Metaheurísticas

Seminario 2. Problemas de optimización con técnicas basadas en búsqueda local

Problema de Asignación Cuadrática (QAP)

- Definición del Problema
- Representación y Ejemplo: Diseño de un Hospital. Solución Greedy
- Búsquedas por Trayectorias Simples
- La Biblioteca QAPLIB

Definición del Problema

- El Problema de Asignación Cuadrática (QAP) está considerado como uno de los problemas de optimización combinatoria más complejos

- Es NP-completo. Incluso la resolución de problemas pequeños de tamaño n>25 se considera una tarea computacionalmente muy costosa

- El problema general consiste en encontrar la asignación óptima de n unidades a n localizaciones, conociendo la distancia entre las primeras y el flujo existente entre las segundas

Definición del Problema

- Sean n unidades (u_i, i=1,...,n) y n localizaciones (l_j, j=1,...,n). Se dispone de dos matrices $F=(f_{ij})$ y $D=(d_{ij})$, de dimensión (n×n) con la siguiente interpretación:
  - F es la matriz de flujo, es decir, f_{ij} es el flujo que circula entre la unidad i y la j
  - D es la matriz de distancias, es decir, d_{kl} es la distancia existente entre la localización k y la l

- El costo de asignar simultáneamente u_i a l_k y u_j a l_l es:

$$f_{ij} \cdot d_{kl}$$

Definición del Problema

- La definición matemática del problema consiste en minimizar el costo de las asignaciones:

$$
\begin{aligned}
\min \quad & \sum_{i,j=1}^{n} \sum_{k,p=1}^{n} f_{ij} d_{kp} x_{ij} x_{kp} & \\
\text{s.a.} \quad & \sum_{i=1}^{n} x_{ij} = 1 & 1 \leq j \leq n, \\
& \sum_{j=1}^{n} x_{ij} = 1 & 1 \leq i \leq n, \\
& x_{ij} \in \{0,1\} & 1 \leq i, j \leq n.
\end{aligned}
$$

- Como se puede ver, la función de coste es cuadrática, lo que da nombre al problema y lo complica sustancialmente

Definición del Problema

- **Problema de la asignación cuadrática, *QAP*:**

  Dadas n unidades y n localizaciones posibles, el problema consiste en determinar la asignación óptima de las unidades en las localizaciones conociendo el flujo existente entre las primeras y la distancia entre las segundas

- Si se considera una permutación para representar las asignaciones, se verifican directamente las restricciones del problema:

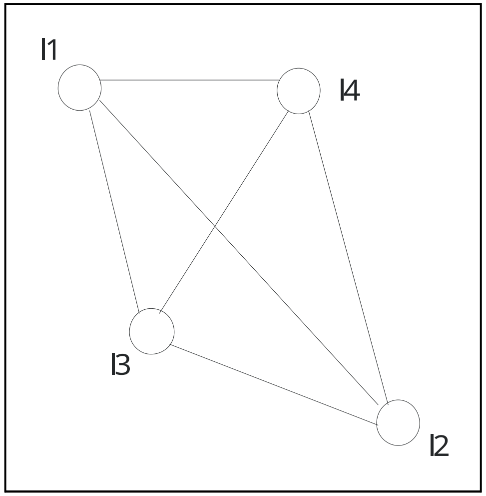
$$QAP = \min_{S \in \Pi_N} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{S(i)S(j)} \right)$$

Representación y Ejemplo: Diseño de un Hospital

Elshafei, A.N. (1977). Hospital Layout as a Quadratic Assignment Problem. Operations Research Quarterly 28, 167-179.

- Supongamos que se ha de diseñar un hospital que comprende cuatro unidades distintas:
  - u1: Maternidad
  - u2: Urgencias
  - u3: Unidad de Cuidados Intensivos
  - u4: Cirugía

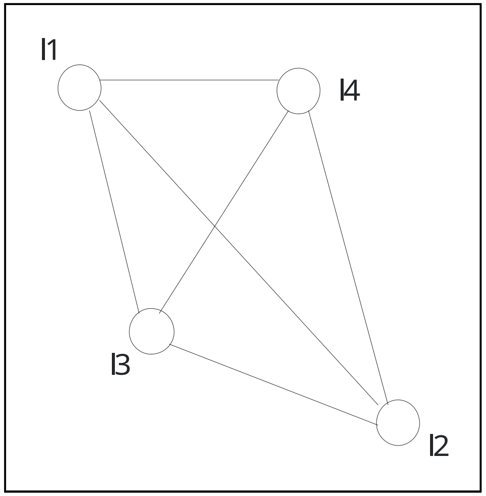
Que han de ser situadas en un edificio con la siguiente distribución:



Representación y Ejemplo: Diseño de un Hospital

- La matriz *D* contiene las distancias existentes entre las diferentes salas
- La matriz *F* recoge el número medio de pacientes que pasan de una unidad a otra cada hora (por ejemplo, podrían ser las medias mensuales, medias anuales, totales anuales, ...)

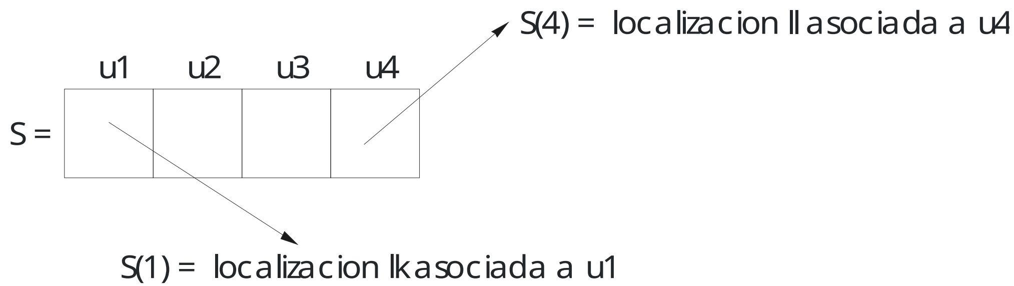
$$D = \begin{pmatrix} 0 & 12 & 6 & 4 \\ 12 & 0 & 6 & 8 \\ 6 & 6 & 0 & 7 \\ 4 & 8 & 7 & 0 \end{pmatrix} \qquad F = \begin{pmatrix} 0 & 3 & 8 & 3 \\ 3 & 0 & 2 & 4 \\ 8 & 2 & 0 & 5 \\ 3 & 4 & 5 & 0 \end{pmatrix}$$



u_1: Maternidad. u_2: Urgencias. u_3: Unidad de Cuidados Intensivos. u_4: Cirugía.

Representación y Ejemplo: Diseño de un Hospital

- Las soluciones son permutaciones del conjunto $N=\{1, 2, 3, 4\}$
- Para entenderlo mejor, podemos pensar en su representación en forma de vector permutación: *las posiciones se corresponden con las unidades y el contenido de las mismas con las localizaciones (salas del hospital) en la que se sitúan las unidades correspondientes*:

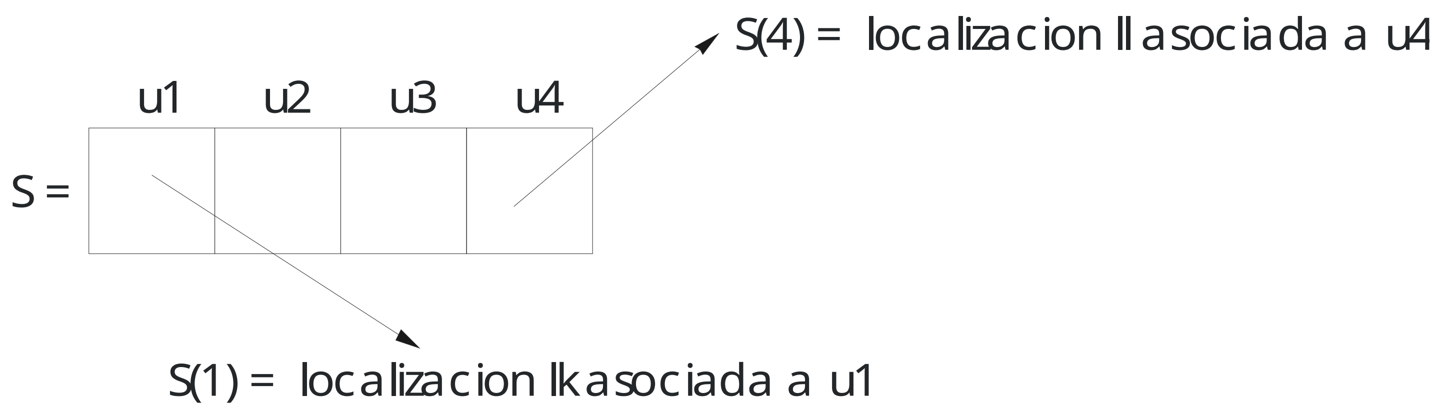


- En este caso, usamos una permutación para representar una asignación, al contrario que en el TSP en el que representa un orden
- Esto nos permite verificar de forma sencilla las restricciones del problema, lo que sería más complicado en otras representaciones como una matriz binaria

Representación y Ejemplo: Diseño de un Hospital

- Así, la solución S={3,4,1,2} representa la siguiente distribución de asignaciones:

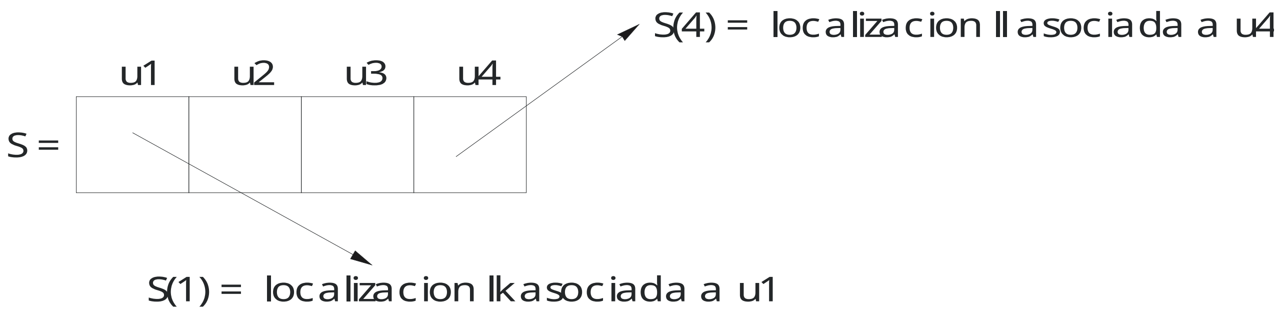
u1 ⇒ l3 u2 ⇒ l4 u3 ⇒ l1 u4 ⇒ l2



Representación y Ejemplo: Diseño de un Hospital

- cuyo costo $C(S)$ es:

$$
\begin{array}{llllll}
 & f_{12}\cdot d_{34} & + \; f_{13}\cdot d_{31} & + \; f_{14}\cdot d_{32} & \quad 3\cdot 7 + 8\cdot 6 + 3\cdot 6 \; + & \\
+ & f_{21}\cdot d_{43} & + \; f_{23}\cdot d_{41} & + \; f_{24}\cdot d_{42} & \quad 3\cdot 7 + 2\cdot 4 + 4\cdot 8 \; + & \\
+ & f_{31}\cdot d_{13} & + \; f_{32}\cdot d_{14} & + \; f_{34}\cdot d_{12} & \quad 8\cdot 6 + 2\cdot 4 + 5\cdot 12 \; + & \\
+ & f_{41}\cdot d_{23} & + \; f_{42}\cdot d_{24} & + \; f_{43}\cdot d_{21} & \quad 3\cdot 6 + 4\cdot 8 + 5\cdot 12 & = \; 374
\end{array}
$$



- Cada asignación unidad-localización influye globalmente en la red, es decir, en todas las transferencias que se efectúan desde la unidad en cuestión

Otras Aplicaciones

- Diseño óptimo de teclados para distintos idiomas en función de la frecuencia de pares de letras. Unid: letras; loc: teclas; flujo: frecuencia de pares de letras; dist: distancia entre las teclas en el teclado

  Burkard, R.E. and J. Offerman (1977). Entwurf von Schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme. Z. Operations Research 21, B121-B123

- Cableado óptimo de placas madre: localización de componentes para reducir el cableado. Unid: componentes; loc: posición en la placa; flujo: nº de cables conectando componentes; dist: distancia entre locs.

  Brixius, N.W. and K.M. Anstreicher (2001). The Steinberg Wiring Problem. In: The Sharpest Cut, The Impact of M. Padberg and His Work, M. Grötschel, ed., SIAM, 2004, 293-307

- Asignación óptima de directores a oficinas: Unid: directores; loc: oficinas; flujo: frecuencia de interacciones entre directores; dist: distancia entre oficinas

  Hanan, M. and J.M. Kurtzberg (1972). A Review of the Placement and Quadratic Assignment Problems, SIAM Review 14, 324-342

Otras Aplicaciones

- Diseño de turbinas: Localización de las aspas de la turbina (con masa ligeramente distinta por la fabricación) de modo que el centro de gravedad coincida con el eje del motor

  J. Mosevich (1986). Balancing hydraulic turbine runners--A discrete combinatorial optimization problem, European Journal of Operational Research 26(2), 202-204

- Diseño óptimo de campus

  J.W. Dickey, J.W. Hopkins (1972). Campus building arrangement using TOPAZ, Transportation Research 6, 59–68

- Diseño de parques forestales

  Bos, J. (1993). A quadratic assignment problem solved by simulated annealing. Journal of Environmental Management, 37(2), 127-145

- Diseño de líneas de producción

  Geoffrion, A.M., and G.W. Graves (1976). Scheduling Parallel Production Lines with Changeover Costs: Practical Applications of a Quadratic Assignment/LP Approach. Operations Research 24, 595-610

Solución Greedy

- La complejidad del problema ha provocado que se hayan aplicado muchos algoritmos aproximados para su resolución

- Analizando la función objetivo podemos determinar que una buena fórmula heurística para resolver el problema es:

Asociar unidades de gran flujo con localizaciones céntricas en la red y viceversa

Solución Greedy

- Podemos construir un algoritmo greedy usando esta heurística mediante dos vectores, el potencial de flujo y el de distancia:

$$\hat{f}_i = \sum_{j=1}^{n} f_{ij} \quad ; \quad i = 1, \ldots, n$$

$$\hat{d}_k = \sum_{l=1}^{n} d_{kl} \quad ; \quad k = 1, \ldots, n$$

- Cuanto mayor sea f_i, más importante es la unidad en el intercambio de flujos y cuanto menor sea d_k, más céntrica es la localización. Por tanto:

el algoritmo irá seleccionando la unidad i libre con mayor $\hat{f}_i$
y le asignará la localización k libre con menor $\hat{d}_k$

Solución Greedy

Algoritmo Greedy-QAP

1. Calcular los potenciales $\hat{f}_i$ y $\hat{d}_k$.
2. $S \leftarrow \emptyset$.
3. Repetir para $x = 1$ hasta n (asignaciones 1 a n):

   3.1. Escoger la unidad u_i no asignada aún ($u_i \notin S$) con mayor valor de $\hat{f}_i$.

   3.2. Escoger la localización l_k no asignada aún ($l_k \notin S$) con menor valor de $\hat{d}_k$.

   3.3. $a_x = (u_i, l_k)$. $S \leftarrow S \cup a_x$.

4. Calcular el costo de S, $C(S)$. Devolver S y $C(S)$.

Solución Greedy

- Si aplicamos el algoritmo greedy propuesto al ejemplo del hospital obtenemos los siguientes resultados:

$$\hat{f} = \begin{pmatrix} 14 \\ 9 \\ 15 \\ 12 \end{pmatrix} \qquad \hat{d} = \begin{pmatrix} 22 \\ 26 \\ 17 \\ 19 \end{pmatrix}$$

$$S = \{(u_3,l_3), (u_1,l_4), (u_4,l_1), (u_2,l_2)\}$$

$$\begin{aligned} C(S) = \; & f_{12} \cdot d_{42} + f_{13} \cdot d_{43} + f_{14} \cdot d_{41} & & 3 \cdot 8 + 8 \cdot 7 + 3 \cdot 4 + \\ & + f_{21} \cdot d_{24} + f_{23} \cdot d_{23} + f_{24} \cdot d_{21} & & 3 \cdot 8 + 2 \cdot 6 + 4 \cdot 12 + \\ & + f_{31} \cdot d_{34} + f_{32} \cdot d_{32} + f_{34} \cdot d_{31} & & 8 \cdot 7 + 2 \cdot 6 + 5 \cdot 6 + \\ & + f_{41} \cdot d_{14} + f_{42} \cdot d_{12} + f_{43} \cdot d_{13} & = \; & 3 \cdot 4 + 4 \cdot 12 + 5 \cdot 6 = 364 \end{aligned}$$

Búsquedas por Trayectorias Simples

- **Representación**: Problema de asignación: una permutación $\pi=[\pi(1), \ldots, \pi(n)]$ en el que las posiciones del vector $i=1,\ldots,n$ representan las unidades y los valores $\pi(1), \ldots, \pi(n)$ contenidos en ellas las localizaciones. Permite verificar las restricciones

- **Operador de vecino de intercambio y su entorno**: El entorno de una solución π está formado por las soluciones accesibles desde ella a través de un movimiento de intercambio

  Dada una solución (asignación de unidades a localizaciones) se escogen dos unidades distintas y se intercambia la localización asignada a cada una de ellas ($Int(\pi,i,j)$):

$$\pi=[\pi(1), \ldots, \pi(i), \ldots, \pi(j), \ldots, \pi(n)]$$

$$\pi'=[\pi(1), \ldots, \pi(j), \ldots, \pi(i), \ldots, \pi(n)]$$

Búsquedas por Trayectorias Simples

- <u>*Int*(π,*i*,*j*) verifica las restricciones</u>, si la solución original π es factible siempre genera una solución vecina π' factible

- Su aplicación provoca que el tamaño del entorno sea:

$$|E(\pi)| = \frac{n \cdot (n-1)}{2}$$

- Las instancias del QAP no suelen ser demasiado grandes y el cálculo factorizado del coste de una solución se realiza de forma eficiente ($O(n)$), permitiendo explorar el entorno completo

  Aún así, dicha exploración requería $O(n^3)$ por lo que es recomendable utilizar una estrategia avanzada, considerando una modalidad de lista de candidatos y seleccionado primero los movimientos más prometedores

Búsqueda Local para el QAP

Stützle, Iterated local search for the quadratic assignment problem, European Journal of Operational Research 174 (2006) 1519–1539

- Algoritmo de búsqueda local del primer mejor: en cuanto se genera una solución vecina que mejora a la actual, se aplica el movimiento y se pasa a la siguiente iteración
  - Se detiene la búsqueda cuando se ha explorado el vecindario completo sin obtener mejora
- Se considera una factorización para calcular el coste de π' a partir del de π considerando sólo los cambios realizados por el movimiento de intercambio
- Se usa la técnica *don't look bits* para la lista de candidatos
  - Se emplea un vector binario de tamaño *n* que asocia un bit a cada unidad
  - Si dicho bit está activado en la iteración actual, no se considera ningún movimiento "que arranque" de la unidad en cuestión

BL-QAP: Factorización del Movimiento de Intercambio

- Sea $C(\pi)$ el coste de la solución original π. Para generar π', el operador de vecino $Int(\pi,r,s)$ escoge dos unidades r y s e intercambia sus localizaciones $\pi(r)$ y $\pi(s)$:
  $\pi=[\pi(1), \ldots, \pi(r), \ldots, \pi(s), \ldots, \pi(n)]$

  $\pi=[\pi(1), \ldots, \pi(r), \ldots, \pi(s), \ldots, \pi(n)]$ $\pi'=[\pi(1), \ldots, \pi(s), \ldots, \pi(r), \ldots, \pi(n)]$

- Por lo tanto, quedan afectados 2·n sumandos, los relacionados con las dos ***viejas*** y las dos ***nuevas*** localizaciones de las dos unidades alteradas

- El coste del movimiento (la diferencia de costes entre las dos soluciones) $\Delta C(\pi,r,s)=C(\pi')-C(\pi)$ se puede factorizar como:

$$\sum_{k=1,k\neq r,s}^{n} \left[\begin{array}{l} f_{rk}\cdot(\boxed{d_{\pi(s)\pi(k)}} - \boxed{d_{\pi(r)\pi(k)}}) + f_{sk}\cdot(\boxed{d_{\pi(r)\pi(k)}} - \boxed{d_{\pi(s)\pi(k)}}) + \\ f_{kr}\cdot(\boxed{d_{\pi(k)\pi(s)}} - \boxed{d_{\pi(k)\pi(r)}}) + f_{ks}\cdot(\boxed{d_{\pi(k)\pi(r)}} - \boxed{d_{\pi(k)\pi(s)}}) \end{array} \right]$$

nuevas

viejas

BL-QAP: Factorización del Movimiento de Intercambio

- Si $\Delta C(\pi,r,s)$ es negativo ($\Delta C(\pi,r,s)<0$), la solución vecina π' es mejor que la actual π (el QAP es un problema de minimización) y se acepta. Si no, se descarta y se genera otro vecino

- El pseudocódigo de la BL del Primer Mejor del Tema 2 de Teoría quedaría:

Repetir

$\pi' \leftarrow$ GENERA_VECINO(π_{act});

Hasta ($\Delta C(\pi,r,s)<0$) **O**
(se ha generado E(π_{act}) al completo)

- El coste $C(\pi')$ de la nueva solución vecina es: $C(\pi')=C(\pi)+\Delta C(\pi)$. Sólo es necesario calcularlo para la solución vecina aceptada

BL-QAP: Definición de la Lista de Candidatos: *Don't Look Bits*

- Técnica que permite focalizar la BL en una zona del espacio de búsqueda en la que potencialmente puede ocurrir algo
- Reduce significativamente el tiempo de ejecución con una reducción muy pequeña de la eficacia de la BL
- Sólo es aplicable con la BL del primer mejor
- En la primera iteración, todos los bits están a 0, es decir, todos las unidades están activadas en un bucle externo y todos sus movimientos pueden ser considerados para explorar el entorno:
  - Si tras probar todos los movimientos asociados a esa unidad, ninguno provoca una mejora, se pone su bit a 1 para desactivarla en el futuro
  - Si una unidad está implicada en un movimiento que genera una solución vecina con mejor coste, se pone su bit a 0 para reactivarla

BL-QAP: Definición de la Lista de Candidatos: *Don't Look Bits*

procedure iterative improvement
  for $i = 1$ **to** n **do**
    if $\texttt{dlb}[i] = 0$ **then**
      $improve\_flag \leftarrow false$
      for $j = 1$ **to** n **do**
        CheckMove(i, j)
        if move improves **then**
          ApplyMove(i, j); $\texttt{dlb}[i] \leftarrow 0$, $\texttt{dlb}[j] \leftarrow 0$
          $improve\_flag \leftarrow true$
      endfor
      if $improve\_flag = false$ **then** $\texttt{dlb}[i] \leftarrow 1$
  end
end iterative improvement

IMPORTANTE: Este es el bucle interno de la BL, el de exploración del vecindario de la solución actual. Sea cual sea la BL usada, siempre habrá un bucle externo que repetirá el proceso mientras se produzca mejora

La Biblioteca QAPLIB

- La QAPLIB es una biblioteca que contiene distintas instancias del QAP llevando un registro de las mejores soluciones obtenidas hasta el momento para las mismas y de las cotas teóricas de la calidad de la mejor solución que se puede obtener

- Es accesible en la Web en las dirección siguiente:

  https://coral.ise.lehigh.edu/data-sets/qaplib/

- En dicha dirección pueden encontrarse tanto los datos como las soluciones de distintas instancias del problema, relacionadas con diferentes aplicaciones

La Biblioteca QAPLIB

- El formato de los ficheros de datos es:

n
A
B

$$\min_{S \in \Pi_N} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot b_{S(i)S(j)} \right)$$

donde *n* es el tamaño del problema y *A* y *B* son, respectivamente las matrices de flujo y distancia, de acuerdo a nuestra codificación de las soluciones del problema

- El formato de los ficheros de soluciones (óptimas o mejores conocidas) es:

n   *sol*
p

donde *n* es el tamaño del problema, *sol* es el coste de la solución y *p* es la permutación correspondiente

La Biblioteca QAPLIB

- La siguiente tabla es un ejemplo de la información que proporciona la QAPLIB:

	name	n	feas.sol.		permutation/bound	gap
	Tai12a	12	224416	(OPT)	(8,1,6,2,11,10,3,5,9,7,12,4)	
	Tai12b	12	39464925	(OPT)	(9,4,6,3,11,7,12,2,8,10,1,5)	
	Tai15a	15	388214	(OPT)	(5,10,4,13,2,9,1,11,12,14,7,15,3,8,6)	
	Tai15b	15	51765268	(OPT)	(1,9,4,6,8,15,7,11,3,5,2,14,13,12,10)	
	Tai17a	17	491812	(OPT)	(12,2,6,7,4,8,14,5,11,3,16,13,17,9,1,10,15)	
	Tai20a	20	703482	(OPT)	(10,9,12,20,19,3,14,6,17,11,5,7,15,16,18,2,4,8,13,1)	
*	Tai20b	20	122455319	(OPT)	(8,16,14,17,4,11,3,19,7,9,1,15,6,13,10,2,5,20,18,12)	
*	Tai25a	25	1167256	(OPT)	(9,4,6,11,5,1,15,10,14,3,17,12,19,18,23,8,21,2,22,7,16,20,24,25,13)	
*	Tai25b	25	344355646	(OPT)	(4,15,10,9,13,5,25,19,7,3,17,6,18,20,16,2,22,23,8,11,21,24,14,12,1)	
	Tai30a	30	1818146	(Ro-TS)	1706855 (L&P)	6.12 %
*	Tai30b	30	637117113	(OPT)	(4 8 11 15 17 20 21 5 14 30 2 13 6 29 10 26 27 24 28 22 12 9 7 23 19	
	Tai35a	35	2422002	(Ro-TS)	2216627 (L&P)	8,48 %