se describirán algunas formas de la regla de aceptación.

Caminata aleatoria

Algoritmo 2: Caminata aleatoria

1 $u \leftarrow u_0$
2 $f_u \leftarrow f(u)$
3 **repeat**
4 | Generar vecino v de u
5 | $f_v \leftarrow f(v)$
6 | $u \leftarrow v$
7 **until** *cumplir criterio de terminación*

Todos los vecinos son aceptados.

Algoritmo 3: Aceptación avara

```
1 u ← u_0
2 f_u ← f(u)
3 repeat
4 |   Generar vecino v de u
5 |   f_v ← f(v)
6 |   if f_v es mejor que f_u then
7 |   |   u ← v
8 until cumplir criterio de terminación
```

Aceptación por umbral

Threshold acceptance

Se aceptan vecinos cuya diferencia en evaluación con el estado actual no es mayor que un umbral T. Este umbral se disminuye cuando no ha habido mejora en los últimos n ciclos.

Algoritmo 4: Aceptación por umbral para maximización

```
1  T ← T_0
2  u ← u_0
3  f_u ← f(u)
4  repeat
5  |   Generar vecino v de u
6  |   f_v ← f(v)
7  |   if (f_v − f_u) > −T then
8  |   |   u ← v
9  |   if no ha habido mejora en n ciclos then
10 |   |   T ← T − ΔT
11 until cumplir criterio de terminación
```

Se aceptan únicamente vecinos que son mejores.

Algoritmo del diluvio

Great Deluge **(Dueck, 1993)**

Algoritmo 5: Algoritmo del diluvio para maximización

```
1  u ← u_0
2  N ← N_0
3  f_u ← f(u)
4  repeat
5  |   Generar vecino v de u
6  |   f_v ← f(v)
7  |   if f_v > N then
8  |   |   u ← v
9  |   |   N ← N + ΔN
10 until cumplir criterio de terminación
```

Se aceptan vecinos que son mejores que un nivel de agua N. Este nivel se modifica cada vez que se acepta un vecino mejor.

Algoritmo record-a-record

***Record-to-record travel* (Dueck, 1993)**

Algoritmo 6: Record-a-record para maximización

```
1  u ← u_0
2  f_u ← f(u)
3  f_mejor ← f_u
4  repeat
5  |   Generar vecino v de u
6  |   f_v ← f(v)
7  |   if f_v > (f_mejor − F) then
8  |   |   u ← v
9  |   if f_v > f_r then
10 |   |   f_mejor ← f_v
11 until cumplir criterio de terminación
```

Se acepta un vecino si tiene una diferencia no mayor a F del mejor encontrado hasta el momento.

Recocido simulado se basa en la idea de simular el proceso de enfriamiento de un material llamado *recocido*.
En recocido simulado, todo vecino igual o mejor que el estado actual es aceptado (línea 7). Los vecinos peores se aceptan con una probabilidad que aumenta con c (temperatura), y disminuye durante mayor sea la diferencia en evaluación del vecino con el estado actual (línea 10). La probabilidad de aceptar un vecino peor está dada por

$$\exp\left(-\frac{f_v - f_u}{c}\right)$$

Algoritmo 7: Recocido Simulado para minimización

```
1  u ← u_0
2  f_u ← f(u)
3  repeat
4      Generar vecino v de u
5      f_v ← f(v)
6      if f_v ≤ f_u then
7          u ← v
8      else
9          if rand() < exp(−(f_v − f_u)/c) then
10             u ← v
11     if se cumple criterio para disminuir c then
12         c ← α · c
13 until cumplir criterio de terminación
```

Problema del vendedor viajero

Traveling salesman problem (TSP)

- Se tienen N ciudades, y se desea la ruta que parte de una ciudad, visita todas las ciudades, regresa a la ciudad inicial, y que tiene un mínimo costo.
- En el TSP euclidiano, el costo es la distancia euclidiana entre las ciudades, y por lo tanto, el costo de ir de la ciudad A a la ciudad B es el mismo que el costo de ir de la ciudad B a la ciudad A.
- En TSP el espacio de búsqueda es de cardinalidad $N!$

Problema del vendedor viajero

Traveling salesman problem (TSP)

- Se tienen N ciudades, y se desea la ruta que parte de una ciudad, visita todas las ciudades, regresa a la ciudad inicial, y que tiene un mínimo costo.
- En el TSP euclidiano, el costo es la distancia euclidiana entre las ciudades, y por lo tanto, el costo de ir de la ciudad A a la ciudad B es el mismo que el costo de ir de la ciudad B a la ciudad A.
- En TSP el espacio de búsqueda es de cardinalidad $N!$

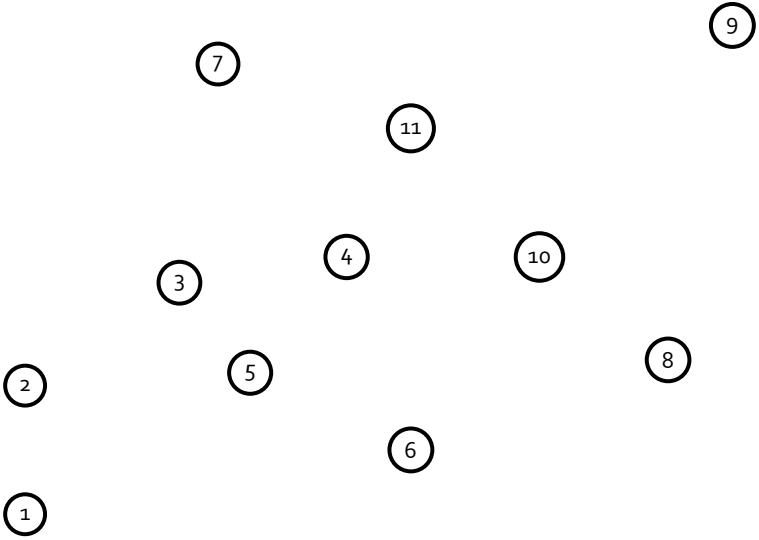
Problema del vendedor viajero

Traveling salesman problem (TSP)

- Se tienen N ciudades, y se desea la ruta que parte de una ciudad, visita todas las ciudades, regresa a la ciudad inicial, y que tiene un mínimo costo.
- En el TSP euclidiano, el costo es la distancia euclidiana entre las ciudades, y por lo tanto, el costo de ir de la ciudad A a la ciudad B es el mismo que el costo de ir de la ciudad B a la ciudad A.
- En TSP el espacio de búsqueda es de cardinalidad $N!$

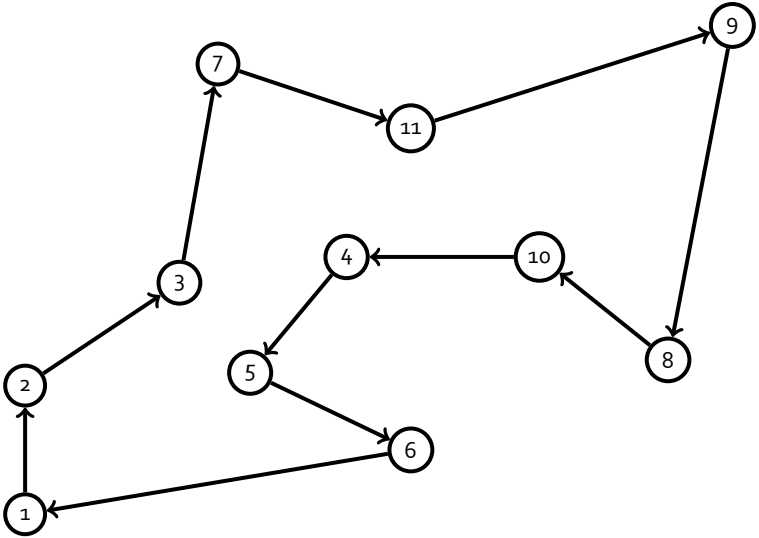
Ejemplo de TSP

Problema...



Ejemplo de TSP

...una solución



En inversión se escogen aleatoriamente dos posiciones en la ruta; la subruta definida por estas dos posiciones se inverte y reinserta. Por ejemplo si tenemos la ruta (estado)

$$\begin{bmatrix}1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\end{bmatrix}$$

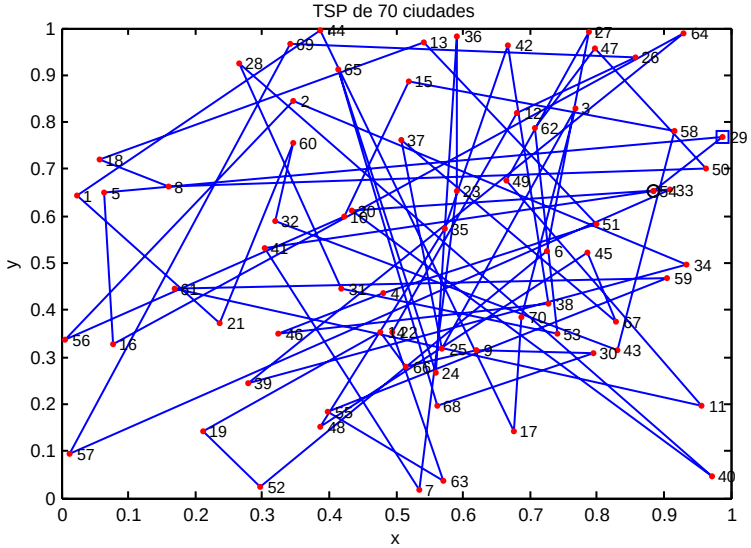
un vecino podría ser

$$\begin{bmatrix}1 & 2 & 3 & 8 & 7 & 6 & 5 & 4 & 9 & 10\end{bmatrix}$$

si las posiciones 4 y 8 hubieran sido escogidas aleatoriamente.

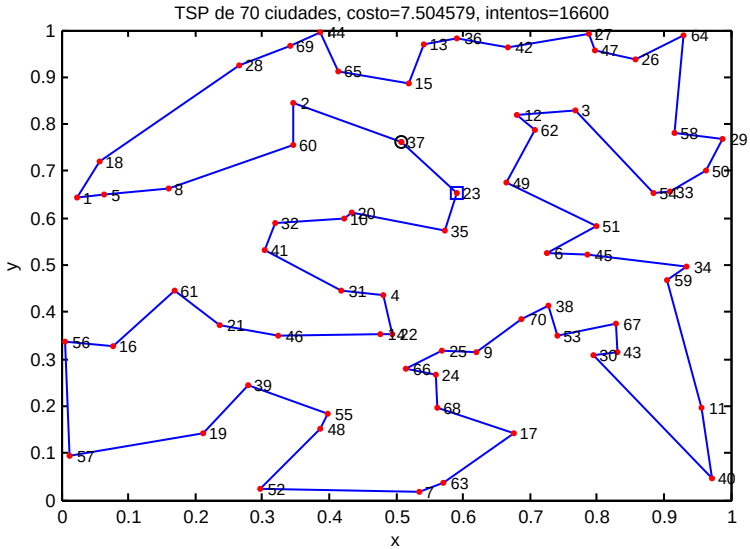
Ejemplo de TSP

Estado inicial



Ejemplo de TSP

Solución



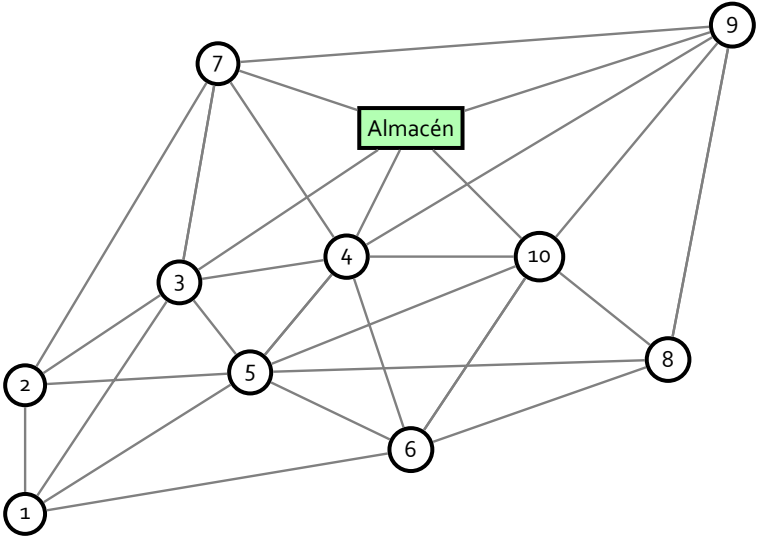
Problema de ruteo de vehículos

***vehicle routing problem* (VRP)**

- Se tienen *N* clientes que se deben visitar
- Se tiene una flotilla de vehículos.
- Se tienen almacenes de donde parten y a donde regresan los vehículos
- Se tiene una red de caminos donde pueden viajar estos camiones.
- Se desea encontrar las rutas que deben seguir los vehículos para que visitar a todos los clientes de manera que se minimice alguna definición de costo.
- Usualmente, el costo está relacionado con la distancia recorrida por los vehículos.

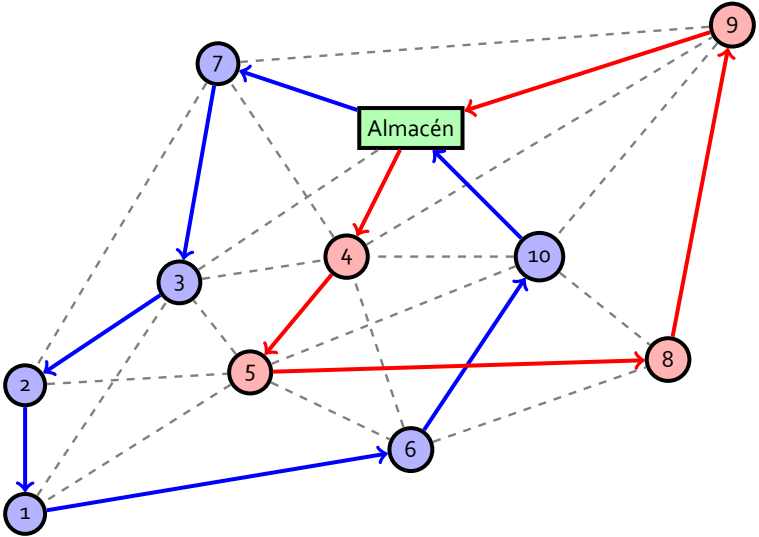
Ejemplo de VRP

Problema...



Ejemplo de VRP

...una solución



Variantes de VRP

- (CVRP) Vehículos con capacidades máximas
  Los clientes tienen demandas de lo que se les debe entregar, y los vehículos tienen una capacidad máxima que pueden transportar. La solución no puede violar la capacidad máxima de los vehículos. La flotilla de vehículos puede ser uniforme o no uniforme, es decir, se pueden tener vehículo con diferentes capacidades.
- (MDVRP) Múltiples almacenes
  Se tienen múltiples almacenes desde donde se pueden atender las demandas de los clientes.
- (VRPTW) Clientes con ventanas de tiempo
  Los clientes establecen ventanas de tiempo en las cuales pueden recibir a los vehículos. Se debe tomar en cuenta el tiempo de recorrido, y el tiempo de servicio a cada cliente.

Variantes de VRP

- (CVRP) Vehículos con capacidades máximas
  Los clientes tienen demandas de lo que se les debe entregar, y los vehículos tienen una capacidad máxima que pueden transportar. La solución no puede violar la capacidad máxima de los vehículos. La flotilla de vehículos puede ser uniforme o no uniforme, es decir, se pueden tener vehículo con diferentes capacidades.
- (MDVRP) Múltiples almacenes
  Se tienen múltiples almacenes desde donde se pueden atender las demandas de los clientes.
- (VRPTW) Clientes con ventanas de tiempo
  Los clientes establecen ventanas de tiempo en las cuales pueden recibir a los vehículos. Se debe tomar en cuenta el tiempo de recorrido, y el tiempo de servicio a cada cliente.

Variantes de VRP

- (CVRP) Vehículos con capacidades máximas
  Los clientes tienen demandas de lo que se les debe entregar, y los vehículos tienen una capacidad máxima que pueden transportar. La solución no puede violar la capacidad máxima de los vehículos. La flotilla de vehículos puede ser uniforme o no uniforme, es decir, se pueden tener vehículo con diferentes capacidades.
- (MDVRP) Múltiples almacenes
  Se tienen múltiples almacenes desde donde se pueden atender las demandas de los clientes.
- (VRPTW) Clientes con ventanas de tiempo
  Los clientes establecen ventanas de tiempo en las cuales pueden recibir a los vehículos. Se debe tomar en cuenta el tiempo de recorrido, y el tiempo de servicio a cada cliente.

Variantes de VRP

- Red de caminos no uniforme y flotilla no uniforme
  Se tiene una red de caminos de diferentes capacidades, y vehículos con diferentes características que no puede todos circular sobre todos los tramos de la red.
- Recolección y entrega
  En esta variante del problema, los vehículos deben entregar y recoger carga de los clientes. Es posible que también se tenga que tomar en cuenta el acomodo de la carga dentro de cada camión.
- Problemas estocásticos y no estacionarios
  En esta variante es posible que los tiempos de recorrido cambien dependiendo de la hora del día, o que sean estocásticos, es decir que el tiempo de recorrido de un punto pueda variar con una cierta distribución de probablidad.

En la página http://neo.lcc.uma.es/vrp/vrp-flavors/ se presetan descripciones de éstas y otras variantes del VRP.

Variantes de VRP

- Red de caminos no uniforme y flotilla no uniforme
  Se tiene una red de caminos de diferentes capacidades, y vehículos con diferentes características que no puede todos circular sobre todos los tramos de la red.
- Recolección y entrega
  En esta variante del problema, los vehículos deben entregar y recoger carga de los clientes. Es posible que también se tenga que tomar en cuenta el acomodo de la carga dentro de cada camión.
- Problemas estocásticos y no estacionarios
  En esta variante es posible que los tiempos de recorrido cambien dependiendo de la hora del día, o que sean estocásticos, es decir que el tiempo de recorrido de un punto pueda variar con una cierta distribución de probablidad.

En la página http://neo.lcc.uma.es/vrp/vrp-flavors/ se presetan descripciones de éstas y otras variantes del VRP.

Variantes de VRP

- Red de caminos no uniforme y flotilla no uniforme
  Se tiene una red de caminos de diferentes capacidades, y vehículos con diferentes características que no puede todos circular sobre todos los tramos de la red.
- Recolección y entrega
  En esta variante del problema, los vehículos deben entregar y recoger carga de los clientes. Es posible que también se tenga que tomar en cuenta el acomodo de la carga dentro de cada camión.
- Problemas estocásticos y no estacionarios
  En esta variante es posible que los tiempos de recorrido cambien dependiendo de la hora del día, o que sean estocásticos, es decir que el tiempo de recorrido de un punto pueda variar con una cierta distribución de probablidad.

En la página http://neo.lcc.uma.es/vrp/vrp-flavors/ se presetan descripciones de éstas y otras variantes del VRP.

Variantes de VRP

- Red de caminos no uniforme y flotilla no uniforme
  Se tiene una red de caminos de diferentes capacidades, y vehículos con diferentes características que no puede todos circular sobre todos los tramos de la red.
- Recolección y entrega
  En esta variante del problema, los vehículos deben entregar y recoger carga de los clientes. Es posible que también se tenga que tomar en cuenta el acomodo de la carga dentro de cada camión.
- Problemas estocásticos y no estacionarios
  En esta variante es posible que los tiempos de recorrido cambien dependiendo de la hora del día, o que sean estocásticos, es decir que el tiempo de recorrido de un punto pueda variar con una cierta distribución de probablidad.

En la página http://neo.lcc.uma.es/vrp/vrp-flavors/ se presetan descripciones de éstas y otras variantes del VRP.

Problemas de Solomon para CVRPTW

- (Solomon, 1987) presentó 56 instancias del problema CVRPTW.
- Sus base de datos de problema ha sido utilizada extensamente para probar algoritmos para el problema de VRPTW.
- En los problema de Solomon se tiene un solo almacén.
- Los vehículos tienen una capacidad máxima igual.
- Los clientes tienen demandas conocidas.
- Los clientes tienen ventanas de tiempo definidas como hora más temprana y hora más tarde en la que pueden ser atendidos.
- Se desea minimizar, primero, el número de vehículos utilizados y el costo de operarlos, y después, el costo es la distancia total recorrida por los vehículos

Problemas de Solomon para CVRPTW

- (Solomon, 1987) presentó 56 instancias del problema CVRPTW.
- Sus base de datos de problema ha sido utilizada extensamente para probar algoritmos para el problema de VRPTW.
- En los problema de Solomon se tiene un solo almacén.
- Los vehículos tienen una capacidad máxima igual.
- Los clientes tienen demandas conocidas.
- Los clientes tienen ventanas de tiempo definidas como hora más temprana y hora más tarde en la que pueden ser atendidos.
- Se desea minimizar, primero, el número de vehículos utilizados y el costo de operarlos, y después, el costo es la distancia total recorrida por los vehículos

Problemas de Solomon para CVRPTW

- (Solomon, 1987) presentó 56 instancias del problema CVRPTW.
- Sus base de datos de problema ha sido utilizada extensamente para probar algoritmos para el problema de VRPTW.
- En los problema de Solomon se tiene un solo almacén.
- Los vehículos tienen una capacidad máxima igual.
- Los clientes tienen demandas conocidas.
- Los clientes tienen ventanas de tiempo definidas como hora más temprana y hora más tarde en la que pueden ser atendidos.
- Se desea minimizar, primero, el número de vehículos utilizados y el costo de operarlos, y después, el costo es la distancia total recorrida por los vehículos

Problemas de Solomon para CVRPTW

- (Solomon, 1987) presentó 56 instancias del problema CVRPTW.
- Sus base de datos de problema ha sido utilizada extensamente para probar algoritmos para el problema de VRPTW.
- En los problema de Solomon se tiene un solo almacén.
- Los vehículos tienen una capacidad máxima igual.
- Los clientes tienen demandas conocidas.
- Los clientes tienen ventanas de tiempo definidas como hora más temprana y hora más tarde en la que pueden ser atendidos.
- Se desea minimizar, primero, el número de vehículos utilizados y el costo de operarlos, y después, el costo es la distancia total recorrida por los vehículos

Problemas de Solomon para CVRPTW

- (Solomon, 1987) presentó 56 instancias del problema CVRPTW.
- Sus base de datos de problema ha sido utilizada extensamente para probar algoritmos para el problema de VRPTW.
- En los problema de Solomon se tiene un solo almacén.
- Los vehículos tienen una capacidad máxima igual.
- Los clientes tienen demandas conocidas.
- Los clientes tienen ventanas de tiempo definidas como hora más temprana y hora más tarde en la que pueden ser atendidos.
- Se desea minimizar, primero, el número de vehículos utilizados y el costo de operarlos, y después, el costo es la distancia total recorrida por los vehículos

Problemas de Solomon para CVRPTW

- (Solomon, 1987) presentó 56 instancias del problema CVRPTW.
- Sus base de datos de problema ha sido utilizada extensamente para probar algoritmos para el problema de VRPTW.
- En los problema de Solomon se tiene un solo almacén.
- Los vehículos tienen una capacidad máxima igual.
- Los clientes tienen demandas conocidas.
- Los clientes tienen ventanas de tiempo definidas como hora más temprana y hora más tarde en la que pueden ser atendidos.
- Se desea minimizar, primero, el número de vehículos utilizados y el costo de operarlos, y después, el costo es la distancia total recorrida por los vehículos

Problemas de Solomon para CVRPTW

- (Solomon, 1987) presentó 56 instancias del problema CVRPTW.
- Sus base de datos de problema ha sido utilizada extensamente para probar algoritmos para el problema de VRPTW.
- En los problema de Solomon se tiene un solo almacén.
- Los vehículos tienen una capacidad máxima igual.
- Los clientes tienen demandas conocidas.
- Los clientes tienen ventanas de tiempo definidas como hora más temprana y hora más tarde en la que pueden ser atendidos.
- Se desea minimizar, primero, el número de vehículos utilizados y el costo de operarlos, y después, el costo es la distancia total recorrida por los vehículos

Datos del cliente *i*

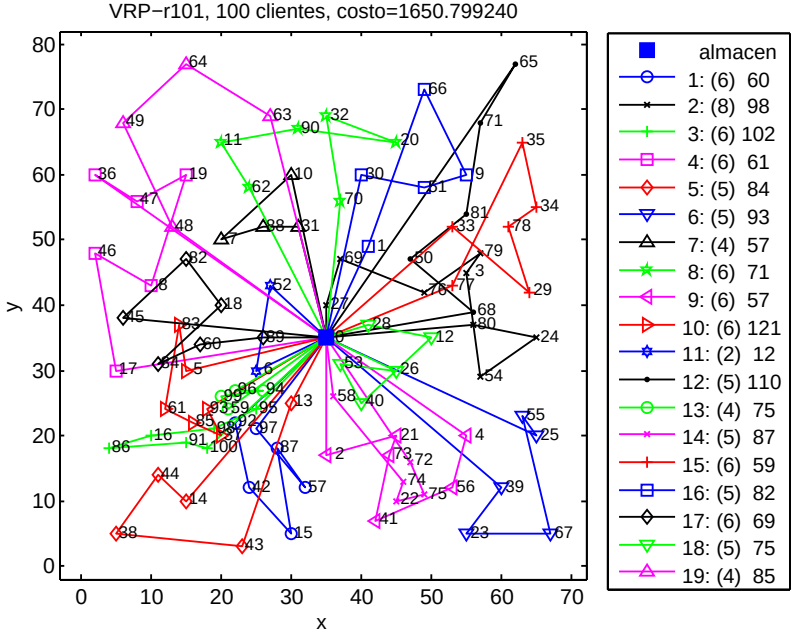
- s_i unidades de tiempo para entregar. (*Service time*)
- d_i demanda.
- $[e_i, l_i]$ ventana de tiempo para el cliente ***i*,** donde e_i es el tiempo más temprano de atención (*earliest*) y l_i es el tiempo más tarde de atención (*latest*). (También se les llama *ready time* y *due date*.)
- b_i tiempo en el que se inicia el servicio para el cliente *i*. (Esto es parte de la solución.)
- $b_j = \max\{e_j, b_i + s_i + t_{ij}\}$ Porque el cliente no va a recibir al camión antes de su ventana de tiempo.

Más datos del problema

Además, se deben tener los siguientes datos del problema:

- t_{ij} tiempo para viajar del cliente i al cliente j
- d_{ij} distancia directa entre cliente i y cliente j

Es posible que en lugar de la distancia y el tiempo entre clientes, se den las coordenadas (x_i, y_i) de la colocación geográfica del cliente i y la distancia se calcule como la distancia euclidiana. En este caso, el tiempo es la distancia multiplicada por una velocidad fija de 1.


VRP−r101, 100 clientes, costo=1650.799240
x
y
almacen
1: (6) 60
2: (8) 98
3: (6) 102
4: (6) 61
5: (5) 84
6: (5) 93
7: (4) 57
8: (6) 71
9: (6) 57
10: (6) 121
11: (2) 12
12: (5) 110
13: (4) 75
14: (5) 87
15: (6) 59
16: (5) 82
17: (6) 69
18: (5) 75
19: (4) 85

Mejor solución conocida para el problema R101

Costo total: 1650.799240
Tiempo máximo: 219.055385
Capacidad máxima: 200

no.	clientes	terminación	capacidad	ruta
1:	(6)	160.20	60	92 42 15 87 57 97
2:	(8)	213.10	98	27 69 76 79 3 54 24 80
3:	(6)	219.04	102	95 98 16 86 91 100
4:	(6)	197.41	61	36 47 19 8 46 17
5:	(5)	186.27	84	14 44 38 43 13
6:	(5)	215.54	93	39 23 67 55 25
7:	(4)	159.50	57	31 88 7 10
8:	(6)	213.10	71	62 11 90 20 32 70
9:	(6)	184.00	57	2 21 73 41 56 4
10:	(6)	218.25	121	5 83 61 85 37 93
11:	(2)	120.18	12	52 6
12:	(5)	177.42	110	65 71 81 50 68
13:	(4)	160.26	75	59 99 94 96
14:	(5)	219.06	87	72 75 22 74 58
15:	(6)	208.70	59	33 29 78 34 35 77
16:	(5)	191.39	82	30 51 9 66 1
17:	(6)	200.06	69	45 82 18 84 60 89
18:	(5)	153.18	75	28 12 40 53 26
19:	(4)	202.80	85	63 64 49 48

Mejor solución conocida para el problema R101

i	d_i	e_i	b_i	l_i	ruta
1:	10	161	166	171	16
2:	7	50	50	60	9
3:	13	116	116	126	2
4:	19	149	149	159	9
5:	26	34	34	44	10
6:	3	99	99	109	11
7:	5	81	90	91	7
8:	9	95	104	105	4
9:	16	97	107	107	16
10:	16	124	124	134	7
11:	12	67	76	77	8
12:	19	63	63	73	18
13:	23	159	165	169	5
14:	20	32	32	42	5
15:	8	61	61	71	1
16:	19	75	77	85	3
17:	2	157	157	167	4
18:	12	87	87	97	17
19:	17	76	77	86	4
20:	9	126	126	136	8
21:	11	62	70	72	9
22:	18	97	97	107	14
23:	29	68	68	78	6
24:	3	153	162	163	2
25:	6	172	172	182	6
26:	17	132	132	142	18
27:	16	37	37	47	2
28:	16	39	39	49	18
29:	9	63	63	73	15
30:	21	71	71	81	16
31:	27	50	50	60	7
32:	23	141	147	151	8
33:	11	37	37	47	15
34:	14	117	117	127	15
35:	8	143	143	153	15
36:	5	41	41	51	4
37:	8	134	134	144	10
38:	16	83	90	93	5
39:	31	44	44	54	6
40:	9	85	87	95	18
41:	5	97	104	107	9
42:	5	31	39	41	1
43:	7	132	132	142	5
44:	18	69	69	79	5
45:	16	32	32	42	17
46:	1	117	124	127	4
47:	27	51	59	61	4
48:	36	165	165	175	19
49:	30	108	108	118	19
50:	13	124	124	134	12
51:	10	88	90	98	16
52:	9	52	52	62	11
53:	14	95	104	105	18
54:	18	140	142	150	2
55:	2	136	136	146	6
56:	6	130	130	140	9
57:	7	101	110	111	1
58:	18	200	200	210	14
59:	28	18	18	28	13
60:	3	162	162	172	17
61:	13	76	76	86	10
62:	19	58	58	68	8
63:	10	34	35	44	19
64:	9	73	73	83	19
65:	20	51	51	61	12
66:	25	127	131	137	16
67:	25	83	90	93	6
68:	36	142	146	152	12
69:	6	50	54	60	2
70:	5	182	182	192	8
71:	15	77	77	87	12
72:	25	35	35	45	14
73:	9	78	84	88	9
74:	8	149	149	159	14
75:	18	69	69	79	14
76:	13	73	77	83	2
77:	14	179	179	189	15
78:	3	96	96	106	15
79:	23	92	97	102	2
80:	6	182	182	192	2
81:	26	94	101	104	12
82:	16	55	55	65	17
83:	11	44	51	54	10
84:	7	101	110	111	17
85:	41	91	91	101	10
86:	35	94	94	104	3
87:	26	93	93	103	1
88:	9	74	74	84	7
89:	15	176	181	186	17
90:	3	95	97	105	8
91:	1	160	160	170	3
92:	2	18	18	28	1
93:	22	188	188	198	10
94:	27	100	100	110	13
95:	20	39	39	49	3
96:	11	135	135	145	13
97:	12	133	133	143	1
98:	10	58	58	68	3
99:	9	83	83	93	13
100:	17	185	185	195	3

Funciones de vecindad para VRPTW

(Bräysy y Gendreau, 2005a)

Las funciones para generar vecinos se clasifican en aquellas que modifican a una sola ruta y las que modifican a dos (o más) rutas.

Funciones para una ruta:

- 2-opt
- Or-opt
- Intercambio intra-ruta

Funciones para dos rutas:

- 2-opt$*$
- Operador de relocación (*Relocate operator*)
- Operador de intercambio (*Exchange operator*)
- Intercambio cruzado (*Cross exchange operator*)
- Operador de intercambio GENI (*GENI exchange operator*)

Funciones de vecindad para VRPTW

(Bräysy y Gendreau, 2005a)

Las funciones para generar vecinos se clasifican en aquellas que modifican a una sola ruta y las que modifican a dos (o más) rutas.
Funciones para una ruta:

- 2-opt
- Or-opt
- Intercambio intra-ruta

Funciones para dos rutas:

- 2-opt∗
- Operador de relocación (*Relocate operator*)
- Operador de intercambio (*Exchange operator*)
- Intercambio cruzado (*Cross exchange operator*)
- Operador de intercambio GENI (*GENI exchange operator*)

Funciones de vecindad para VRPTW

(Bräysy y Gendreau, 2005a)

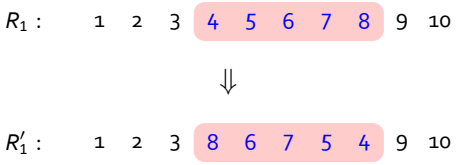
Las funciones para generar vecinos se clasifican en aquellas que modifican a una sola ruta y las que modifican a dos (o más) rutas.
Funciones para una ruta:

- 2-opt
- Or-opt
- Intercambio intra-ruta

Funciones para dos rutas:

- 2-opt$*$
- Operador de relocación (*Relocate operator*)
- Operador de intercambio (*Exchange operator*)
- Intercambio cruzado (*Cross exchange operator*)
- Operador de intercambio GENI (*GENI exchange operator*)

Se escogen dos posiciones aleatorias en una ruta, y se invierte la subruta entre esas dos posiciones.



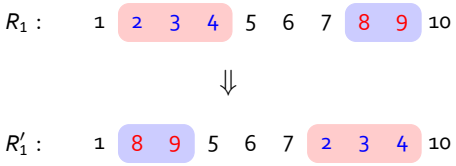
Se escoge aleatoriamente una subruta y una posición a lo largo de la ruta. La subruta se coloca después de la posición escogida.

R_1 : 1 2 3 **4 5 6** 7 8 9 10

⇓

R'_1 : 1 2 3 7 8 **4 5 6** 9 10

Se escogen aleatoriamente dos subrutas y se intercambia su posición.



Para cada ruta se escoge aleatoriamente una posición. Se coloca el cliente de la posición en la primera ruta después de la posición de la segunda ruta.

R_1 : 1 2 3 4 5 6 7 8

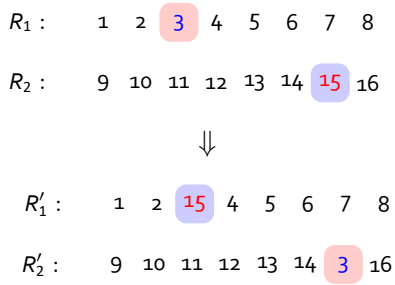
R_2 : 9 10 11 12 13 14 15 16

$\Downarrow$

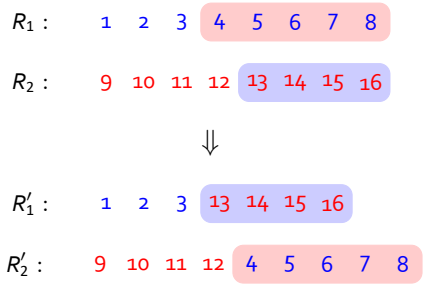
R_1' : 1 2 4 5 6 7 8

R_2' : 9 10 11 12 13 14 15 3 16

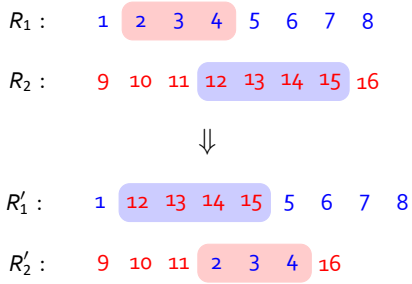
Para cada ruta se escoge aleatoriamente una posición. Se intercambian de ruta los clientes en las posiciones.



Se escoge aleatoriamente una arista en cada ruta. Se intercambian las subrutas después de las aristas.



Para cada ruta se escoge aleatoriamente una subruta. Se intercambian las subrutas.



Destrucción y reconstrucción

Ruin and Recreate (R&R)

- Es una forma de generar vecinos propuesta por Schrimpf, Schneider, Stamm-Wilbrandt, y Dueck.
- En lugar de generar pequeñas modificaciones al estado u, se hacen modificaciones grandes.
- De alguna forma, se destruye una parte significativa del estado u, y luego de alguna forma, posiblemente complicada, se reconstruye el estado para generar el vecino.
- La generación del vecino por medio de R&R tiene dos partes: destrucción y reconstrucción.
- En la destrucción se elimina alguna significativamente del estado. En la reconstrucción se reconstruyen la parte eliminada de manera que se produzca un estado válido.

Destrucción

Ruin

Existen diferentes formas en las que se podría realizar la destrucción en el problema de VRPTW:

Destrucción radial Se selecciona aleatoriamente un cliente c. Se escoge un número aleatorio A tal que $A \leq [F \cdot N]$ donde $0 < F < 1$. Se elimina c y sus $A - 1$ vecinos más cercanos del estado. Estos clientes eliminados se guardan en el conjunto B. Los vecinos más cercanos se obtienen de acuerdo a la distancia euclidiana.

Destrucción aleatoria Se selecciona un número aleaorio A tal que $A \leq [F \cdot N]$ donde $0 < F < 1$. Se eliminan A clientes del estado escogidos aleatoriamente, y se guardan en el conjunto B. Nótese que a diferencia de la destrucción radial, ahora se escogen los clientes aleatoriamente.

Destrucción secuencial Se selecciona un número aleaorio A tal que $A \leq [F \cdot N]$ donde $0 < F < 1$. Se eliminan A clientes sucesivos de una ruta escogida aleatoriamente.

Mejor inserción Se toman clientes del conjunto B en orden aleatorio. Para cada cliente que se va a insertar en el estado, se evalúa el costo de incluírlo en una de la rutas definidas en el estado, y se escoge la ruta y posición de menor aumento de costo. En caso de que no sea posible insertarlo en ninguna ruta existente sin violar alguna restricción, se crea una ruta adicional para este cliente. Se procede de esta forma hasta que no haya más clientes en el conjunto B. El nuevo estado, es el vecino.

Referencias

Aarts, E., Korst, J., y Michiels, W. (2005). Simulated annealing. En E. Burke y G. Kendall (Eds.), *Search methodologies* (pp. 187–210). Berlin: Springer US. Descargado de http://dx.doi.org/10.1007/0-387-28356-0_7 doi: 10.1007/0-387-28356-0_7

Bräysy, O., y Gendreau, M. (2005a). Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Science*, *39*(1), 104–108. Descargado de http://dx.doi.org/10.1287/trsc.1030.0056

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and record-to-record travel. *Journal of Computational Physics*, *104*, 86–92.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., y Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, *159*(139–171).

Solomon, M. M. (1987). Algorithms for the vehicle routing problem with time windows. *Operations Research*, *35*(2), 254–265.