Metaheurística GRASP

a) Explicación del algoritmo

El algoritmo básicamente itera una cantidad de veces, en principio desconocida, y aplica primero la heurística golosa aleatorizada (con distintos valores de alfa). Una vez que tiene una solución factible generada por la heurística golosa aleatorizada, aplica un algoritmo heurístico de busqueda local sobre ella, obteniendo una solución nueva, mejor o igual (nunca peor). Como hay dos algoritmos distintos de búsqueda local (utilizan distintas vecindades), GRASP siempre va a elegir el mismo durante una ejecución y va a estar determinado por un parámetro de entrada.
Otra cosa configurable en GRASP es quienes van a formar parte de la RCL. Considerando lo que explicamos sobre el algoritmo goloso y sus distintas decisiones y criterios, en realidad existen tres listas restrictas de candidatos:

- La que está conformada por los pares de nodos candidatos a ser la arista "máxima" que tome el algoritmo en la etapa inicial.
- La que dado un nodo u está conformada por los posibles pares de u (ver definición de Par en la explicación de la heurística golosa).
- La que a la hora de ubicar un nodo o un par de nodos dentro de la partición P, contiene los pares de posiciones que son candidatos a "la mejor manera" de ubicar los nodos.

Estas listas de candidatos son construidas en base a tres parámetros, uno para cada una, $alfa$, $beta$, $gamma$. Cada uno de esos parámetros representa un porcentaje, que indica que tan distantes pueden estar del valor óptimo los valores contenidos en cada lista.

La cantidad de veces que itera el algoritmo va a estar relacionado con la cantidad de mejoras que hubo en las últimas iteraciones. Se trata de un valor entero configurable (otro parámetro, aca vamos a llamarlo $delta$), de manera tal que representa una cantidad de iteraciones. Si se hizo esa cantidad de iteraciones de manera seguida y no se observó ninguna mejora en la solución con la que se está trabajando, entonces el algoritmo termina.
La solución devuelta por GRASP siempre es la mejor encontrada hasta el momento, esto significa que se va almacenando y se pisa cuando se encuentra una solución mejor.

a) Experimentación

Las instancias que generamos para realizar la experimentación de la calidad de las soluciones y la performance de GRASP, fueron grafos completos. Al igual que como experimentamos con otros algoritmos, los pesos de las aristas de los grafos completos se determinaron de manera aleatoria en un rango de 0 a 50.
Todas las mediciones que presentaremos a continuación se realizaron sobre el mismo conjunto de instancias, el cual tiene 100 grafos completos con vértices de 1 a 90.
El objetivo de esta experimentación es poder encontrar una configuración conveniente de los parámetros de GRASP y ver si se adapta a otros grafos que no necesariamente sean completos.

Vamos a presentar gráficos comparativos sobre la performance y la calidad de las soluciones que nos brinda GRASP con distintas configuraciones. Cuando hablemos de Vecindad 1 y Vecindad 2, vamos a estar haciendo referencia al primer y segundo algoritmo de búsqueda local que presentamos.

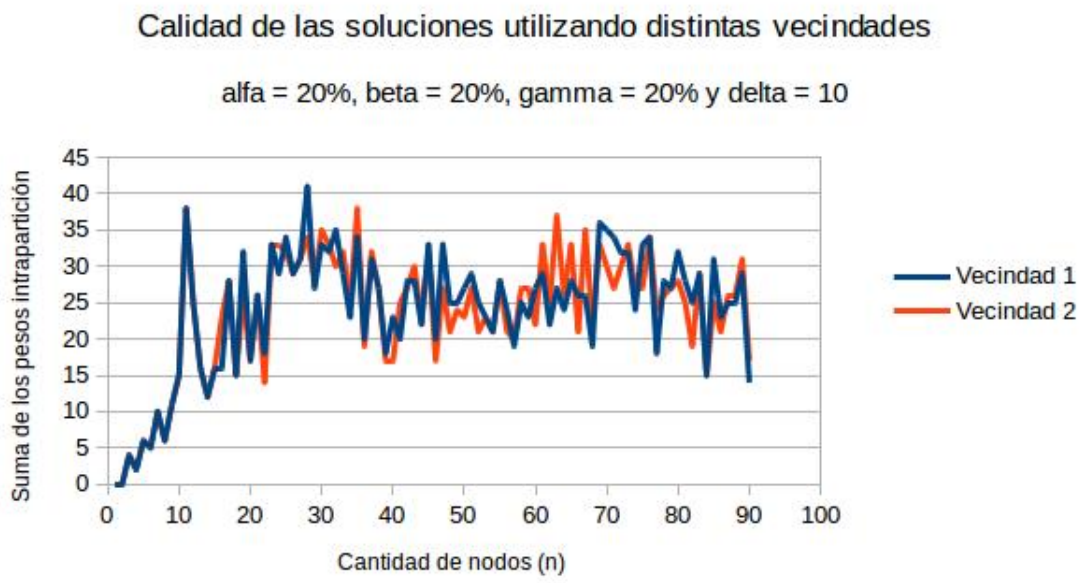

Figure 1: En este experimento los porcentajes que determinan las RCL se mantienen bajos. También, delta es un número relativamente bajo. Podemos observar que la calidad de las soluciones que da GRASP utilizando una vecindad u otra, es similar. Solo en el intervalo de $n \in [60..70]$ se puede ver que las soluciones cuando se utiliza la vecindad 1 son mejores, pero no es un rango muy significativo.

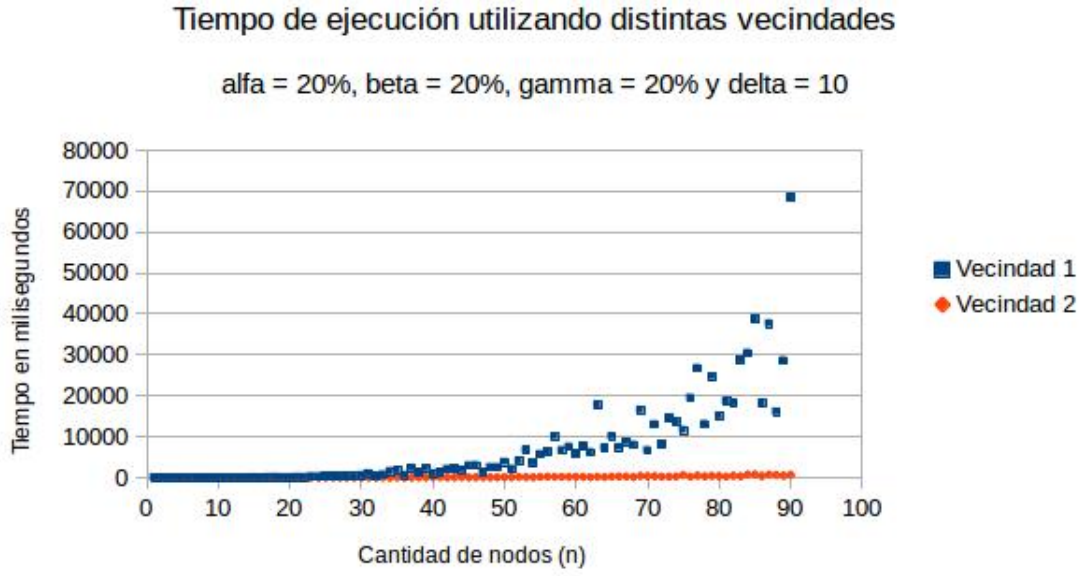

Figure 2: En este gráfico, podemos contemplar una diferencia significativa entre el tiempo de ejecución de GRASP utilizando una vecindad y la otra. Se puede ver incluso como esa diferencia tiende a hacerse más grande asintóticamente.

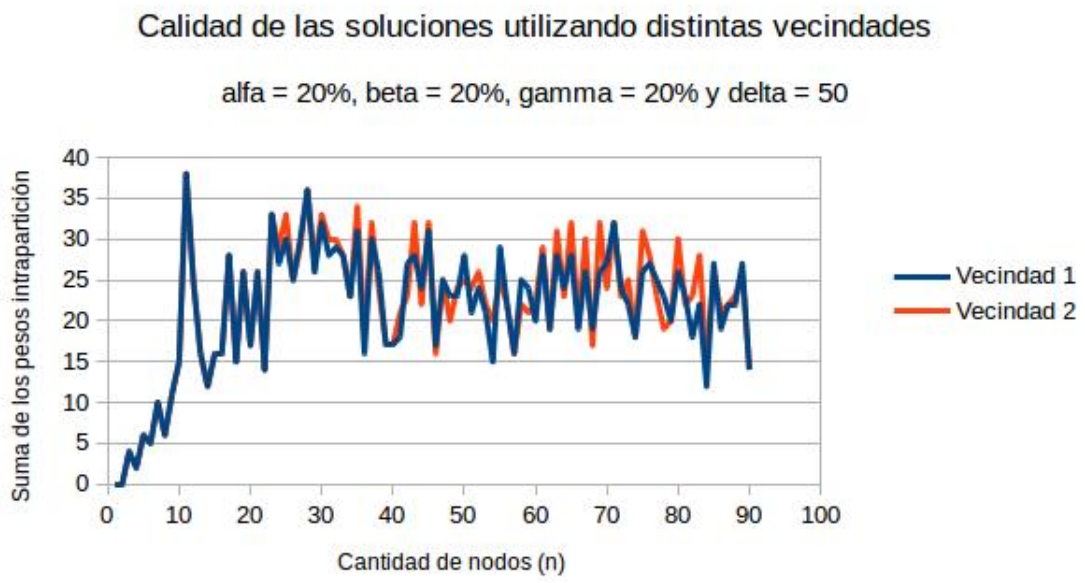

Figure 3: En este caso decidimos aumentar delta. Intuitivamente podríamos creer que el hecho de aumentar esto nos puede llegar a dar soluciones mejores, debido a que aumentan las chances de encontrar una mejor solución. Sin embargo, esto no paso en este caso e incluso algunas soluciones fueron peores. Esto se debe a que hay otros factores en juego además del delta (como el alfa, beta y gamma) y, como está todo aleatorizado, puede suceder que tratando de mejorar nuestras soluciones aumentando el delta, al correr nuevamente el algoritmo tengamos mala suerte y la aleatoriedad nos juegue en contra. Una vez más se puede observar que la calidad de las soluciones haciendo uso de la vecindad 1 o la vecindad 2, es similar.

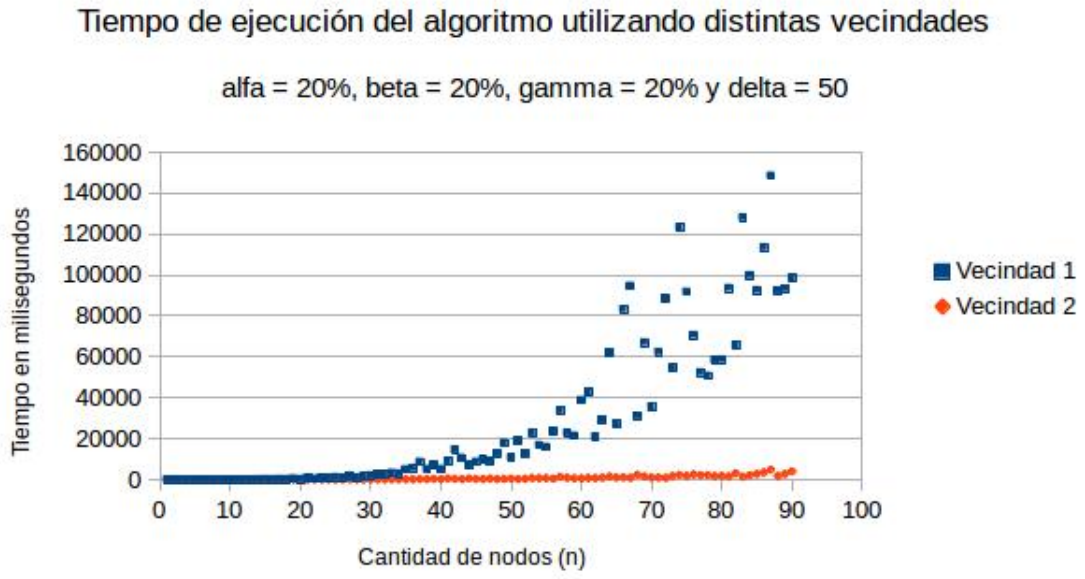

Figure 4: Podemos observar también como el tiempo de ejecución del algoritmo al aumentar ese parámetro, se incrementa bastante. Si bien no es del todo visible con la vecindad 2 porque el tiempo que tarda GRASP al hacer uso de ella es despreciable con respecto al que tarda cuando usa la vecindad 1, el tiempo igualmente aumenta porque, de una forma u otra estamos haciendo más iteraciones, independientemente de la vecindad a utilizar.

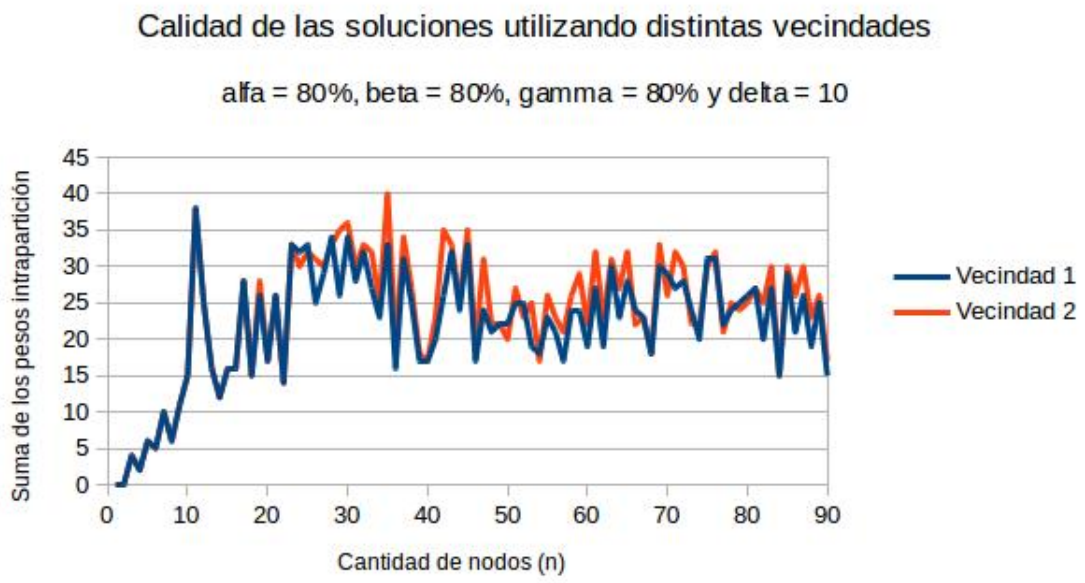

Figure 5: Ahora, probamos aumentar bastante los parámetros alfa, beta y gamma. Lo que produce esto es aumentar el tamaño de nuestras listas de candidatos (ya que ahora podrán estar dentro de ellas los que cumplan estar hasta un 80% lejos del mejor). Vemos que las soluciones que provee GRASP utilizando una vecindad u otra, siguen siendo similares en cuanto a calidad. Empezamos a pensar que esto es independiente de los valores de alfa, beta y gamma.

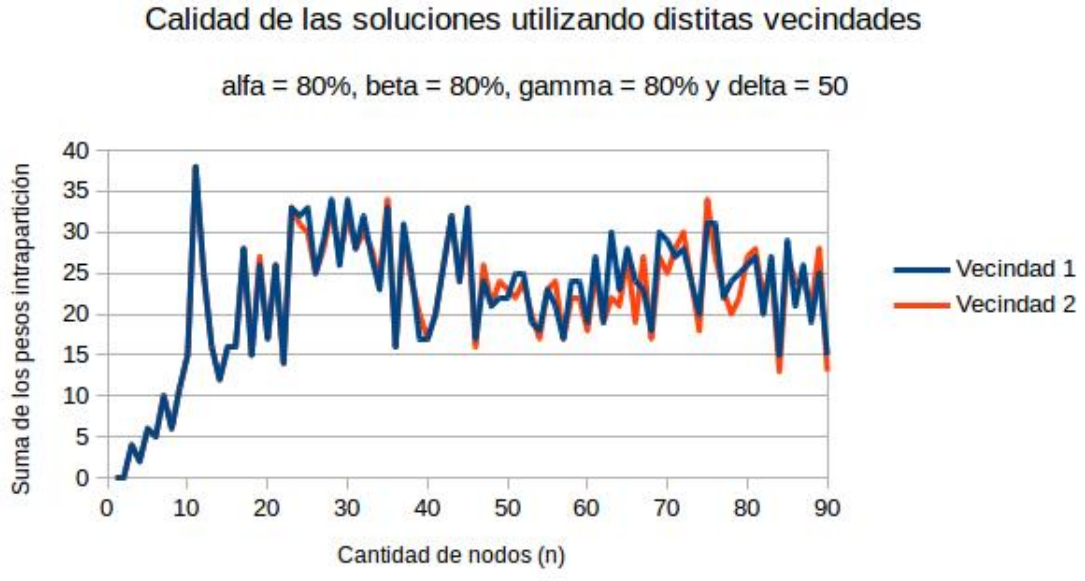

Figure 6: Una vez más, aumentamos el delta y vemos que, con ninguna de las dos vecindades hay una mejora realmente notable, la cual justifique el costo necesario para realizar este cambio.

Dado que las soluciones que provee GRASP haciendo uso de cualquiera de las dos vecindades son muy similares en cuanto a calidad, al parecer lo que nos conviene hacer es utilizar la vecindad 2. Esto es así debido a que las soluciones que produce GRASP con la vecindad 2 son muy similares a cuando utiliza la vecindad 1, pero con un tiempo de ejecución muchísimo menor.

Como los experimentos realizados tienen siempre el mismo valor para alfa, beta y gamma, sería interesante ver si las soluciones que provee GRASP utilizando valores más distantes (y por ende distintos) de alfa, beta y gamma, siguen siendo similares con una vecindad u otra, en términos de calidad.
Para ver esto realizamos una última experimentación, con distintos valores de alfa, beta y gamma:

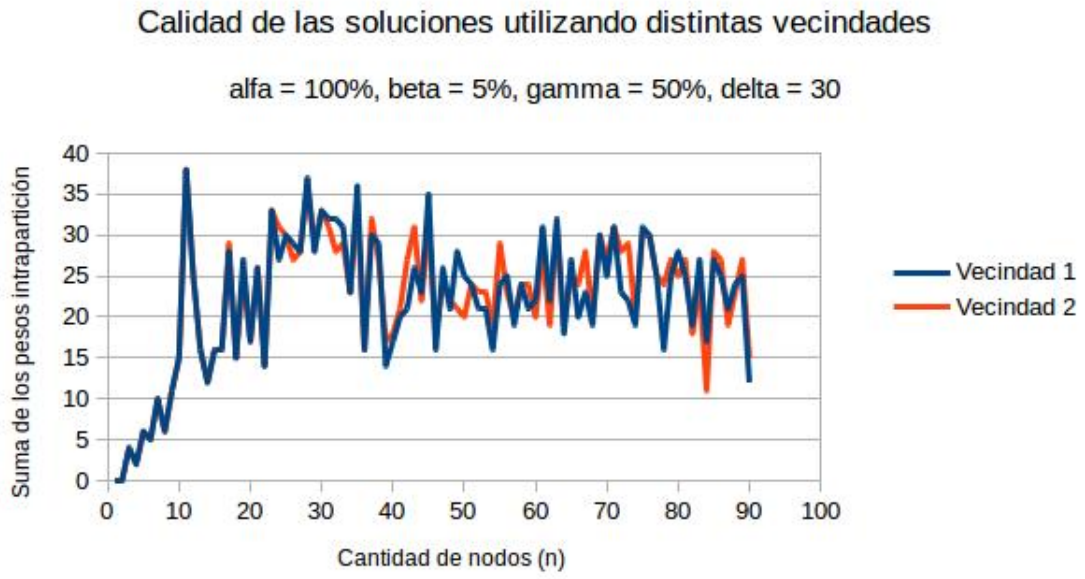

Figure 7: Vemos que, salvo casos muy particulares, las soluciones con una vecindad u otra siguen siendo similares a pesar de que alfa, beta y gamma ahora no comparten el mismo valor (y son notablemente distintos).

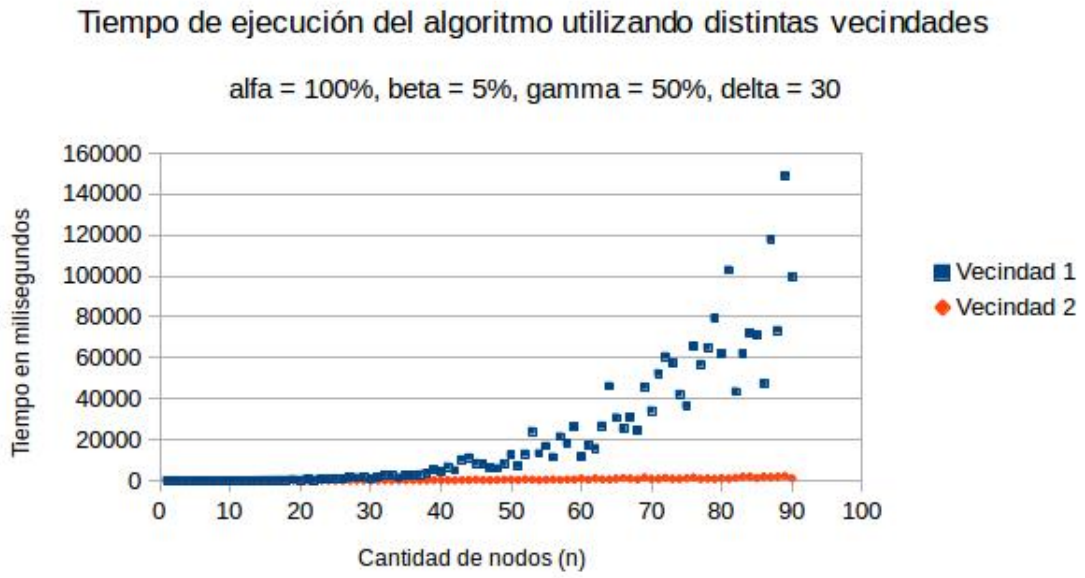

Figure 8: Acompañamos al gráfico anterior, con éste que nos provee alguna información sobre los tiempos de ejecución con los valores anteriores. Se puede ver que los tiempos son similares a cuando alfa = 80%, beta = 80%, gamma = 80% y delta = 50.

Finalmente, podemos concluir que una buena configuración de los parámetros de GRASP puede ser tomar valores de alfa, beta y gamma relativamente bajos (20 puede ser un buen número según nuestros gráficos), ya que al parecer, no tiene mucho sentido agrandar demasiado el tamaño de nuestras RCL's si lo que buscamos es mejorar la calidad de las soluciones. Por otra parte, el valor del delta tampoco necesita ser grande (10 podría ser un valor razonable según el primer gráfico). Finalmente, es claro que es conveniente utilizar la vecindad 2, ya que solo se poseen dos vecindades distintas y GRASP con cualquiera de ellas arroja soluciones similares a las que daría si utilizara la otra, con la diferencia de un tiempo de ejecución significativamente menor.