Algoritmos Genéticos

D. Penazzi

Universidad Nacional de Córdoba

31 de marzo de 2021

Tabla de Contenidos

Problemas de optimización

- Hay problemas en los cuales hay que minimizar o maximizar alguna función.
- Pej, determinar $\chi(G)$, o calcular la distancia mínima entre dos vértices.
- Algunos de esos problemas, como el problema de calcular la distancia mínima, tienen algoritmos polinomiales que los resuelven, y luego en el curso veremos varios mas.
- Pero para otros problemas no se conocen algoritmos polinomiales que los resuelvan.
- Pej, el problema de calcular $\chi(G)$.
- Pero hay que resolverlos igual. Cómo hacemos?

Determinismo vs no determinismo

- En muchos casos tenemos que resignarnos a aceptar sólo una aproximación a la solución óptima.
- Muchas veces con eso basta, pero incluso asi a veces no hay algoritmos polinomiales de aproximación.
- Ahora bien, cuando hablaba de “algoritmos", en realidad quería decir “algoritmos determinísticos.”
- En algunos casos, si los algoritmos determinísticos fallan, se pueden usar algoritmos no determinísticos
- Estos algoritmos toman en ciertas partes decisiones aleatorias para poder avanzar.

Exploración Versus Explotación (del conocimiento)

- Hay dos tipos básicos de estrategia de búsqueda:
- (Algunos algoritmos mezclan un poco de ambas estrategias)
  - Busqueda “ciega":
    - Explora en forma aleatoria lo mas posible el espacio de posibles soluciones, y se queda con la mejor.
  - Busqueda heurística:
    - Explota conocimiento sobre el problema, y busca en forma dirigida para tratar de mejorar la solución que se tiene.

Exploración Versus Explotación (del conocimiento)

- Respecto de explorar:
  - Si las soluciones óptimas o soluciones aproximadas a las óptimas no son densas, no las podrá encontrar.
  - Un ejemplo para coloreo seria ordenar los vertices al azar y usar Greedy.
  - En algunos casos especificos de grafos esto puede funcionar, pero dado que hay $n!$ ordenes posibles, si la solución óptima involucra un orden muy particular es muy poco probable que la encontremos asi.
- Respecto de explotar conocimiento:
  - Requiere conocer cosas especificas del problema para explotarlas.
  - Problema de los mínimos/máximos locales:
  - si se esta cerca de uno, la busqueda encontrará ese en vez del mínimo/máximo global

Algunos algoritmos

Hill Climbing

- Se cambia algo de la solución y se testea a ver si la nueva solución es mejor.
- Si lo es, nos quedamos con esa y seguimos buscando a partir de ella.
- Si no, nos quedamos con la solución original y seguimos buscando desde ella.

Simulated Annealing Parecido a Hill Climbing:

- Tambien cambiamos algo al azar y tambien, si la nueva solución es mejor, se acepta.
- Pero si es peor se rechaza sólo con una cierta probabilidad.
- Esa probabilidad de rechazo al principio es baja y luego va aumentando.

Varias soluciones

- La idea de simmulatead annealing es que al permitir explorar algunas soluciones que no necesariamente mejoran la situación, permite “salir" de la cercania de un máximo/mínimo local al cual se haya llegado demasiado tempranamente.
- En vez de buscar una sola vez una posible solución óptima, podriamos hacerlo varias veces comenzando desde puntos distintos, o desde el mismo punto pero tomando distintas decisiones aleatorias.
- Tambien podriamos hacer eso con varias soluciones en paralelo.

Multiples posibilidades en paralelo

- Si lo hacemos en paralelo, se abre una posibilidad interesante:
  - Podriamos hacer que las posibles soluciones se comuniquen entre si.
  - Para que puedan “aprender”de las otras.
- Esas son dos carácterísticas de los algoritmos genéticos:
  1. No trabajan sobre una SOLA posible solución, sino sobre una población de posibles soluciones.
  2. Hay interacción entre esas distintas posibles soluciones.

Poblaciones

- Los algoritmos genéticos tratan de imitar el proceso de selección natural.
- Al igual que lo que ocurre en la naturaleza, usan operadores de:
  1. *mutación*
  2. *crossover* (cruzamiento)
  3. *selección*

Mutación, Crossover y Selección

Mutación Una solución (individuo) es cambiada al azar.

- Es el operador mas simple de los tres.
- La probabilidad de que ocurra una mutación suele ser baja.
- Regulando esa probabilidad regula que tanto explora el algoritmo (versus explotar lo que ya se tiene)

Crossover Se mezclan dos soluciones para obtener dos nuevas soluciones, imitando el crossover de cromosomas de la naturaleza.

Selección Se eligen a quienes se les va a aplicar el operador de Crossover. (“reproducción").

Crossover

- En la naturaleza los genes se agrupan en cromosomas. Por ejemplo en humanos hay 46 cromosomas.
- En la mayoria de las especies los cromosomas se agrupan en pares: en humanos tenemos cromosomas 1a y 1b, 2a y 2b, ..., 23a y 23b.
- En la reproducción se combinan partes de 1a y 1b “cortandolos" en algún lado y juntando la primera parte de 1a con la segunda de 1b y (en otro nuevo cromosoma) la primera parte de 1b con la segunda de 1a y lo mismo con 2a y 2b, etc.
- Nota: los humanos somos diploides. Los AG trabajan como si fueramos haploides.

Selección

- En la naturaleza está determinado con cual cromosoma se hará el crossover, en los AG el operador de selección se encarga de eso
- La selección se usa usando una función de fitness (adaptación) que debe ser tal que mayor adaptación implique mejor solución.
- (por ejemplo, mientras menos colores tenga un coloreo, mas “fit" será).

Selección

- La selección no es determinística, sino probabilística:
  - Individuos mas “fit" tendrán mas probabilidades de ser seleccionados, pero no certeza.
  - Individuos menos “fit" tendrán menos probabilidad de ser seleccionados, pero no imposibilidad.
- Individuos pueden ser seleccionados mas de una vez.
- Los individuos mas “fit" serán (probablemente) seleccionados mas veces que los menos fit, y algunos de los menos fit no serán seleccionados nunca.

Esquema de un AG

- Construir al azar Poblacion P(0) y tomar t=0
- While(no se satisfagan condiciones para parar)
  - Aplicar selección a P(t) para determinar quienes se usarán para el crossover
  - Aplicar crossover a los seleccionados
  - Mutar algunos de ellos
  - P(t+1)= la nueva población asi construida.
  - t=t+1
- EndWhile
- Return el mejor individuo de P(t)

- Los Operadores de Selección y Crossover hacen una busqueda mediante la estrategia de explotación de ganancias ya obtenidas.
- Pueden provocar una dominación de la población por individuos mas fit conduciendo a un mínimo/máximo local.
- El operador de Mutación es el que se encarga de producir individuos que puedan romper un temprano acercamiento a un mínimo/máximo local.
- Crossover debe ser tal que produzca nuevos cromosomas pero mantenga intáctos algunos "genes".
- Un “gen", si bien no está definido siempre de la misma forma en la literatura, en general es un fragmento de cromosoma lo suficientemente estable como para hacer sentir su presencia en el fenotipo a través de la evolución.

- Recordemos que la selección natural funciona porque son los genes los que estan “luchando" para replicarse: la aparente lucha de individuos y especies es un side effect.
- Pero estamos interesados en ese “side effect"
- asi que debemos estar seguros que el efecto primario se mantiene:
- si no hay un número suficiente de genes que se mantengan en el crossover, un AG degenera en busqueda aleatoria ciega.

Otras componentes de un AG

- Hay que decidir cómo se codificarán las soluciones del problema, es decir, que estructura tienen los “cromosomas"
- Hay que decidir cómo será la función de adaptación.
- Hay que decidir cuanta será la “presión evolutiva"
  - Si es muy alta (se penaliza mucho los menos fit y se premian mucho los mas fit) entonces puede producirse que un número reducido de individuos dominen la población
  - Si es muy baja, los genes “buenos" no tienen estádisticamente chances de hacerse sentir.

Ejemplos de Cromosomas

Un cromosoma puede ser:

- Una string de bits.
- Una string de números, ya sea enteros o de punto flotante.
- En estos casos (enteros o punto flotante) pueden ser arbitrarios o estar sujetos a condiciones de contorno. (pej, estar entre -2 y 4)
- Una permutación de elementos. (“order based codification")
- Esta última se usa mucho por ejemplo en el problema del viajante: recorrer n ciudades distintas visitando una sola vez a cada una, de forma tal que el costo total de transporte sea mínimo.

Ejemplos de Operadores de Mutación

- En el caso de una string de bits, enteros o flotantes, un típico operador de Mutación toma una entrada al azar y la cambia aleatoriamente.
  - En el caso de bits, lo cambia de 1 a 0 o 0 a 1, pero en el caso de enteros o flotantes el tamaño del cambio también puede ser aleatorio
  - En casos con condiciones de contorno se puede usar un operador que lo “tire contra la pared".
  - Ejemplo, si las entradas deben estar entre -100 y 100:
  - $(4, -15, 100, 5, 9, -33)$
  - Esto sirve para asegurarse de explorar las regiones cerca de la frontera, que pueden no ser exploradas si no se hace esto.
- Si estamos en el caso de un order based codification, se permutan dos elementos al azar.

Ejemplos de Operadores de Crossover

Single Point Crossover Mimíca lo que ocurre realmente en la naturaleza: si los progenitores son P y Q, se cortan ambos en un mismo punto y los hijos son la primera parte de P con la segunda de Q y viceversa.

- P=ACGT|FRGADE H1=ACGTKERABC
- Q=BEFC|KERABC H2=BEFCFRGADE

Two Point Crossover Con dos puntos de corte y alternando

- P=ACGT|FRG|ADE H1=ACGTKERADE
- Q=BEFC|KER|ABC H2=BEFCFRGABC

Multiple Point Crossover Muchos puntos de corte.

Ejemplos de Crossover (cont)

Máscara Usando máscara aleatoria de bits, el primer hijo tiene las entradas de P que correspondan con los 1s de la máscara y las entradas de Q que correspondan con los 0s. (el otro al revés).

- Los de Multiple Point y Máscara, si bien mesclan mas eficientemente, pueden destruir mas genes.
- Obvio que mientras mas chicos los genes, mas probabilidad de sobrevivir tiene, pero mientras mas grande mas probable es que ejerza algúna influencia determinante en el fenotipo
- Si se produce demasiada mezcla, lo único que se puede llegar a considerar como “gen" serán entradas individuales, bordeando peligrosamente la busqueda ciega.

Crossover en los casos de Order Based codification

- En este caso no podemos simplemente COPIAR los fragmentos en forma alternada, pues lo que quede probablemente no sea una permutación. Por ejemplo, lo siguiente NO SIRVE:
- P=ABCD|EFGHIJK H1=ABCDKJIDGAH
- Q=BEFC|KJIDGAH H2=BEFCEFGHIJK
- Y vemos que ni H1 ni H2 son permutaciones.

Crossover en los casos de Order Based codification

- Una forma es copiar un fragmento, y en el otro fragmento, copiar los elementos que faltan en el orden en el cual estan en el otro progenitor.
- P=ABCDEFGHIJK H1=ABCDEFKJIGH
- Q= BEFC|KJIDGAH H2=BEFCADGHIJK
- Como antes, puede haber single point, two point, muiltiple point o crossover con máscaras

PMX

- Otra forma se llama Partial Mixing Crossover.
- Intercambiamos el o los bloques que se eligan, y en el resto, si hay duplicados con elementos de adentro del bloque, se intercambian con otro elemento de forma que no haya duplicación.
- El elemento elegido para el intercambio se deduce de la biyección dada por el segmento intercambiado
- H1=ABFCKJIDGHE
- H2=BKCDEFGHIAJ
- F C K J I D E-K G-I
- C D E F G H H-D-C-F-J –> H-J
- Como antes, puede haber single point, two point, muiltiple point o crossover con máscaras

Cyclic Crossover

- Intercambiar dos elementos en la misma posición.
- Esto provocará una perdida de la inyectividad en ambos cromosomas
- Ir reemplazando las repeticiones en forma cíclica.
- H1=BECDKFGHIJA
- H2=ABFCEJIDGHK

Valores Esperados de Reproducción

- Como dijimos, en los algoritmos genéticos es fundamental la función de adaptabilidad (“fitness function").
- Cada individuo tendrá una fitness F_i dada por esa función.
- Independientemente de cómo es la función de fitness o que valores toma, se suele normalizar su valor usando los valores esperados de reproducción, o “esperanzas".
- Lo mas usual es tomar

$$E_i = \frac{F_i}{\overline{F}} \quad \text{donde } \overline{F} = \frac{\sum F_i}{n}$$

Ruleta

- Los E_i se usan para seleccionar aleatoriamente quienes se reproducirán.
- Uno de los métodos mas usados es el de Ruleta.
- Se calculan $S_j = \sum_{i \leq j} E_i$.
- Se tiran n números al azar r_k entre 0 y n.
- Para cada k, se selecciona para reproducirse el primer j tal que $S_j \geq r_k$.

Stochastic Universal Sampling (SUS)

- El sistema SUS es igual que el de ruleta, excepto que en vez de seleccionar n números al azar entre 0 y n, se seleccionan de la siguiente manera:
  - Tomar un número r al azar entre 0 y 1.
  - Elegir al azar una permutación σ de $\{0, 1, ..., n-1\}$.
  - Tomar $r_k = \sigma(k) + r$, $k = 0, ..., n-1$
- La ventaja de SUS sobre ruleta es que el individuo i tiene garantizado que se reproducirá al menos $\lfloor E_i \rfloor$ veces.

Métodos con remainder

- Otra forma de garantizar que el individuo i se reproduzca al menos $\lfloor E_i \rfloor$ veces es con los métodos de remainder (resto).
- A cada individuo i se le asignan inicialmente $\lfloor E_i \rfloor$ “tokens" de reproducción.
- Se calcula $m = n - \sum \lfloor E_i \rfloor$. Si $m = 0$ ya terminamos.
- Si $m > 0$, se distribuyen esos m tokens faltantes usando algún sistema (por ejemplo ruleta) tomando en vez de las esperanzas, los restos $R_i = E_i - \lfloor E_i \rfloor$.
- Luego de seleccionados los individuos, se deben mezclar al azar para determinar las parejas.

Convergencia temprana

- Un par de técnicas que se usan para prevenir esto son:
  1. Ranking
  2. Sigma scaling
- Veamos un poco de cada una:
- En Ranking, no importa el valor de F_i sino sólo el orden entre ellas.
- El individuo que esté mas arriba en el ranking tendra un cierto % mas de probabilidad de reproducirse que el segundo, este mas que el tercero, etc,
- pero sólo depende de esa posición y no de cuanto “mejor" es.

Ejemplo de función usada para ranking

- Si se ordenan los individuos de menor a mayor, una formula común es:
- $$LP(p) = 2 - SP + 2(SP - 1).\frac{p-1}{n-1}$$
- donde SP es un parámetro entre 1 y 2 y p es la posición en el ranking.
- SP es la abreviación de “selection pressure”.
- Pej, veamos que pasa en los extremos:

Ejemplo de función usada para ranking

- Si $SP = 1$ la formula queda $LP(p) = 1 + 2(1-1).\frac{p-1}{n-1} = 1$, es decir, no hay ninguna presión: todos son seleccionados con la misma probabilidad.
- Si $SP = 2$ la formula queda $LP(p) = 2.\frac{p-1}{n-1}$.
- Pej con 11 invididuos quedaria $LP(1) = 0$, $LP(2) = 0,2$, $LP(3) = 0,4$,$LP(4) = 0,6$, ...etc
- Regulando SP podemos variar esta presión.

Sigma Scaling

- En Sigma scaling se cambia cómo se calculan las esperanzas de acuerdo con la variación de las fitnesses.
- Una formula muy usada es $E_i = 1 + \frac{F_i - \overline{F}}{2\sigma}$
- Esta forma tiene la ventaja de un “ajuste automático":
- Si la variación de la fitness de la población es grande (σ grande) entonces los individuos mas extremos no influyen tanto y se limita la posibilidad de convergencia temprana.
- Si el tiempo ha pasado y la variación es poca, entonces ahi queremos que si aparece un individuo mas fit que otros tenga mas “influencia", y como el σ será mas chico, entonces lo hará.

Otras técnicas

- Otras técnicas usadas son:
  - Luego de una cierta cantidad de generaciones, o bien si se detecta poca variabilidad de las fitness, provocar una “catástrofe":
    - quedarse con algunos los individuos mas fit, eliminar al resto y generar nuevos individuos al azar para reemplazarlos.
  - “Islas”:
    - Tener poblaciones de poblaciones, aisladas entre si (islas) pero permitir “migraciones”entre islas luego de un cierto tiempo.
- Hay muchas otras cosas para ver sobre esto, pero al ser sólo un pantallazo general, ya es suficiente.