Algoritmo de islas (genético)

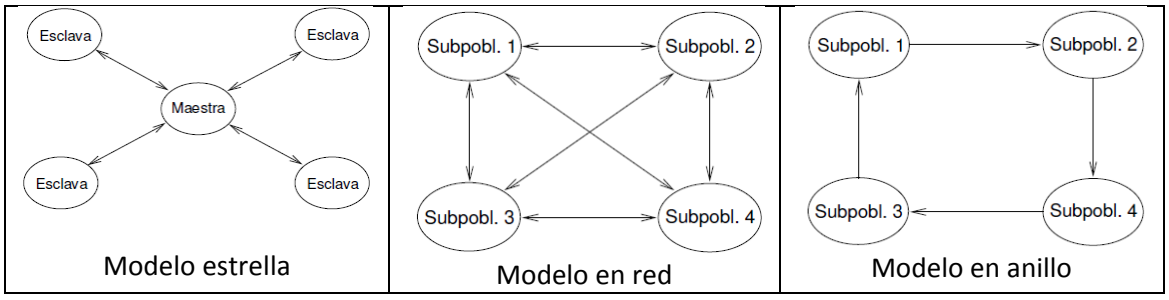
Introducción

El un modelo genético con separación en islas pretende paralelizar la evolución de cada población durante un número de generaciones concretos. Con esto economizamos en tiempo de cómputo distribuyendo las tareas.

Con el algoritmo genético básico tendemos a obtener máximos locales, al subidividir las poblaciones en islas tendremos varios máximos locales. Mas tarde, el intercambio de individuos entre distintas islas produce una reinyección que enriquece la información genética de las distintas islas con los máximos de las otras subpoblaciones.

Existen diferentes tipos de emigraciones:

- Emigración en estrella: existe una isla maestra con la que se comunican todas las esclavas, la maestra tiene la mejor población, aquellas cuya función objetivo tiene mayor valor. A la hora de hacer emigraciones se pasan los mejores.
- Emigración en red: no existe ninguna maestra, todas se comunican con todas.
- Emigración en anillo: solo hay un sentido en la comunicación, cada subpoblación envía sus mejores individuos siempre a la siguiente población vecina. Esta última es la que vamos a utilizar.



Modelo estrella | Modelo en red | Modelo en anillo

Representación

Para este problema de coloración crearemos un genoma con tantos genes como nodos tenga el grafo. El valor de cada gen será un color distinto, por simplificar, un número. Como peor caso, habrá tantos colores como nodos haya. Si el grafo fuera plano existe un teorema en el que dicho grafo sería 4-coloreable, pero esto no lo podemos presuponer.

La población se subidivide en N-islas (4 en nuestro caso), cada subpoblación es un conjunto de genes de una población inicial.

La función objetivo busca primeramente que no haya conflictos en la posible solución, conseguimos un grafo sin conflicto intentará optimizarlo de manera que contengan el mínimo número de colores distintos.

Salas Cantalejo, Alberto
Bellés Camacho, Víctor Juan

Pseudocódigo

A continuación describiremos los principales métodos utilizados para la realización de este modelo de islas.

El algoritmo principal, al que se le pasa el tamaño de la población, procuraremos que sea divisible por el número de islas que tiene nuestro algoritmo (4), la frecuencia que hace referencia al número de generaciones que querremos antes de generar una emigración. M, la probabilidad de mutaciones en cada generación. G, el número de generaciones totales que se van a realizar.

`modeloIslas (tamPoblacion frecuencia m g) {`	

`generaPoblacionInicial tamPoblacion (length *grafoProblema*)`
`generaIslas *poblacionInicial*`
`generaGenes`
	Inicializamos la población aleatoriamente y la repartimos entre las islas

`let* (poblacionOrdenada '())`
`for i from 0 to 3{`
`for j from 0 to g do{`
`(algoGenetico i frecuencia m)`
`generaciones[i]++`
`}`
`}`
	Parte paralelizable del algoritmo, donde se realiza el algoritmo genético.

`setf poblacionOrdenada (evaluaPoblacion)`
`first poblacionOrdenada`
`}`
	Ordena el resultado y coge el mejor elemento (primero).

El algoritmo genético realiza una generación sobre la isla seleccionada, si esta en la frecuencia-esima generación realiza una migración de los mejores individuos.

`algoGenetico (posisla frecuencia m){`	
`evaluacion = (evaluacionIndividuos posisla)` `viajeros = (numeroDeViajeros )`	Evalúa las islas
`if (*generaciones*[posisla] % frecuencia == 0)` `migracion viajeros evaluacion posisla`	Envía viajeros llegado el momento

`(muta *varislas*[posislas] m))`
`(*generaciones*[posisla]++)`
`)`
	Genera mutaciones

El algoritmo de migraciones es el concepto que se crea en esta variación del genético, comunica cada isla con la siguiente como el esquema de anillo mostrado anteriormente. Cuando envías un individuo, expulsas el peor de la isla y en la isla origen generas otro individuo a partir de un cruce, la idea es mantener siempre el mismo número de invidividuos en todo momento.

`migracion (viajantes evaluacion posisla){`	

`tam = length (*varislas*[0])`
`evaluacionDestino (destinoIsla posisla)`
`evaluacionOrigen evaluación`
	Según la isla recalcula variables necesarias

`(if (viajantes !=0)`	Si hay viajantes

`for i from 0 to (viajantes-1) do`
`mejor = (first (nth i evaluacionOrigen))`
`peor =(first (nth i evaluacionDestino))`
`(setf (*varislas*[posilsa])`
`(remove mejor (nth posisla *varislas*)))`
`(push (cromosomaAleatorio (length mejor))`
`(nth posisla *varislas*))`
	Pone el mejor de una isla en otra al que le quitamos el peor, en hueco del origen creamos un cromosoma para mantener el tamaño de población

`(if (posisla !=3){`
`(setf (nth (+ i 1) *varislas*)`
`(subst mejor peor (nth (+ i 1)`
`*varislas*))`
`}else{`
`(setf (nth 0 *varislas*)`
`(subst mejor peor (nth 0`
`*varislas*))`
`}`
`}  }`
	La migración es cíclica, por lo que apuntamos a la siguiente. Si estamos en la última, a la primera

Para el resto de métodos mirar el fichero adjunto en lisp donde están el resto de métodos también comentados.

Ejemplos

El problema de coloración de mapas con el mapa de Andalucía. Hemos representado el grafo como una lista de pares (provincia, lista-vecinas). Los genomas como antes describimos vienen dados por un listado de 8 genes (uno por provincia) y el valor de cada gen deberá corresponder con un número de 1 a 8 para indicar el color.

Además de un ciclo de 6 nodos y el típico problema de K5.

Otros problemas

Para poder reutilizar el mismo algoritmo para otros problemas que no sean de coloración habría que adaptar la función objetivo que nos cualifica cuan buena es cada solución. El algoritmo en si deberá bastar, pues selecciona los individuos mas/menos cualificados para continuar evolucionando la población. También debería adaptarse la representación de los individuos.

Resultados y comparación

Problema	Población	Generaciones	Frecuencia	Tiempo (seg)	Resultado
Andalucía	12	10	2	1,4	6
	16	30	5	4,46	5
	20	100	5	12,083	5
	20	500	20	51,80	5
	20	2000	20	516.60	4
Ciclo 6	12	10	2	0,883	4
	16	100	5	6,25	3
	20	2000	20	255,583	2
K5	12	10	2	1,06	(-2)
	16	100	5	6,08	5
	20	2000	20	246,300	5

El tiempo utilizado en cada problema crece no de una manera totalmente lineal. Como observamos en el crecimiento del número de individuos y el número de generaciones a realizar se mantiene el crecimiento del tiempo al de población.

Tenemos que tener en cuenta que estas mediciones se hacen dentro de un sistema que no esta dedicado totalmente al cómputo y se generan interrupciones en el procesamiento. A mayor tiempo de computo, mayor número de interrupciones y mas desviado estará la medición.

Conforme dejemos más tiempo al algoritmo trabajar, irá optimizando el resultado. Comparándolo con el otro trabajo, la complejidad que tiene el hecho de mutar y tener diferentes individuos en una población le genera una gran dificultad. (Observemos que tabú solo trabaja con 1 individuo).

Como dato a destacar, si el número de iteraciones es demasiado pequeño, puede que no nos dé un número válido (como pasa en K5), que nos ha devuelto un total de 2 conflictos al final del procesamiento.

Dificultades y errores

- La dificultad de optimizar en el momento que alcanza un estado válido.
- La emigración de los individuos como representa el modelo de grislas, pues esta se hace dentro del propio algoritmo.
- El bajo desarrollo que tiene el entorno, tiene funciones muy básicas que dificultan la creación de métodos cuando crece en complejidad.
- Criterio de parada en un estado óptimo.
- El momento en el que pasamos de una población de 20 individuos nos genera un error.
- La idea de este algoritmo consiste en paralelizar cada isla. Como no tenemos desarrollado un sistema que nos permita la computación en paralelo hemos ingeniado una solución que comparte en medio de las iteraciones, dicha implementación hace que pierda eficiencia.

Salas Cantalejo, Alberto
Bellés Camacho, Víctor Juan

Bibliografía y referencias

GRISLAS: Un algoritmo genético paralelo que combina los modelos de grillas e islas para encontrar soluciones óptimas cercanas al problema del agente viajero - http://www.redalyc.org/articulo.oa?id=133117550002
El mismo link pero en pdf - http://www2.unalmed.edu.co/~pruebasminas/index.php?option=com_docman&task=doc_view&gid=431&tmpl=component&format=raw&Itemid=285

Artificial intelligence: a modern approach, Stuart Russel y Peter Norvig, páginas 126-129

Algoritmos genéticos, Natyhelem Gil Londoño - http://www.monografias.com/trabajos-pdf/algoritmos-geneticos/algoritmos-geneticos.pdf

Algoritmos genéticos, Jorge Arranz y Antonio Parra - http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf

Algoritmos genéticos, Universidad del País Vasco, Dpto. ciencias de computación e inteligencia artificial - http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf

Salas Cantalejo, Alberto
Bellés Camacho, Víctor Juan