UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERIA DE SISTEMAS – INVESTIGACION DE OPERACIONES

UN MODELO PARA EL PROBLEMA DE LA DIVERSIDAD MÁXIMA
UN ESTUDIO PARA LA SELECCIÓN DE LO MEJOR Y LO MÁS DIVERSO

TESIS
QUE PARA OPTAR POR EL GRADO DE:
DOCTOR EN INGENIERÍA

PRESENTA:
FERNANDO SANDOYA SÁNCHEZ

TUTOR PRINCIPAL
RICARDO ACEVES GARCÍA, FACULTAD DE INGENIERÍA
COMITÉ TUTOR
MAYRA ELIZONDO CORTÉS, FACULTAD DE INGENIERIA
RAFAEL MARTI CUNQUERO, UNIVERSIDAD DE VALENCIA

MÉXICO, D. F. OCTUBRE 2013

JURADO ASIGNADO:

Presidente:	IDALIA FLORES DE LA MOTA
Secretario:	MAYRA ELIZONDO CORTÉS
Vocal:	RICARDO ACEVES GARCÍA
1 er. Suplente:	MANUEL ORDORICA MELLADO
2 d o. Suplente:	RAFAEL MARTÍ CUNQUERO

México D.F., México

TUTOR DE TESIS:

RICARDO ACEVES GARCÍA

FIRMA

DEDICATORIA

Dedicado a todos los seres queridos con los que

nunca se pasa el tiempo…

sino más bien siempre se lo disfruta:

mis hijos Fernando, Ariadna y Nathalie,

mi esposa Talia, mis padres y mis hermanos,

sin el aliento de ellos no hubiera sido posible

realizar este trabajo con éxito.

AGRADECIMIENTOS

Una tesis doctoral es un trabajo arduo que implica sacrificios y que forma al individuo no solo académicamente, sino que fortalece el intelecto, y modifica positivamente la forma de ver el mundo, pero también es un proceso que involucra el trabajo previo y la colaboración de varias personas, por tanto esta etapa no puede verse completada exitosamente si no hubiera la guía y colaboración de expertos del área, los cuales teniendo al tiempo como uno de sus mayores bienes, no dudaron en dedicar parte de él al desarrollo de esta investigación. Por ello mi agradecimiento a estos profesionales, y excelentes amigos también, por su continuo apoyo.

Sobre todo agradezco la colaboración de Rafael Martí, un científico de primera línea que lidera la investigación en metaheurísticas y optimización combinatoria, de Ricardo Aceves por la tutoría que ha realizado a este trabajo, por sus acertadas críticas y sugerencias

También agradezco a la Escuela Superior Politécnica del Litoral, de cuya planta docente me enorgullezco de pertenecer, por el apoyo institucional permanente que me proporcionó para la realización de estos estudios doctorales.

RESUMEN

Seleccionar elementos diversos es una bastante tarea común en muchos problemas de la vida real. El problema de la diversidad máxima (MDP por sus siglas en inglés) consiste tradicionalmente en la selección de un subconjunto de elementos de un conjunto dado, de tal manera que una medida de la diversidad presente en el subconjunto de los elementos seleccionados sea maximizada. En la literatura se reportan algunas variantes de este problema, de acuerdo con la función de diversidad considerada, en particular, en esta investigación se resuelve un nuevo problema, denominado Max-mean, que maximiza la diversidad promedio entre los elementos del subconjunto seleccionado, y en el que el número de elementos seleccionados es también una variable de decisión. Se muestra que la búsqueda de su solución óptima es un problema fuertemente NP-duro. Por otro lado, por las características de las distancias entre los elementos considerados en el nuevo modelo, tiene aplicaciones en la selección de los grupos de personas representativas de la población, o en la selección de un grupo de instituciones, etc. En la primera parte, se propone este nuevo modelo, se estudian algunas de sus propiedades, y se propone un algoritmo muy eficiente para resolverlo sobre la base de una hibridación de metaheurísticas, para estimar la eficiencia del método propuesto se adaptan los mejores algoritmos que se reportan en la literatura para resolver problemas similares. En la segunda parte de esta investigación se extiende el problema de optimización considerado, agregando una restricción adicional: se busca también seleccionar los mejores elementos, los mejores de acuerdo a algún criterio. En este contexto, el problema de optimización se convierte en un problema clásico de optimización multiobjetivo. Con técnicas de procesamiento simples este problema puede ser transformado en el problema uniobjetivo anterior. De esta manera, se resuelve el problema general de la selección de lo mejor y lo más diverso.

ABSTRACT

Diversity selection is a common task in multiple real-world problems. The maximum diversity problem traditionally consists in selecting a subset of items from a given set, such that a measure of the diversity present in the subset of selected elements is maximized. In literature are reported several variants of this problem, according to the diversity function considered, in this investigation it solves a new problem, called the Max-mean model, which maximizes the average diversity between elements in the subset selected, and in on which the number of elements selected is also a decision variable, we show that finding an optimal solution to this problem is strongly NP-hard. The characteristics of the inter-element distances considered in the new model has applications in the selection of groups of people representative of a population, or in the selection of a group of institutions, etc. In the first part of this research, this new model is proposed, we study some of its properties, and is designed a highly efficient algorithm to solve it, based on a hybridization of metaheuristics with good performance in other similar combinatorial problems, to estimate the efficiency of the proposed method are adapted the best algorithms to solve similar problems, as reported in literature. In the second part of this research extends the optimization problem considered, an additional constraint is added: want also select the best items, the best according to some criterion. In this context, with this additional objective, the optimization problem becomes a classical multiobjective optimization problem. With simple processing techniques this problem can be transformed into the problem uniobjetivo above, for which we developed a very efficient method. This will solve the general problem of selecting the best and most diverse.

ÍNDICE GENERAL

ÍNDICE GENERAL 1
ÍNDICE DE FIGURAS 3
ÍNDICE DE TABLAS: 5
Introducción 6
Objetivo general y objetivos específicos de la investigación 16
Metodología 17
Capítulo 1.
Distancias, similitud y diversidad 22
1.1. Definiciones 22
1.2. Medidas de similitud 24
1.3. Equidad, diversidad y dispersión 27
Capítulo 2.
El Problema de la Diversidad Máxima 41
2.1. Formulaciones y Modelos de Programación Matemática 43
2.2. Complejidad Computacional 49
2.3. Correlación entre los problemas 51
2.4. Propiedades del Problema del Máximo Promedio Max-Mean 57
Capítulo 3.
Procedimientos Metaheurísticos en Optimización Combinatoria 60
3.1. Métodos heurísticos simples 60
3.2. Indicadores de calidad de un algoritmo heurístico 62
3.3. Clasificación de los métodos heurísticos: constructivos y de búsqueda local 64
3.4. Métodos Combinados 66
3.5. Metaheurísticas 71
Capítulo 4.
Diseño de heurísticas eficientes para resolver el problema Max-Mean 79
4.1. Relación entre el problema Max-Sum y el problema Max-Mean 80
4.2. Diseño de Heurísticas previas adaptadas al problema Max-Mean 91

4.3. Heurística GRASP3 98
Capítulo 5.
Resultados numéricos con los problemas de prueba 110
5.1. Calibración del parámetro α 111
5.2. Rendimiento de las heurísticas en problemas pequeños 113
5.3. Solución de problemas medianos con GRASP1, GRASP2 y GRASP3. 116
5.4. GRASP3 con Rencadenamiento de Trayectorias para problemas grandes 122
5.5. Comparación de la heurística GRASP3 con un Optimizador de caja negra 129
5.6. Resumen de los métodos desarrollados 133
Capítulo 6.
Un Caso de aplicación para el problema Max-Mean 135
6.1. Los grupos de personas más diversos son más eficientes 135
6.2. Diversidad en identidad y diversidad funcional, perspectivas y heurísticas 137
6.3. Cómo seleccionar el equipo de trabajo más productivo 138
6.4. Hipótesis básicas y relación entre habilidad y diversidad 140
6.5. Resolución de un caso 145
6.6. Extensión al problema de seleccionar equipos de trabajo 147
Capítulo 7.
La selección del conjunto más diverso y mejor como un problema Multiobjetivo 152
7.1. Problemas de optimización con objetivos múltiples 152
7.2. Transformación del Problema Multiobjetivo en uno del tipo Max-Mean 162
7.3. Resultados numéricos con los problemas de prueba 165
CONCLUSIONES 170
REFERENCIAS 172
ANEXO 1. Tablas con los resultados detallados de los experimentos numéricos 176
ANEXO 2. Conceptos y Demostraciones 188

ÍNDICE DE FIGURAS

FIGURA 1. ¿CÓMO SELECCIONAR 10 PERSONAS DE UN GRUPO DE 49 PERSONAS, CON DIVERSIDAD MAXIMA? 9
FIGURA 2. DIAGRAMA DE FLUJO DE LA METODOLOGÍA DE LA INVESTIGACIÓN 21
FIGURA 3. LOS VECTORES REPRESENTAN LAS CARACTERÍSTICAS DE DOS INDIVIDUOS EN TRES CASOS, EL ÁNGULO INDICA LA MEDIDA DE SIMILITUD ENTRE ELLOS 25
FIGURA 4: OCHO ELEMENTOS SELECCIONADOS DE UN CONJUNTO DE 25 CON EL MODELO MAX-SUM Y CON EL MODELO MAX-MIN 29
FIGURA 5: OCHO ELEMENTOS SELECCIONADOS DE UN CONJUNTO DE 25 CON EL MODELO MAX-MINSUM Y CON EL MODELO MIN-DIFF 29
FIGURA 6. POSICIONES DE 12 PUNTOS UBICADOS ALEATORIAMENTE EN UN PLANO CARTESIANO 33
FIGURA 7. DOS SUBCONJUNTOS SELECCIONADOS DEL EJEMPLO MOSTRADO EN LA FIGURA 6 34
FIGURA 8. UBICACIÓN DE 12 ELEMENTOS DEL EJEMPLO 2 36
FIGURA 9. DOS SUBCONJUNTOS SELECCIONADOS EN EL EJEMPLO 2 37
FIGURA 10. UBICACIÓN DE 500 ELEMENTOS DEL EJEMPLO 3 38
FIGURA 11. DOS SUBCONJUNTOS SELECCIONADOS ARBITRARIAMENTE DEL CONJUNTO DE ELEMENTOS DEL EJEMPLO 3 38
FIGURA 12. SOLUCIONES DE LOS MODELOS MAX-SUM, MAX-MIN, MAX-MINSUM Y MIN-DIFF PARA UNA INSTANCIA EUCLIDIANA CON *N*=25, *M*=7 56
FIGURA 13. ESQUEMA DE FUNCIONAMIENTO DE LOS PROCEDIMIENTOS DE BÚSQUEDA LOCAL 65
FIGURA 14. ESQUEMA DE UN ALGORITMO GRASP GENERAL 73
FIGURA 15. ESTRUCTURA DE LOS EJEMPLOS DE PRUEBA TIPO I 82
FIGURA 16. ESTRUCTURA DE LOS EJEMPLOS DE PRUEBA TIPO II 82
FIGURA 17. SOLUCIÓN DEL PROBLEMA MAX-MEAN A TRAVÉS DE LAS SOLUCIONES DEL MODELO MAX-SUM 83
FIGURA 18: FORMA CUASI-CÓNCAVA DE LOS VALORES ÓPTIMOS DEL PROBLEMA MAX-SUM DIVIDIDOS PARA *M* 84
FIGURA 19. EVOLUCIÓN DE LOS VALORES ÓPTIMOS DEL PROBLEMA MAX-SUM DIVIDIDOS PARA SU CORRESPONDIENTE VALOR DE *M* PARA LOS 10 EJEMPLOS DE PRUEBA DE TIPO I 89
FIGURA 20. VALORES ÓPTIMOS DE TODOS LOS PROBLEMAS MAX-SUM DIVIDIDOS POR *M* PARA CADA VALOR DE *M* PARA CADA UNO DE LOS 10 EJEMPLOS DE PRUEBA DE TIPO I 90
FIGURA 21. EVOLUCIÓN DE LOS VALORES ÓPTIMOS DEL PROBLEMA MAX-SUM DIVIDIDOS PARA SU CORRESPONDIENTE VALOR DE *M* PARA LOS 10 EJEMPLOS DE PRUEBA DE TIPO II 90
FIGURA 22. VALORES ÓPTIMOS DE TODOS LOS PROBLEMAS MAX-SUM DIVIDIDOS POR *M* PARA CADA VALOR DE *M* PARA CADA UNO DE LOS 10 EJEMPLOS DE PRUEBA DE TIPO II 91
FIGURA 23. FASE DE CONSTRUCCIÓN GRASP 92
FIGURA 24. FASE DE CONSTRUCCIÓN DEL ALGORITMO GRASP_C2 ADAPTADO AL PROBLEMA MAX-MEAN . 96
FIGURA 25. ALGORITMO GRASP 2$\boldsymbol{b}$ 98
FIGURA 26. FASE DE CONSTRUCCIÓN DEL MÉTODO GRASP3 100
FIGURA 27. ALGORITMO DE BÚSQUEDA LOCAL PARA GRASP3 103
FIGURA 28. ESQUEMA REPRESENTATIVO DEL RENCADENAMIENTO DE TRAYECTORIAS 106
FIGURA 29. ESQUEMA DEL PROCEDIMIENTO *PR* EN SU FASE DE BÚSQUEDA LOCAL 107
FIGURA 30. SEUDO CÓDIGO DE LA FASE DE RENCADENAMIENTO DE TRAYECTORIAS DEL ALGORITMO GRASP3+PR 108

FIGURA 31. PERFIL DE BÚSQUEDA PARA LAS CONSTRUCCIONES GRASP EN PROBLEMAS DE PRUEBA TIPO I 112
FIGURA 32. PERFIL DE BÚSQUEDA DE LA FASE CONSTRUCTIVA DE GRASP 3 PARA LOS VALORES DE $\boldsymbol{\alpha}$ MÁS SIGNIFICATIVOS PARA EJEMPLOS TIPO I 112
FIGURA 33. PERFIL DE BÚSQUEDA DE LA FASE CONSTRUCTIVA DE GRASP 3 PARA DISTINTOS VALORES DE $\boldsymbol{\alpha}$ PARA EJEMPLOS TIPO II 113
FIGURA 34. NÚMERO PROMEDIO DE MEJORAS EN CADA TIPO DE VECINDAD EN EL ALGORITMO GRASP3 EN LOS EJEMPLOS DE PRUEBA DE TIPO I Y DE TIPO II 121
FIGURA 35. APORTE A LA MEJORA DE LA FUNCIÓN OBJETIVO EN VALOR POR CADA TIPO DE VECINDAD ... 122
FIGURA 36. CURVA DE LAS DESVIACIONES PROMEDIO RESPECTO A LOS MEJORES RESULTADOS, OBTENIDOS CON RENCADENAMIENTO DE TRAYECTORIAS USANDO DISTINTOS VALORES DE $\boldsymbol{dth*}$ PARA PROBLEMAS TIPO I 123
FIGURA 37. CURVA DE LAS DESVIACIONES PROMEDIO RESPECTO A LOS MEJORES RESULTADOS, OBTENIDOS CON RENCADENAMIENTO DE TRAYECTORIAS USANDO DISTINTOS VALORES DE $\boldsymbol{dth*}$ PARA PROBLEMAS TIPO II 124
FIGURA 38. PERFIL DE BÚSQUEDA EN INSTANCIAS GRANDES DE TIPO I 127
FIGURA 39. DETALLE DEL PERFIL DE BÚSQUEDA EN INSTANCIAS DE TIPO I 127
FIGURA 40. PERFIL DE BÚSQUEDA EN INSTANCIAS GRANDES DE TIPO II 128
FIGURA 41. DETALLE DEL PERFIL DE BÚSQUEDA EN INSTANCIAS DE TIPO II 128
FIGURA 42. DESVIACIÓN DE LOS RESULTADOS ENCONTRADOS CON EVOLVER RESPECTO A LA SOLUCIÓN ENCONTRADA CON GRASP3 132
FIGURA 43. PERFIL DE BÚSQUEDA DEL ALGORITMO GENÉTICO EJECUTADO POR EVOLVER EN PROBLEMAS DE TAMAÑO MEDIANO *N* = 150 133
FIGURA 44. ¿CUÁL ES LA MEJOR SELECCIÓN DE UN EQUIPO DE TRABAJO A PARTIR DE UNA POBLACIÓN DE PERSONAS CAPACITADAS PARA RESOLVER UN PROBLEMA PERO CON DISTINTOS GRADOS DE HABILIDAD? 141
FIGURA 45. SOLUCIONES DOMINADAS, ÓPTIMOS DE PARETO Y ÓPTIMO IDEAL 155
FIGURA 46. LA REGIÓN PARETO DOMINADA Y LA REGIÓN PARETO $\boldsymbol{\varepsilon}$-DOMINADA 157
FIGURA 47. APROXIMACIONES DE LA FRONTERA DE PARETO 158
FIGURA 48. HIPERVOLUMEN GENERADO POR UN CONJUNTO DE APROXIMACIÓN DE LA FRONTERA DE PARETO CON DOS FUNCIONES OBJETIVO 160
FIGURA 49. APROXIMACIÓN DE LA FRONTERA DE PARETO PARA UNA INSTANCIA DE TIPO I DE TAMAÑO MEDIANO $\boldsymbol{n = 150}$ 166
FIGURA 50. DETALLE DE LA FRONTERA DE PARETO PARA UNA INSTANCIA DE TIPO I DE TAMAÑO MEDIANO $\boldsymbol{n = 150}$ 167
FIGURA 51 LAS SOLUCIONES DEL PROBLEMA MULTIOBJETIVO Y EL NÚMERO DE ELEMENTOS SELECCIONADOS PARA UNA INSTANCIA DE TIPO I DE TAMAÑO MEDIANO $\boldsymbol{n = 150}$ 167
FIGURA 52. APROXIMACIÓN DE LA FRONTERA DE PARETO PARA UNA INSTANCIA DE TIPO II DE TAMAÑO MEDIANO $\boldsymbol{n = 150}$ 168
FIGURA 53. DETALLE DE LA FRONTERA DE PARETO PARA UNA INSTANCIA DE TIPO II DE TAMAÑO MEDIANO $\boldsymbol{n = 150}$ 168
FIGURA 54. SOLUCIONES DEL PROBLEMA MULTIOBJETIVO Y EL NÚMERO DE ELEMENTOS SELECCIONADOS PARA UNA INSTANCIA DE TIPO II DE TAMAÑO MEDIANO $\boldsymbol{n = 150}$ 169

ÍNDICE DE TABLAS:

TABLA 1. CARACTERÍSTICAS OBSERVADAS EN DOS PROFESORES DE UNA UNIVERSIDAD 26
TABLA 2. DISTANCIAS EUCLIDIANAS PARA EL EJEMPLO 1 33
TABLA 3. MEDIDAS DE DISPERSIÓN PARA LOS SUBCONJUNTOS $\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\ \boldsymbol{y}\ \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}$ DEL EJEMPLO 1 35
TABLA 4. MEDIDAS DE DISPERSIÓN PARA LOS SUBCONJUNTOS DEL EJEMPLO 2. 37
TABLA 5. MEDIDAS DE LA DIVERSIDAD PARA LOS DOS SUBCONJUNTOS DEL EJEMPLO 3. 39
TABLA 6. CORRELACIONES ENTRE LOS MEJORES, PROMEDIO Y PEORES RESULTADOS EN 30 INSTANCIAS CON $D_{IJ} \in$ U[0, 20], N =20, M= 5 54
TABLA 7. CORRELACIÓN ENTRE EL MEJOR, EL PEOR Y EL RESULTADO PROMEDIO EN 30 INSTANCIAS CON $D_{IJ} \in$U[-10, 10], N =20, M= 5 55
TABLA 8. NÚMERO DE VECES EN LAS CUALES LAS SOLUCIONES DE LOS MODELOS COINCIDIERON ENTRE LAS 30 INSTANCIAS PROBADAS. ESCENARIO I 55
TABLA 9. NÚMERO DE VECES EN LAS CUALES LAS SOLUCIONES DE LOS MODELOS COINCIDIERON ENTRE LAS 30 INSTANCIAS PROBADAS. ESCENARIO II 55
TABLA 10. TALLA DE LAS FORMULACIONES LINEALES DE LOS PROBLEMAS MAX-SUM Y MAX-MEAN 85
TABLA 11. TALLA DE LOS PROBLEMAS MAX-SUM Y MAX-MEAN PARA DIFERENTES VALORES DE $\boldsymbol{n}$ 86
TABLA 12. SOLUCIONES DEL PROBLEMA MAX-MEAN CON CPLEX12 87
TABLA 13. TIEMPOS DE PROCESAMIENTO DE PROBLEMAS TIPO I VS. TIEMPOS DE PROBLEMAS TIPO II EN EL MÉTODO DE RESOLVER $\boldsymbol{n} - \mathbf{1}$ PROBLEMAS MAX-SUM 88
TABLA 14. TIEMPOS DE PROCESAMIENTO DE PROBLEMAS TIPO I VS. TIEMPOS DE PROBLEMAS TIPO II EN EL MÉTODO DE RESOLVER DIRECTAMENTE LA FORMULACIÓN DEL PROBLEMA MAX-MEAN 89
TABLA 15. RENDIMIENTO DE LOS MÉTODOS EN PROBLEMAS PEQUEÑOS 114
TABLA 16. RESULTADOS DE LAS HEURÍSTICAS CON BÚSQUEDA LOCAL PARA PROBLEMAS PEQUEÑOS $\boldsymbol{n} = \mathbf{30}$ 115
TABLA 17. RESULTADOS FASE CONSTRUCTIVA PARA PROBLEMAS DE TAMAÑO $\boldsymbol{n} = \mathbf{150}$ 118
TABLA 18. RESULTADOS DE LOS MÉTODOS GRASP (CONSTRUCCIÓN + BÚSQUEDA LOCAL) EN LOS EJEMPLOS DE PRUEBA CON $\boldsymbol{n} = \mathbf{150}$ 118
TABLA 19. ESTADÍSTICOS DEL NÚMERO DE VECES QUE LA EXPLORACIÓN DE CADA VECINDAD MEJORÓ LA SOLUCIÓN EN EJEMPLOS DE PRUEBA DE TIPO I Y TIPO II 120
TABLA 20. ESTADÍSTICOS DEL APORTE A LA MEJORA DE LA FUNCIÓN OBJETIVO POR LA EXPLORACIÓN DE CADA TIPO DE VECINDAD 121
TABLA 21. COMPARACIÓN DE LOS RESULTADOS OBTENIDOS CON GRASP1, GRASP2, GRASP3 Y GRASP3 CON RENCADENAMIENTO DE TRAYECTORIAS EN LOS EJEMPLOS DE PRUEBA CON N =500 124
TABLA 22. COMPARACIÓN DE LA HEURÍSTICA GRASP3 CON EL OPTIMIZADOR EVOLVER EN EL PROBLEMA MAX-MEAN 131
TABLA 23. RESULTADOS PROMEDIO SOBRE LAS 10 CORRIDAS DE CADA ALGORITMO 146
TABLA 24. RESULTADOS INDIVIDUALES PARA CADA UNA DE LAS 10 CORRIDAS DE LOS ALGORITMOS 147

Introducción

El proceso de seleccionar objetos, actividades, ideas, personas, proyectos, recursos, etc. es una de las actividades que frecuentemente realizamos los seres humanos con algún objetivo, y basados en uno o más criterios: económicos, de espacio, afectivos, políticos, etc. Así, por ejemplo, en su vivencia diaria las personas deben seleccionar qué medios de transporte y qué rutas utilizarán para llegar a un destino determinado de acuerdo al precio, tiempo de viaje y comodidad. Al finalizar el mes un trabajador tendrá que seleccionar en qué bienes o servicios gastar su sueldo con el fin de satisfacer sus necesidades. O en la planificación de las vacaciones una persona tendrá que elegir un número dado de destinos que visitar de acuerdo al precio y placer que le proporcionarán. En todos estos casos se trata de elegir el mejor subconjunto de elementos a partir de un conjunto grande de posibilidades, el mejor en algún sentido, y en muchos casos además estaremos interesados en que los elementos seleccionados no se parezcan entre sí, sino más bien que tengan características diferentes para que representen la diversidad existente en el conjunto original. Por lo tanto, a la persona que va a seleccionar su itinerario para sus vacaciones no le interesaría visitar lugares similares y aprovecharía mejor su tiempo yendo a sitios diversos, y a los mejores. Así el problema planteado es normalmente un problema de optimización multiobjetivo. Por supuesto que a este nivel las personas toman estas decisiones intuitivamente y no optimizando el correspondiente problema de decisión sobre todos los criterios que están en juego.

Pero el sentido común, generalmente, no es un buen consejero en problemas donde se quiere optimizar la toma de decisiones, y procedimientos simples que aparentemente ofrecen soluciones eficientes llevan a malas decisiones, y sólo con la aplicación de modelos matemáticos se puede garantizar la obtención de soluciones eficientes. Y aunque para los casos anteriores parece razonable tomar las decisiones de una forma intuitiva, ya que no está mucho en juego, en otras actividades humanas la selección de aquel subconjunto tiene implicaciones económicas o de otro tipo que vuelve a la selección del mejor subconjunto, y al más diverso, una decisión crucial y difícil de obtener, que requiere de un correcto proceso de optimización guiado por alguna metodología formal.

Por ejemplo, cuando vamos a formar un equipo de trabajo en la empresa para emprender algún proyecto, nos interesaría seleccionar a los integrantes de acuerdo a sus aptitudes, afinidades

para trabajar en equipo, edades y otras que nos parezcan relevantes. Un inversionista buscará repartir su capital entre diferentes inversiones de tal manera que se diversifique el riesgo, pero que maximice su rendimiento esperado, y por tanto intentará no tener mucho dinero invertido en actividades similares que puedan ser afectadas por alguna crisis económica que impacte a actividades de un mismo sector. El gobierno buscará elegir sitios muy dispersos para localizar cierto número de instalaciones que representen algún nivel de peligro para la población como almacenes de armas y explosivos, cárceles, depósitos de basura, etc., pero a su vez estos lugares deberán ser los mejores en términos de su topografía y ubicación.

En el análisis de un grupo social nos podría interesar estudiar la diversidad presente en el mismo a través de un estudio detallado de algunos individuos representativos, estos individuos seleccionados deberían tener características presentes en la diversidad del grupo, y además ser los mejores. Estudios recientes, como el de Castillo y otros (1), han determinado que la diversidad en un grupo de personas aumenta la capacidad de estos grupos para resolver problemas, y por lo tanto, lleva a obtener grupos más eficientes en las empresas, escuelas, gobierno, etc., estos autores proporcionan una justificación teórica para este fenómeno empíricamente conocido. Investigadores pioneros de este último hecho, como Page establece en su libro, (2), The difference – How the power of diversity creates better groups, firms, schools, and societies (2007), que: "Perspectivas y herramientas diversas permiten a los grupos de personas hallar más y mejores soluciones y contribuir a la productividad total". Como resultado de lo cual, el problema de la identificación de grupos diversos de personas se convierte en un punto clave en las grandes empresas e instituciones.

Por otro lado, cuando nos interesamos en seleccionar no solo en lo más diverso, sino también en lo mejor (o lo más importante), el problema se vuelve un problema de optimización multiobjetivo, siendo el primer objetivo más difícil que el segundo, ya que lo diverso es una característica que se mide, en general, con base en alguna relación entre cada pareja de elementos, es decir, es una característica global del subconjunto seleccionado. Mientras que lo más importante, o lo mejor, es una característica que puede ser medida por medio de una ponderación o peso, que refleje de importancia de cada elemento, como lo establece Meinl en (3).

Es decir, si $V = \{1, 2, \cdots, n\}$ es el conjunto original, y M es el subconjunto a seleccionar, $M \subset V$, de cardinalidad conocida o no, buscaremos optimizar simultáneamente los dos objetivos siguientes:

$$Max\ f_1(M) = div(M) \tag{1.1}$$

$$Max\ f_2(M) = \sum_{i \in M} w(i) \tag{1.2}$$

En la ecuación (1.1) la función objetivo $div(M)$ representa la medición que hemos realizado de la diversidad en el subconjunto seleccionado, M, la cual es una característica que depende de las relaciones de similaridad o de las diferencias entre cada pareja de elementos. En cambio el segundo objetivo (1.2) es más fácil de medir ya que es la suma de los pesos o escore que tenga individualmente cada elemento seleccionado, $w(i), i \in M$.

Abordaremos en una primera parte el objetivo descrito en la ecuación (1.1), con lo que se configura el denominado Problema de la Diversidad Máxima; es decir el problema que consiste en seleccionar un subconjunto de elementos, de cardinalidad conocida o no, a partir de un conjunto dado de tal manera que una medida de la diversidad o la dispersión presente en el subconjunto seleccionado sea maximizada. En la literatura se describen muchos modelos para lograr este fin, así como también se exploran muchas aplicaciones prácticas, como en (4), (5), (6), (7), (8); en particular, en esta investigación se aborda un nuevo modelo, denominado *Modelo de Dispersión del Máximo Promedio*, que representaremos como el modelo Max-Mean, en el cual se selecciona el subconjunto que maximiza la distancia promedio entre sus elementos, de esta manera el número de elementos seleccionados también es una variable de decisión.

Así, en el primer objetivo, al proponer una función que maximiza alguna medida de la diversidad o dispersión total, estamos tratando de determinar un subconjunto de individuos que representen a la población de la cual fueron extraídos, en toda su diversidad, y de acuerdo con ciertas características, con lo que en esos conjuntos la redundancia o información similar se vería minimizada.

Pero ¿por qué estos problemas de la diversidad máxima son complicados? imaginemos que se tienen $n = 49$ personas, como en la Figura 1, de las cuales vamos a seleccionar $m = 10$ de ellas con algún propósito, pero requerimos que esta selección sea la mejor posible en el sentido

que representen en toda su diversidad al grupo original. En total existen $\binom{49}{10} \approx 8.2 \times 10^9$ maneras de seleccionar un subconjunto de 10 elementos, es más, si intentáramos resolver el asunto explorando todas las soluciones posibles y de ahí escoger la mejor, necesitaríamos evaluar además la diversidad de cada uno de los subconjuntos, pero el cálculo con la más sencilla de las medidas de diversidad, tal como se demuestra en la sección 1.3.7, es del orden $\mathcal{O}(m^2)$, es decir, alrededor de 100 operaciones para evaluar la diversidad de cada una de las soluciones factibles, por lo que se necesitan alrededor de 8.2×10^{11} operaciones elementales, que es un número muy grande para resolver el problema en un tiempo razonable.

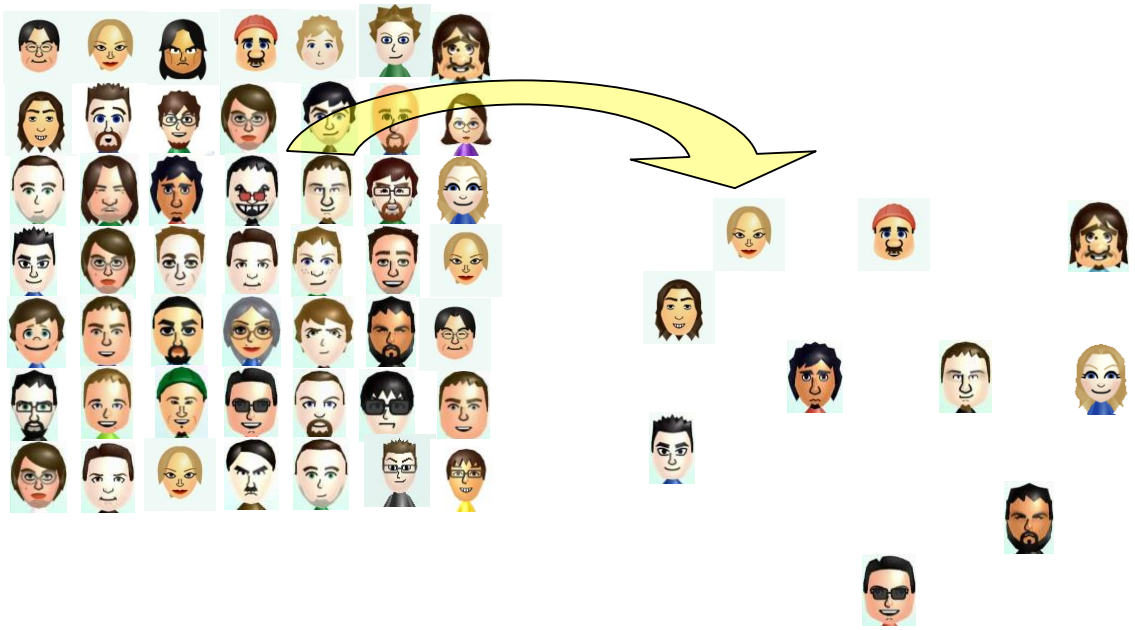


Figura 1. ¿Cómo seleccionar 10 personas de un grupo de 49 personas, con diversidad maxima?

Pero si las soluciones factibles de este problema son muchas, qué pasa si agregamos una complicación más al problema. Supongamos ahora que el problema es el siguiente: debemos seleccionar un subconjunto de este grupo de 49 personas, pero además no establecemos a priori el número de personas a seleccionarse, sino que debemos establecer como solución cuáles y cuántas personas necesitamos seleccionar para que esa selección sea la mejor posible, en términos de su diversidad. Ahora tenemos que el conjunto de soluciones factibles está compuesto por todos los subconjuntos de cardinalidad $m \geq 2$ que se pueda obtener del conjunto de 49

personas, y este número es: $\sum_{k=2}^{49}\binom{49}{k} = (2^{49} - 50) \approx 5.63 \times 10^{14}$ y con las operaciones que se requieren para evaluar la diversidad de cada una de estas soluciones el problema se vuelve inmanejable, esta explosión combinatoria del número de soluciones factibles determina que este es un problema de optimización combinatoria. Imaginemos qué sucedería cuando la población de la cual se quiere seleccionar aquel subconjunto tenga unos dos mil individuos.

Así el problema es interesante para la teoría matemática porque se trata de un problema de optimización combinatoria que no se puede resolver con técnicas exactas para tallas reales del problema, y por tanto su estudio y solución es importante como un problema de prueba para diseñar técnicas eficientes de solución.

Por otro lado, tradicionalmente en la literatura de Investigación de Operaciones, desde el punto de vista de la aplicabilidad, podemos decir que estos problemas de la diversidad máxima han permitido resolver problemas concretos de gran interés, como por ejemplo:

- El problema de localizar unidades logísticas que son mutuamente competitivas, como almacenes, plantas, centrales de transporte de carga, etc., con el fin de que sus actividades no sean redundantes entre unas y otras unidades logísticas, véase (6).

- El problema de composición de paneles de jurado, que son las listas de personas desde las que se pueden escoger a los miembros del jurado para un juicio particular, véase (9).

- El problema de la ubicación de instalaciones peligrosas, contaminantes, indeseables, o de tipo estratégico para evitar que la cercanía entre ellas genere vulnerabilidad, peligros o accidentes, véase (4).

- En la Industria farmacéutica, la elaboración de nuevas drogas es un proceso largo, que en particular requiere de la realización de experimentos con múltiples moléculas en el proceso de seleccionar una nueva droga efectiva. Estas moléculas deben tener características diversas para que el proceso no sea muy largo y costoso, véase (3).

- En la formulación de políticas de inmigración y admisiones. Algunos países establecen cupos para entregar visas de inmigrantes, una característica deseable es favorecer a

personas de diversas características, de etnicidad, de género, etc., esta aplicación es abordada en (10).

- En las técnicas metaheurísticas basadas en la evolución de una población, como los métodos genéticos y otros de tipo evolutivo, es conveniente que la población de soluciones que se genere, inicialmente o en las iteraciones sucesivas del algoritmo, sea lo más diversa posible para garantizar que el algoritmo evolucione hacia una buena solución, y evitar que se dé el fenómeno de la convergencia rápida hacia una mala solución producto de esa falta de diversidad en la población de soluciones, en estos procedimientos se debe garantizar la diversidad de las soluciones que están siendo probadas, por todo ello es importante disponer de un procedimiento que pueda generar un número suficiente de elementos diversos de la población, véase (11).

- En Agroforestería, ciencia que tiene que ver con la gestión de los sistemas agrícolas y forestales para satisfacer las necesidades del propietario de la tierra o de la sociedad, el interés se centra en características como producir más, ofrecer sombra, diversificar la producción, fijar carbono, reducir erosión, etc., una falta de diversidad en esta gestión ha generado prácticas industriales de monocultivo que provocan un gran impacto ecológico y perturbación sobre el medio ambiente. Por ello interesaría seleccionar las especies vegetales que permitan maximizar algún criterio de diversidad. Aquí la diversidad es crítica, pues las diferencias tienen un gran impacto en el medio ambiente y en las ganancias del propietario. Por ejemplo, dos bananeras, digamos cultivadas con dos especies distintas, con igual riqueza, y abundancias relativas pueden diferir grandemente en términos de manejo, impacto ecológico y valoración del propietario, véase (7).

No aparecen referencias en la literatura de otros problemas concretos que se podrían abordar como problemas de diversidad máxima, se enuncian los siguientes como potenciales casos de aplicación de este problema:

- A un usuario de internet que utiliza un buscador inteligente le gustaría que la búsqueda obtenga páginas diversas relacionadas al tema que está buscando, no le interesaría que el buscador presente muchísimas páginas de internet, de las cuales la mayoría tengan grandes similitudes unas con otras, en este sentido la implementación en el buscador de

un modelo de optimización que maximice la diversidad eliminaría en su selección páginas con información redundante.

- En el sistema de razonamiento basado en casos (Case-based reasoning, CBR), por medio del cual se resuelven nuevos problemas utilizando las soluciones a problemas que han sido previamente resueltos y guardados como casos en una Base de casos. El éxito de cualquier sistema CBR depende de su habilidad para seleccionar el caso correcto para el problema objetivo correcto. La lógica indica que el caso correcto se encuentra entre aquellos que son los más parecidos al problema objetivo, y por lo tanto una considerable investigación se ha invertido en técnicas y estrategias para la evaluación de la similitud de casos. Recientemente se ha hecho evidente que las nociones tradicionales de similitud no siempre son suficientes, inspirando a muchos investigadores a buscar formas alternativas para juzgar la utilidad de un caso en un contexto determinado. Por ejemplo, los investigadores han estudiado la importancia de la capacidad de adaptación junto con la similitud, como se ve en (12), argumentando que mientras que un caso puede parecer similar a un problema objetivo, esto no quiere decir que puede ser adaptado con éxito para este objetivo. También habría una nueva razón, como se reseña en (12), en muchos escenarios de aplicación, la diversidad de los casos de la Base de casos (uno respecto a otro), así como su similitud (en relación con el problema objetivo), debería tenerse en cuenta explícitamente para escoger de mejor manera el caso a seleccionar.

- Una red social es un grupo, generalmente muy grande, de individuos con muchas características y atributos reconocibles y medibles. En los últimos años se ha visto un crecimiento vertiginoso de estos grupos, y la aparición de nuevos. Si se tiene interés en el análisis de estas redes sociales para fines comerciales, o para seleccionar individuos representativos del colectivo para generar productos que les interesen, estos deberán ser diversos para garantizar que llegaremos a la mayoría de elementos de ese colectivo, o con fines de agruparlos para formar equipos de trabajo, formar líderes de la comunidad, aplicar programas sociales por parte del gobierno u ONG's, etc., o con fines de marketing. Los individuos seleccionados serían los patrones con los cuales se debería identificar el resto, estos patrones deberían ser lo más diversos posibles para garantizar que cada uno de ellos sea muy diferente que otro. Estudios recientes demuestran que el interés en

identificar grupos de gran diversidad en una red social, o en una sociedad en general, tiene mucha utilidad práctica, así, en (13) Santos y otros plantean un modelo, basado en la teoría de los juegos de beneficio público (Public Good Games) para calcular el porcentaje de colaboradores dentro de una comunidad en función de la diversidad de la población, y demuestran que en grupos con alta diversidad, en las que el acto de cooperación importa, la realización de tareas será mucho más eficiente, y que el número de los que cooperan aumenta en relación directa a la diversidad del grupo. En (1) se proporciona otra justificación teórica para el fenómeno de que la diversidad en un grupo de personas aumenta la capacidad de los grupos para resolver problemas, y por lo tanto, lleva a obtener grupos más eficientes.

- La selección de equipos de trabajo eficientes en una empresa u organización. Una sociedad diversa crea problemas y oportunidades, en el pasado, gran parte del interés público en la diversidad se centró en temas de justicia y de representación. Más recientemente, sin embargo, ha habido un creciente interés en explotar los beneficios de la diversidad. Por esta razón, se han iniciado esfuerzos por parte de los investigadores por identificar cómo aprovechar esta diversidad en las organizaciones humanas, empezando por investigar el rol que juega la diversidad en los grupos de personas, así por ejemplo en (14), Lu Hong introduce un marco general de trabajo para el modelado de la funcionalidad en la resolución de problemas por parte grupos diversos. En este marco, se determina que los expertos en resolver un problema poseen diferentes formas de representar al problema y algoritmos propios que utilizan para encontrar sus soluciones. Se puede usar este enfoque para establecer un resultado relativo a la composición de un grupo eficiente dentro de una empresa. En el estudio se determina que en la selección de un equipo para resolver problemas a partir de una población de agentes inteligentes, un equipo de agentes seleccionados al azar supera a un equipo compuesto por los agentes mejor capacitados. Este resultado se basa en la intuición de que, cuando el grupo inicial de solucionadores de problemas se hace más grande, los agentes de mejor desempeño necesariamente llegan a ser similares, atascándose en óptimos locales del problema, y su capacidad relativamente mayor es más que descompensada por su falta de diversidad.

- La selección de un grupo de universidades, hacia las cuales un gobierno pudiera ofrecer becas de posgrado, de tal manera que se garantice la formación de investigadores de alto nivel y con distintas visiones del mundo, que puedan aportar a la solución de los grandes problemas nacionales. Por un lado está el deseo de tener una lista de universidades diversas, con características que determinen una formación distinta de los investigadores, pero que además estén bien ubicadas en algún ranquin de universidades reconocido internacionalmente.

En particular en esta investigación estamos interesados en resolver casos relacionados con los dos últimos problemas mencionados.

Esta investigación está organizada de la siguiente manera, en primer lugar se describe el objetivo general, los objetivos específicos y la metodología de la investigación. A continuación, en el Capítulo 1 se estudian conceptos relacionados con la diversidad, y cómo ésta puede ser medida. Posteriormente, en el Capítulo 2 se introduce el clásico Problema de la Diversidad Máxima, en sus diferentes variantes, y el nuevo problema de la Dispersión del Máximo Promedio, con el cual tratamos de resolver el primer objetivo descrito por la ecuación (1.1), también se revisan las formulaciones de programación matemática para estos problemas, y se explora su posible correlación.

En el Capítulo 3 se hace una revisión general de las técnicas metaheurísticas que se usan en esta investigación para el diseño de los algoritmos de optimización, más específicamente, se hace referencia a la metodología GRASP (15), a las técnicas de Búsqueda en Vecindades Variables (16) y al procedimiento de Rencadenamiento de Trayectorias (17).

Posteriormente en el Capítulo 4 se utilizan estas metaheurísticas para diseñar un algoritmo eficiente basado en GRASP con Rencadenamiento de trayectorias en el que la búsqueda local se desarrolla principalmente con una metodología basada en vecindades variables, y también se plantean modificaciones a algoritmos previos, propuestos en la literatura por otros autores en (11) y (18) para resolver problemas que tienen alguna similitud con el problema que se trata en esta investigación.

En el Capítulo 5 se documenta la extensa experimentación computacional que se realizó para calibrar los algoritmos y para establecer comparaciones entre el algoritmo planteado y

algoritmos previos. La experiencia computacional con 120 instancias de diferentes tamaños y características muestra estos detalles, y el mérito del procedimiento propuesto. También se hace la comparación con los resultados que se obtienen al aplicar software comercial para optimización combinatoria, independiente de contexto, específicamente el programa Evolver[1], cuya principal ventaja es que puede ser aplicado a una amplia gama de problemas, aunque con menor precisión. En el Capítulo 6 se explora una aplicación del modelo para la resolución el problema de seleccionar equipos eficientes.

En el Capítulo 7 se desarrolla la extensión del problema, incluyendo los dos objetivos, con lo que el problema se transforma en la selección de lo mejor y lo más diverso, se muestra que una transformación simple lo vuelve un problema multiobjetivo equivalente a un problema de diversidad máxima de tipo Max-Mean, con lo que los algoritmos desarrollados en el Capítulo 4 se pueden aplicar para resolverlo eficientemente.

[1] Evolver es un producto comercial de Palisade Corporation.

Objetivo general y objetivos específicos de la investigación

OBJETIVO GENERAL:

El objetivo fundamental de esta investigación es analizar el Problema de la Diversidad Máxima, en su nueva variante del máximo promedio, que denominamos Max-Mean. Proponer un procedimiento eficiente, para resolver este nuevo modelo, y extender el problema incluyendo un nuevo objetivo: la selección de lo mejor. Usar el modelo extendido en la solución de casos reales.

OBJETIVOS ESPECÍFICOS:

Los objetivos específicos que permitirán lograr el objetivo principal son:

1. Investigar el estado del arte sobre los problemas de la diversidad máxima, realizando una revisión exhaustiva de la literatura relacionada con el análisis de este tipo de problemas, sus modelos, formulaciones, aplicaciones y técnicas de resolución.

2. Investigar el estado del arte sobre metaheurísticas adaptativas de multi-arranque de tipo GRASP, y su eficiencia en la resolución de problemas de optimización combinatoria.

3. Resolver con métodos exactos los modelos matemáticos de los problemas de la diversidad máxima y determinar el grado de correlación entre las soluciones de éstos.

4. Proponer un nuevo modelo de diversidad para plantear un problema de diversidad máxima, denominada del máximo promedio (Max-Mean), que permita abordar ciertos casos reales.

5. Estudiar las propiedades matemáticas de esta nueva variante del problema de la diversidad máxima.

6. Diseñar algoritmos heurísticos eficientes para resolver el problema de la diversidad máxima en su nueva variante.

7. Implementar en un lenguaje de programación adecuado los algoritmos diseñados, las implementaciones logradas deberán permitir resolver problemas grandes.

8. Analizar la eficiencia del método propuesto a través de un estudio comparativo con heurísticas adaptadas propuestas para otros modelos de la diversidad máxima. Determinar las condiciones en las cuales el procedimiento tiene buen desempeño.

9. Modelizar y resolver el problema Max-Mean en software comercial de optimización independiente del contexto y comparar las soluciones obtenidas con las heurísticas propuestas.

10. Identificar problemas asociados a la selección de equipos de personas que pudieran modelarse como un problema de diversidad máxima de tipo Max-Mean.

11. Ampliar el problema de la diversidad máxima como un problema multiobjetivo, incorporando el criterio de seleccionar “lo mejor”.

12. Transformar el problema multiobjetivo en un problema uniobjetivo del tipo Max-Mean.

13. Resolver problemas de prueba, que correspondan al problema multiobjetivo de seleccionar lo mejor y lo más diverso.

Metodología

La presente investigación es de tipo experimental, pero también con un nivel teórico y explicativo, donde se desarrollarán nuevos algoritmos híbridos para el estudio de un problema de la diversidad máxima, el problema de la diversidad del máximo promedio (Max-Mean). Los pasos metodológicos que se realizarán a lo largo de la investigación son los siguientes.

1. Investigación documental. Revisión bibliográfica en textos, revistas y en bases de datos virtuales de todo lo relacionado con el problema de la diversidad máxima y con las metaheurísticas adaptativas.

2. Definición del problema de la diversidad del máximo promedio

3. Identificación de modelos ya conocidos del problema de la diversidad máxima que estén relacionados con el Max-Mean.

4. Determinación de propiedades matemáticas y resultados teóricos para el nuevo modelo. Formulación y prueba de características del problema Max-Mean.

5. Experimentación con métodos exactos. Estudio de las soluciones óptimas exactas de los diferentes modelos considerados para instancias pequeñas. Con estas soluciones realizar un estudio de correlación para identificar posibles similitudes entre las soluciones de los modelos.

6. Experimentación con métodos existentes adaptados. Diseño de experimentos computacionales con heurísticas desarrolladas para otros problemas de diversidad máxima, adaptados a la resolución del problema Max-Mean, uso de técnicas de análisis numérico, programación y estadística. Calibración y mejora de los algoritmos ya desarrollados en base a la experiencia computacional.

7. Diseño de nuevos algoritmos. Elaboración del esquema general de los algoritmos de tipo GRASP y algoritmos híbridos basados en GRASP con Rencadenamiento de Trayectorias que se desarrollarán y el estudio de sus propiedades.

8. Diseño de problemas de prueba adecuados para comprobar la eficiencia de los distintos algoritmos propuestos a medida que aumenta el tamaño de las instancias, y como se comportan con distintas estructuras de datos.

9. Implementación. Elaboración de los programas computacionales para los algoritmos diseñados, empleando Mathematica V.7, que es un lenguaje de alto nivel compatible con otros programas computacionales, como Excel, Evolver, GAMS y R.

10. Experimentación con los programas desarrollados. Diseño de los experimentos computacionales que permitan discernir la eficiencia, rapidez y precisión de los algoritmos.

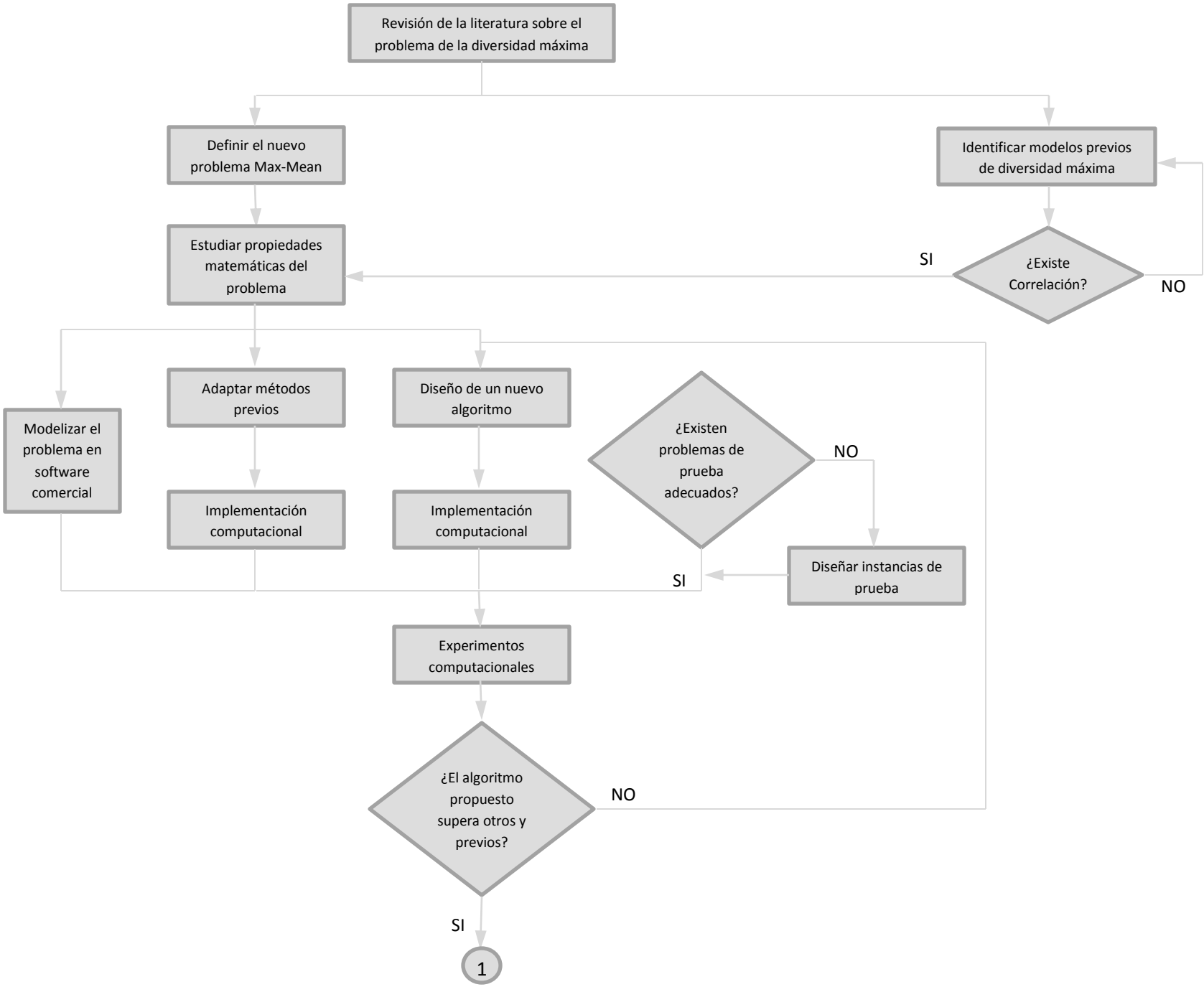
11. Comparación con software comercial. Modelizar y resolver los ejemplos de prueba con Evolver, que es un programa comercial de optimización, que utiliza algoritmos genéticos.

12. Tratamiento del problema Multiobjetivo. Plantear el problema multiobjetivo de seleccionar lo mejor y lo más diverso y transformarlo en un problema multiobjetivo similar

al Max-Mean. Uso de los algoritmos propuestos para la resolución del Max-Mean para resolver el problema multiobjetivo.

13. Solución de problemas de prueba en el que se debe seleccionar lo mejor y lo más diverso. Resolución con los algoritmos propuestos.

El diagrama de flujo con la metodología propuesta para esta investigación se puede observar en la Figura 2.


Revisión de la literatura sobre el problema de la diversidad máxima
Definir el nuevo problema Max-Mean
Identificar modelos previos de diversidad máxima
Estudiar propiedades matemáticas del problema
SI
¿Existe Correlación?
NO
Modelizar el problema en software comercial
Adaptar métodos previos
Diseño de un nuevo algoritmo
¿Existen problemas de prueba adecuados?
NO
Implementación computacional
Implementación computacional
SI
Diseñar instancias de prueba
Experimentos computacionales
¿El algoritmo propuesto supera otros y previos?
NO
SI
1

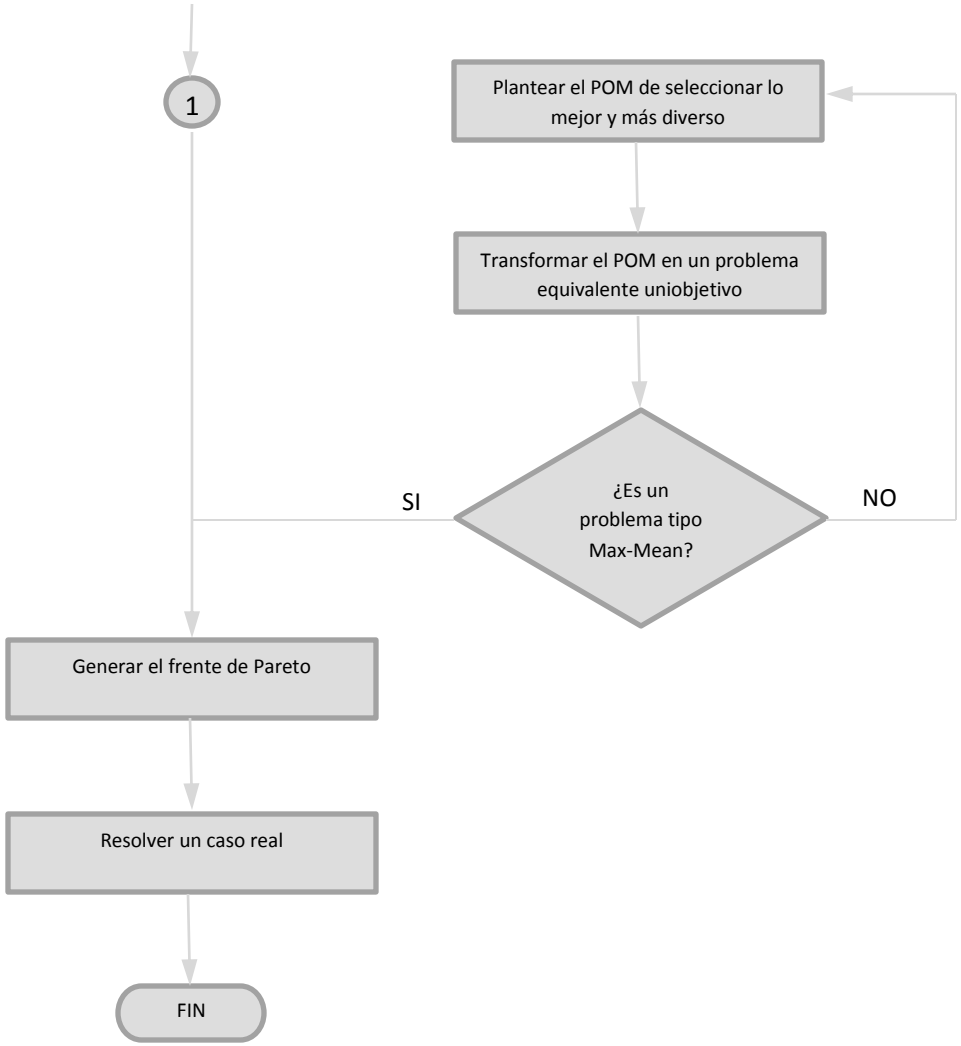


Figura 2. Diagrama de flujo de la metodología de la investigación

Capítulo 1.

Distancias, similitud y diversidad

1.1. Definiciones

Tratando de definir *diversidad* podemos decir que es un término cargado de subjetividad, pues lo que es diverso para un observador puede no serlo para otro. Y aunque la discusión filosófica está fuera del alcance de esta investigación, ya que nuestro interés está en un enfoque cuantitativo, trataremos de dar unas definiciones que se adapten al propósito de los casos a tratarse en esta investigación.

Según el diccionario de la Real Academia de la Lengua se entiende por *similitud* a una relación de semejanza entre personas o cosas. Y aunque es común aceptar que la diversidad es un concepto opuesto a la similitud, los dos términos actúan sobre estructuras diferentes, ya que la similitud es una función local para cada pareja de elementos, y la similitud de un elemento con respecto a otro no depende de la similitud con cualquier otro elemento. En contraste, la diversidad es una característica asociada a todo un conjunto de elementos, que se calcula en función de la disimilitud entre cada par, donde la *disimilitud* es exactamente lo opuesto a la similitud.

Así, la diversidad de un conjunto es calculada como una función de la disimilitud entre las parejas de elementos. Por ejemplo, en (12) se define la diversidad de un conjunto como la disimilitud promedio entre todas las parejas de elementos, de una forma parecida a la propuesta de esta investigación que se presenta en la sección 1.3.3.

Más específicamente, para medir la diversidad existente en un conjunto M, representada con $div(M)$, se requiere primero tener bien definida una relación d_{ij} que describa la relación, distancia, similitud o semejanza entre cada pareja de objetos i, j del conjunto original. La estimación de esta distancia depende del problema concreto que se está analizando, en particular, en sistemas complejos como grupos sociales, una operación fundamental es la valoración de la similitud entre cada pareja de individuos. Muchas medidas de similitud que se han propuesto en la

literatura, tal como refieren Santini y otros en (19), se valora la similitud como una distancia en algún espacio con características adecuadas, generalmente un espacio métrico, incluso en muchos casos se asocia la distancia Euclidiana entre cada pareja de elementos.

En la mayoría de aplicaciones se supone que cada elemento puede representarse por un conjunto de atributos (datos o características), y si definimos x_{ik} como el valor del atributo *k*-ésimo del elemento i, entonces la distancia $d(x_i, x_j)$ será considerada la distancia entre los elementos i, j, representada por d_{ij}. Por ejemplo, utilizando la distancia Euclidiana:

$$d_{ij} = \sqrt{\sum_k (x_{ik} - x_{jk})^2}$$

Bajo este modelo la distancia, d, satisface los axiomas de una métrica, sin embargo la observación empírica de atracciones y diferencias entre individuos hace que debamos abandonar la mayoría de estos axiomas, véase (19).

El primer axioma de métrica establece que la distancia es una función no negativa:

$$d(x_i, x_j) \geq 0, \forall\, x_i, x_j$$

La consideración de este axioma impide reconocer cuando existe una similitud o una disimilitud, una atracción o una repulsión, entre una pareja de individuos, ya que en general se requiere la asignación de un signo, positivo o negativo, para detectar de una manera efectiva estas características.

El segundo axioma establece la obligatoriedad de que la distancia es nula si y sólo si los elementos considerados son los mismos:

$$d(x_i, x_j) = 0 \; si \; y \; sólo \; si \; x_i = x_j$$

Para nuestro objetivo de representar la similitud esto podría ser muy restrictivo, pues podríamos tener que cuando $d(x_i, x_j) = 0$ existe indiferencia entre los individuos, pudiendo ser individuos distintos.

Un tercer axioma establece la desigualdad triangular:

$$d(x_i, x_j) + d(x_j, x_k) \geq d(x_i, x_k)$$

Esta desigualdad es intuitivamente fácil de comprender en el espacio euclidiano, pero para el caso de relaciones entre individuos podría resultar una imposición demasiado fuerte, que no tiene que ser forzosamente satisfecha.

El cuarto y último axioma es el de la simetría:

$$d(x_i, x_j) = d(x_j, x_i)$$

Esta es una propiedad que se puede considerar, ya que no habría a priori ningún motivo para que cambie el valor de la similitud entre i y j y el valor de la similitud entre j y i.

A partir de esto se observa que los modelos geométricos para modelizar las distancias inter-elemento $d_{i,j}$ son demasiado restrictivos: de los cuatro axiomas de distancia tres son cuestionables para ser considerados y únicamente el cuarto es aceptable. Así, los axiomas de distancia fuerzan innecesariamente a un rígido sistema de propiedades que no se adaptan adecuadamente al marco de trabajo de esta investigación: las medidas de similitud.

Por esto es clara la necesidad de considerar valores de las distancias inter-elemento positivas y negativas, pues es la manera más natural de modelar y comprender las relaciones entre las personas integrantes de un colectivo social. Asociaciones positivas pueden atribuirse a rasgos parecidos, atracciones o similitud, lo contrario para valores negativos.

En la literatura se pueden encontrar diferentes medidas de similitud o de disimilitud que se aplican a grupos de personas. Por ejemplo, en (20) se establece que."la medida de similitud del coseno es una medida popular de similitud". Por otro lado en (21) se establece una medida de disimilitud para tratar el problema de la relación entre la diversidad y la productividad en empresas o grupos de personas que se establecen para resolver problemas. Estas medidas son desarrolladas en la sección 1.2.

1.2. Medidas de similitud

Existen varias medidas de similitud que se mencionan en la literatura, en particular aquí se mencionan dos que se acoplan a las características del modelo Max-Mean: la medida de similitud

del Coseno y la medida de similitud en grupos de solucionadores de problemas definida por Lu Hong en (21).

Dados dos individuos i, j con características $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$, $x_j = (x_{j1}, x_{j2}, \dots, x_{jp})$ se define la medida de similitud del Coseno como:

$$d_{ij} = \frac{\sum_{k=1}^{p} x_{ik} x_{jk}}{\sqrt{\sum_{k=1}^{p} {x_{ik}}^2} \sqrt{\sum_{k=1}^{p} {x_{jk}}^2}} \tag{1.3}$$

La interpretación de esta medida de similitud es directa, pues d_{ij} en la ecuación (1.3) puede ser vista como el coseno del ángulo formado entre el vector de características del elemento i y el vector de características del elemento j, y por tanto los d_{ij} tomarán valores en el intervalo $[-1, 1]$. Mientras más pequeño es el ángulo, más cercano a 1 será el coseno y más grande será la similitud de los dos individuos. Y si el ángulo se aproxima a π los individuos estarán en "polos opuestos", con su coseno d_{ij} aproximándose a -1 y más grande será su disimilitud.

En la Figura 3 se representan tres situaciones posibles, en los tres casos están considerados dos individuos a través de su vector de características, la gráfica de la izquierda indica que los individuos son altamente similares, la del centro representa a dos individuos altamente disimilares, y en la derecha se está representando a dos individuos indiferentes en las características medidas, correspondientemente los cosenos de los ángulos que forman sus vectores de características son cercanos a 1, a -1 y a 0.

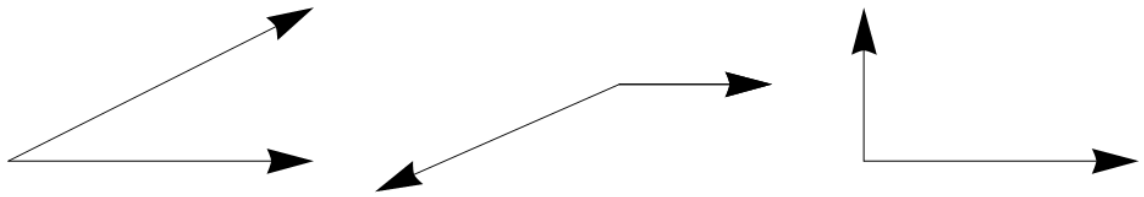


Figura 3. Los vectores representan las características de dos individuos en tres casos, el ángulo indica la medida de similitud entre ellos

Por último, al ser la diversidad un concepto opuesto a la similitud, resulta más útil considerar, para nuestro estudio, la medida $d'_{ij} = -d_{ij}$, que también toma valores en el intervalo $[-1, 1]$ pero con esta medida la maximización de una función objetivo basada en d'_{ij} implicará

buscar la mayor diversidad. En lo que sigue utilizaremos la notación d_{ij} para referirnos a la medida d'_{ij}.

Para ilustrar el cálculo de la estimación de los coeficientes de disimilitud, d_{ij}, en una aplicación real, consideremos la población de profesores de una Universidad, en este caso la Escuela Superior Politécnica del Litoral de Guayaquil - Ecuador, de la que estamos interesados en las características: condición laboral del profesor, género, máximo nivel académico, si el profesor obtuvo su título profesional en la misma Universidad, el tipo de relación laboral del profesor con la Universidad, Unidad a la que pertenece y nivel salarial, todas estas características codificadas adecuadamente.

Supongamos que tenemos dos profesores de esta población, denominados 1 y 2, con las características observadas en la Tabla 1:

Tabla 1. Características observadas en dos profesores de una Universidad

PROFESOR (i)	CONDICIÓN LABORAL	GÉNERO	MÁXIMO NIVEL ACADEMICO	ESTUDIO EN LA MISMA	TIPO DE PROFESOR	UNIDAD A LA QUE PERTENECE	NIVEL SALARIAL
1	1	-1	0.5	-1	-1	0.5	0.3
2	1	-1	1	1	1	0.5	-0.6

Entonces, el coeficiente de disimilitud entre estos profesores es:

$$d_{12} = -\frac{\sum_{k=1}^{p} x_{1k} y_{2k}}{\sqrt{\sum_{k=1}^{p} {x_{1k}}^2} \sqrt{\sum_{k=1}^{p} {y_{2k}}^2}} = -0.112$$

Por otro lado, en (21) los autores plantean el problema de cómo la diversidad presente en un grupo puede aumentar la eficiencia para resolver problemas, en particular en su investigación los autores usan la siguiente medida de disimilitud:

$$d_{ij} = \frac{\sum_{l=1}^{p} \delta(x_{il}, x_{jl})}{p}$$

Dónde:

$$\delta(x_{il}, x_{jl}) = \begin{cases} -1 & si\ x_{il} = x_{jl} \\ |x_{il} - x_{jl}| & si\ x_{il} \neq x_{jl} \end{cases}$$

Es obvio que esta medida toma valores negativos (en el caso de similitud) y positivos (en el caso de disimilitud). La idea aquí es que características iguales de dos individuos son contrarias a la diversidad y por tanto se castigan con -1, mientras que características diferentes de los individuos aportan a la medida de la disimilitud con una cantidad positiva, dependiendo su valor de cuánto valoramos esta característica dentro del conjunto de atributos considerados.

Como ejemplo se observa que, para el caso presentado en la Tabla 1, el coeficiente de disimilitud sería:

$$d_{12} = \frac{-1 - 1 + .5 - 1 + 1 - 1 + 0.9}{7} = -0.09$$

1.3. Equidad, diversidad y dispersión

Una vez entendido el concepto de las distancias inter-elemento d_{ij}, antes de plantear el problema de optimización, viene la cuestión de cómo entender términos como diversidad, dispersión y equidad; y por supuesto, cómo estimar cuantitativamente la diversidad existente en un subconjunto conocido.

En nuestro estudio usamos el término diversidad o *dispersión* indistintamente. El problema es que, el término diversidad en sí es muy vago, y está lleno de subjetividad, en todo caso podemos afirmar que es un concepto asociado a una característica global del subconjunto seleccionado, y que debe ser evaluado en función de las distancias interelemento d_{ij} consideradas. Además la forma de evaluar la diversidad va a depender íntimamente del problema del "mundo real" que se trate.

El creciente interés en el tratamiento de la diversidad también ha originado un esfuerzo por estudiar la gestión de la *equidad*, es decir todas las prácticas y procesos utilizados en las organizaciones para garantizar un trato justo y equitativo de individuos o instituciones. Hablando en términos generales, lo equitativo es aquello que tiene o exhibe equidad, siendo sinónimos los

términos: justo, objetivo o imparcial. Muchos autores, como French, en (22) argumentan que la equidad tiene que ver con la justicia, por ejemplo de la distribución de recursos o de instalaciones o infraestructura de servicio público, y de esta manera el logro de la equidad en la diversidad ha sido identificado dentro como un problema de selección y distribución. Sintetizando lo que se enuncia en la literatura, podemos decir que la equidad representa un argumento concerniente a la justicia de una disposición, entendiendo a esta como el tejido complejo de decisiones, acciones, y resultados en los cuales cada elemento engrana como miembro de un subconjunto dado.

Así, sin ser rigurosos, y de una manera intuitiva, se podría interpretar los subconjuntos diversos como aquellos que están esparcidos "equitativamente" dentro de la distribución de los elementos de todo el conjunto original, en ese sentido nuestra percepción tal vez diría que los objetos seleccionados en la Figura 4, izquierda, obtenidos con el denominado modelo Max-Sum son menos diversos que los seleccionados en la Figura 4, derecha, obtenidos con el denominado modelo Max-Min, modelos que son definidos en la sección 2.1, en esta figura los elementos con íconos de aviones forman el conjunto diverso seleccionado, los otros íconos representan a los elementos no seleccionados.

En este ejemplo, mostrado en la Figura 4 y en la Figura 5, las distancias inter-elemento se calcularon con la métrica Euclidiana, los resultados fueron obtenidos con los modelos de programación matemática formulados en el Capítulo 2 y usando el optimizador con métodos exactos Cplex[2].

[2] IBM ILOG CPLEX Optimization Studio, conocido informalmente como CPLEX, es un paquete de software de optimización que en 2004 ganó el premio INFORMS al mejor programa de optimización lineal y entera, en CPLEX se incluyen los últimos desarrollos del estado del arte de métodos de programación matemática para la resolución de problemas lineales, enteros y cuadráticos.

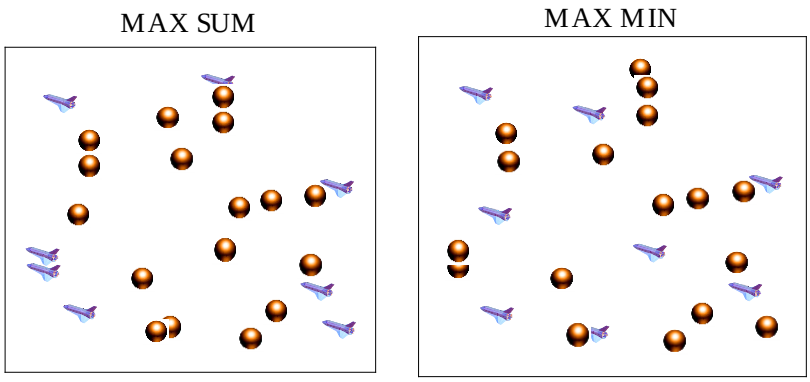


Figura 4: Ocho elementos seleccionados de un conjunto de 25 con el modelo Max-Sum y con el modelo Max-Min

Para los mismos datos del problema de la Figura 4, en la Figura 5 se presentan los subconjuntos más diversos en términos de las medidas de la mínima suma y del diferencial, en estos problemas el interés será maximizar la máxima suma y minimizar el diferencial, respectivamente, y se los denomina Max-MinSum y Min-Diff. La formulación detallada de estos modelos se encuentra en la sección 2.1.

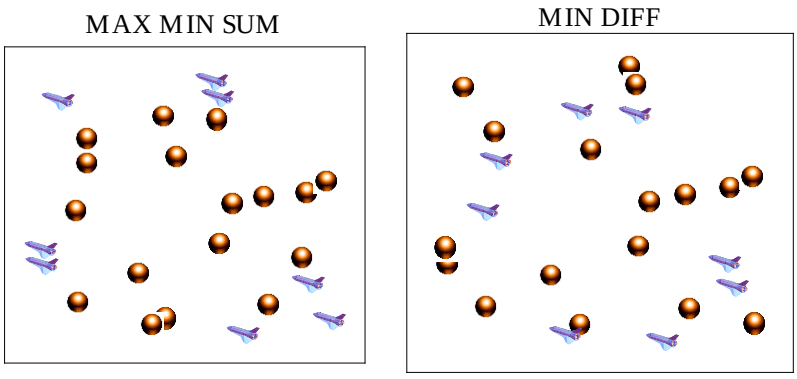


Figura 5: Ocho elementos seleccionados de un conjunto de 25 con el modelo Max-MinSum y con el modelo Min-Diff

El inconveniente que se presenta es que esta forma de ver la diversidad sólo nos es útil en espacios como el euclidiano, además de que no está claro si esta concepción "visual" de la diversidad es la correcta, o si más bien al hacer esta interpretación estamos sesgados por nuestra concepción geométrica de las cosas. En particular con una medida d_{ij} general como la definida en la ecuación (1.3) ya no es posible esta interpretación.

Entonces, una interpretación más general de la diversidad implica abandonar esta concepción intuitiva y geométrica, y establecer la diversidad desde un punto de vista más amplio, olvidando el enfoque euclidiano, y permitiendo utilizar distancias inter-elemento más generales, o relaciones que no están obligadas a cumplir las reglas de una métrica. De esta manera, el problema de medir la diversidad en un conjunto dado no es único, y da lugar a diferentes problemas.

Luego, dado un conjunto $V = \{1, 2, \cdots, n\}$, y cualquier relación inter-elemento d_{ij} definida entre cada pareja de elementos de V, y un subconjunto $M \subset V$, debemos definir como se medirá la diversidad de M: $div(M)$. La manera en que midamos esta diversidad da origen a los diferentes modelos del problema de la diversidad máxima.

A continuación, en las secciones 1.3.1, 1.3.2, 1.3.3, 1.3.4 y 1.3.5 se realiza una breve descripción de estas medidas:

1.3.1. La medida de dispersión de la Suma

Con esta medida se calcula la diversidad de un conjunto como la suma de las distancias inter-elemento de todos sus elementos; es decir, la diversidad de un conjunto M se calcula con la ecuación (1.4):

$$div(M) = \sum_{i<j, i,j \in M} d_{ij} \qquad (1.4)$$

En el objetivo de seleccionar al subconjunto más diverso, este tipo de medida fuerza a seleccionar elementos que estén muy distantes de otros, pero paradójicamente elementos que estén muy cercanos entre sí también podrían favorecer esta medida de diversidad del conjunto, si es que estos elementos se encuentran muy lejos de otros, como se puede observar en la Figura 4 izquierda, en la cual se representa el subconjunto con la mayor diversidad en términos de esta medida.

Esta medida de diversidad da origen al problema de optimización Max-Sum que se detalla en la sección 2.1.1.

1.3.2. La medida de dispersión de la Mínima Distancia

En este caso la diversidad de un conjunto dado se establece como la mínima de las distancias entre las parejas de elementos del conjunto; es decir, como se expresa en la ecuación (1.5).

$$div(M) = \min_{i<j,i,j\in M} d_{ij} \tag{1.5}$$

Este tipo de medida puede ser útil en contextos en los cuales es indeseable elegir elementos muy cercanos, y por tanto tener una mínima distancia que sea grande es importante. Esta medida da origen al problema de optimización Max-Min que se detalla en la sección 2.1.2, y al contrario de la medida de la máxima suma, tal como se observa en la Figura 4, elementos cercanos afectan grandemente a la diversidad, y por tanto al maximizarla se evita que se seleccionen elementos próximos.

1.3.3. La medida de la dispersión Promedio

En este caso se define esta medida como una medida de la diversidad media de los elementos seleccionados. Así, para un conjunto M, la diversidad promedio es calculada por la expresión de la ecuación (1.6)

$$div(M) = \frac{\sum_{i<j,i,j\in M} d_{ij}}{|M|} \tag{1.6}$$

Nótese que esta medida de diversidad está íntimamente asociada con la medida de la dispersión de la Suma, que constituye el numerador de la ecuación (1.6). En la literatura han aparecido últimamente algunas referencias en las cuales se mide la diversidad de esta manera. Así, en el contexto del problema de los sistemas de razonamiento basado en casos (Case-based reasoning, CBR) (12), los autores definen la diversidad de un conjunto de casos, como la disimilitud promedio entre todas las parejas de casos considerados. En tanto que en (18) se define la diversidad de un conjunto como en la ecuación (1.6) dentro del contexto de los modelos de dispersión equitativa.

1.3.4. La medida de dispersión de la Mínima Suma

Dado un conjunto M, esta medida se establece de acuerdo a la ecuación (1.7):

$$div(M) = \min_{i \in M} \sum_{j \in M, j \neq i} d_{ij} \quad (1.7)$$

Intuitivamente la ecuación (1.7) puede ser interpretada como la mínima dispersión agregada asociada a cada uno de los elementos del conjunto.

1.3.5. La medida de dispersión del Diferencial

Esta medida se define de acuerdo a la ecuación (1.8):

$$div(M) = \max_{i \in M} \sum_{j \in M, j \neq i} d_{ij} - \min_{i \in M} \sum_{j \in M, j \neq i} d_{ij} \quad (1.8)$$

Es decir, con esta medida la diversidad se mide como la diferencia entre la máxima y la mínima suma de las distancias desde cada elemento seleccionado a los otros elementos del conjunto seleccionado.

Prokopyev y otros, en (18), plantean originalmente las medidas (1.6), (1.7) y (1.8) en el problema de la diversidad máxima dentro del contexto de los modelos de dispersión equitativa, como una alternativa a las medidas (1.4) y (1.5).

1.3.6. Ejemplos ilustrativos

En esta subsección se plantean tres ejemplos, en los cuales la distancia d_{ij} ha sido calculada como la distancia euclidiana en un plano, con el fin de ilustrar las diferentes medidas de diversidad descritas. Dado su carácter euclidiano permiten realizar una percepción visual de las definiciones.

Todos los ejemplos se han construido sobre un rectángulo $[0,10] \times [0,10]$ del plano. Sobre este rectángulo se han ubicado diferentes cantidades de puntos en posiciones aleatorias y la matriz de distancias d_{ij} que se considera corresponde a la matriz de distancias euclidianas entre estos puntos.

Para el primer ejemplo, en la Figura 6 se muestran las posiciones de $n = 12$ nodos ubicados aleatoriamente, cuya matriz de distancias euclidianas se presenta en la Tabla 2.

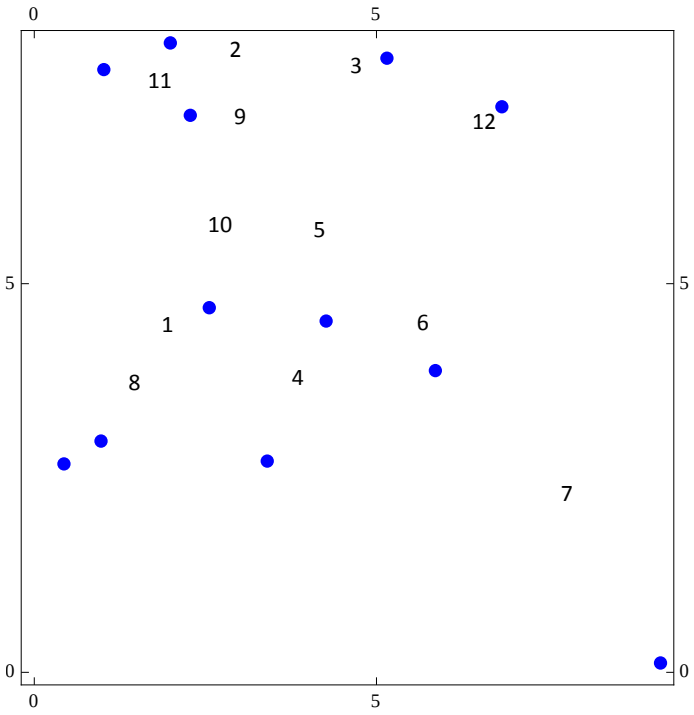


Figura 6. Posiciones de 12 puntos ubicados aleatoriamente en un plano cartesiano

Tabla 2. Distancias Euclidianas para el ejemplo 1

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.000	5.212	6.454	2.439	3.629	4.966	8.660	0.626	4.378	2.325	4.769	7.259
2	5.212	0.000	3.171	5.564	4.241	5.728	10.721	5.632	0.980	3.452	1.035	4.913
3	6.454	3.171	0.000	5.476	3.500	4.088	8.751	7.041	2.964	4.132	4.143	1.793
4	2.439	5.564	5.476	0.000	2.000	2.721	6.307	2.975	4.585	2.150	5.575	5.705
5	3.629	4.241	3.500	2.000	0.000	1.720	6.574	4.255	3.304	1.720	4.587	3.765
6	4.966	5.728	4.088	2.721	1.720	0.000	4.994	5.565	4.857	3.406	6.207	3.532
7	8.660	10.721	8.751	6.307	6.574	4.994	0.000	9.093	9.836	8.025	11.157	7.521
8	0.626	5.632	7.041	2.975	4.255	5.565	9.093	0.000	4.847	2.924	5.101	7.879
9	4.378	0.980	2.964	4.585	3.304	4.857	9.836	4.847	0.000	2.486	1.402	4.550
10	2.325	3.452	4.132	2.150	1.720	3.406	8.025	2.924	2.486	0.000	3.427	4.997
11	4.769	1.035	4.143	5.575	4.587	6.207	11.157	5.101	1.402	3.427	0.000	5.839
12	7.259	4.913	1.793	5.705	3.765	3.532	7.521	7.879	4.550	4.997	5.839	0.000

Para este ejemplo, con el fin de observar cómo se calculan las distintas medidas de diversidad, se han seleccionado arbitrariamente dos subconjuntos diferentes de puntos, que dan lugar a los ejemplos 1a. y 1b:

- En el EJEMPLO 1a. el subconjunto seleccionado es $\{1, 2, 3, 4\}$; y,
- En el EJEMPLO 1b. se seleccionó el subconjunto $\{5, 6, 7, 8\}$.

Estos dos subconjuntos seleccionados se muestran en la Figura 7, en la cual podemos identificar a los objetos que hemos seleccionado como los nodos coloreados con rojo, mientras que los nodos en azul son los elementos del grupo original mostrado en la Figura 6 que no han sido seleccionados.

La selección de estos puntos se ha realizado aleatoriamente, y el objetivo del ejemplo es únicamente tratar de asociar nuestra idea intuitiva de lo que es más diverso con lo que se concluiría a través de la utilización de las medidas de diversidad que se han establecido en la literatura.

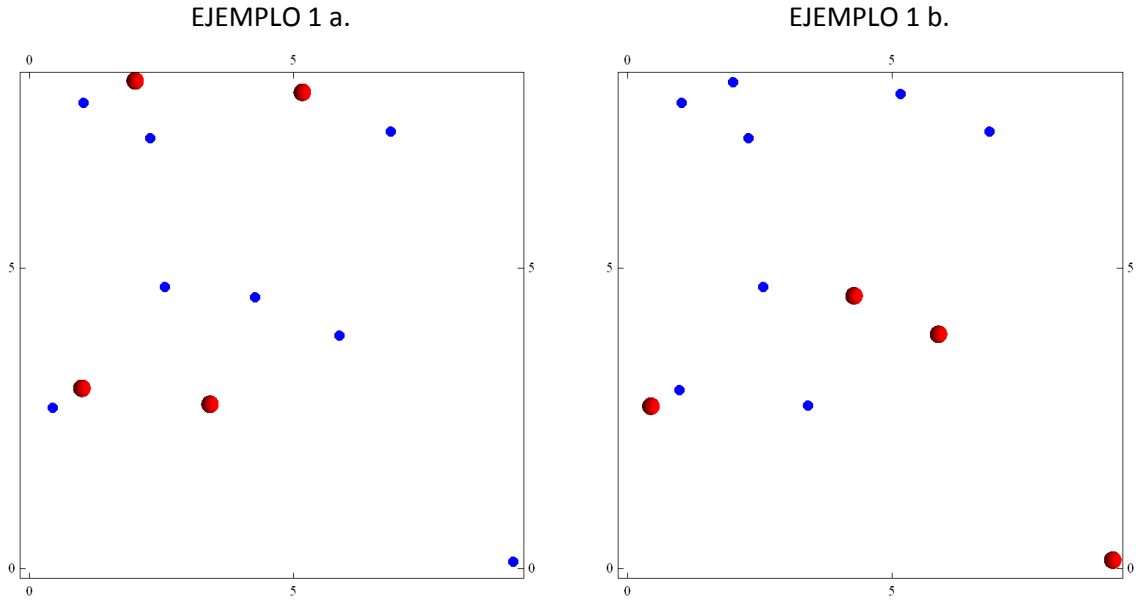


Figura 7. Dos subconjuntos seleccionados del ejemplo mostrado en la Figura 6

Para estos dos casos del ejemplo 1, las medidas de dispersión que se han establecido dan los valores de diversidad que se muestran en la Tabla 3:

Tabla 3. Medidas de dispersión para los subconjuntos $\{1, 2, 3, 4\}$ y $\{5, 6, 7, 8\}$ del ejemplo 1

MEDIDA	EJEMPLO 1a.	EJEMPLO 1b.
Dispersión de la suma	28.315	32.1994
Dispersión de la mínima distancia	2.43878	1.71994
Dispersión promedio	7.07875	8.04984
Dispersión de la mínima suma	13.4786	12.2785
Dispersión del diferencial	1.62194	8.38182

Eso quiere decir, que según las medidas de dispersión de la mínima distancia y de la mínima suma, el subconjunto $\{1,2,3,4\}$ del ejemplo 1a es más diverso que el subconjunto $\{5,6,7,8\}$ del ejemplo 1.b, mientras que, según las medidas de dispersión de la suma, de la dispersión promedio y del diferencial, el subconjunto del ejemplo 1b es más diverso que aquel del ejemplo 1a.

¿Pero se puede decir realmente cuál de las dos interpretaciones es mejor? O más específicamente ¿cuál de los dos subconjuntos es verdaderamente el más diverso? Obviamente la respuesta depende del modelo que hemos asumido para medir la diversidad, cuestión que tiene que ver en gran medida con el problema concreto que se analiza. En unos casos será aconsejable una medida, y en otros una medida diferente.

Para el segundo ejemplo, se muestra en la Figura 8 la ubicación de otros 12 objetos, de los cuales nuevamente se han seleccionado dos subconjuntos de 4 objetos, que se observan en la Figura 9 coloreados con rojo, para este ejemplo se omite la tabla de distancias inter-elemento, que corresponde a las distancias euclidianas entre los nodos.

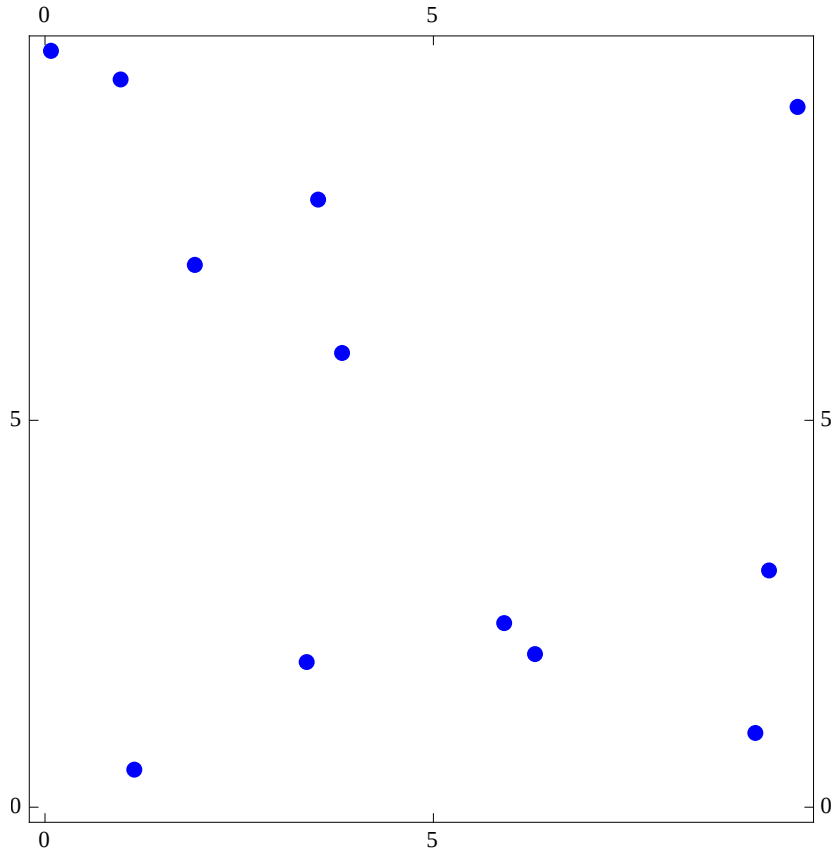


Figura 8. Ubicación de 12 elementos del ejemplo 2

En este segundo ejemplo las medidas de dispersión de los subconjuntos seleccionados se muestran en la Tabla 4. Luego, según las medidas de diversidad de la suma, de la mínima distancia, del promedio y de la mínima suma, el subconjunto del caso 2b es más diverso que el del caso 2a, mientras que según la medida del diferencial, el subconjunto del caso 2a es más diverso que el del caso 2b.

También se observa que unas medidas discriminan más que otras, por ejemplo según la medida de la mínima distancia el caso 2b es mucho más diverso que el caso 2a, pues sus medidas son 3.94 y 2.58, respectivamente, en cambio según la medida de dispersión de la suma estos dos subconjuntos son casi igual de dispersos, ya que sus respectivas medidas son 41.55 y 40.76.

EJEMPLO 2 a.

EJEMPLO 2 b.

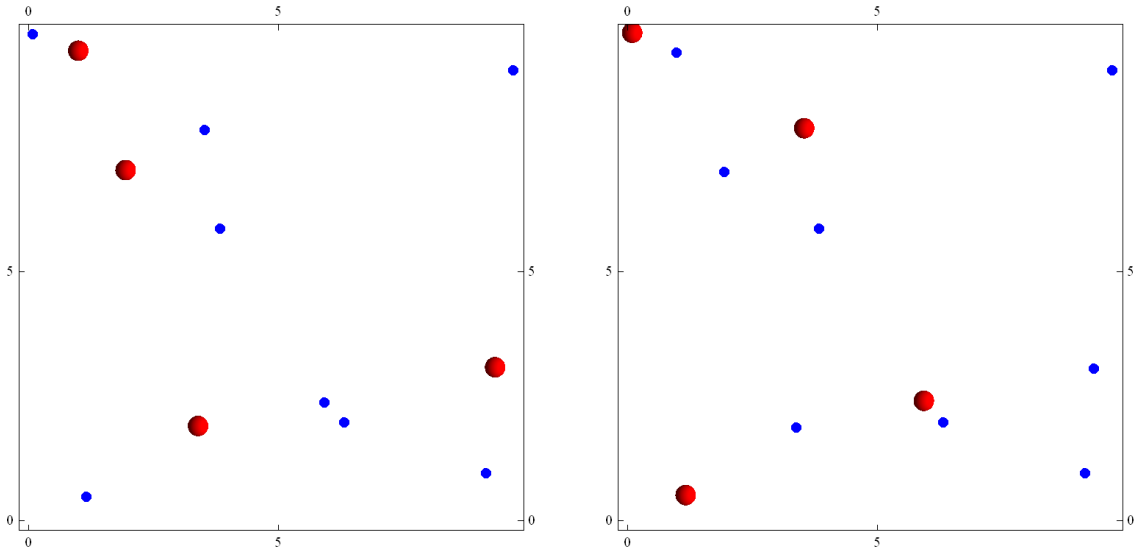


Figura 9. Dos subconjuntos seleccionados en el ejemplo 2

Tabla 4. Medidas de dispersión para los subconjuntos del ejemplo 2.

MEDIDA	EJEMPLO 2a.	EJEMPLO 2b.
Dispersión de la suma	40.7558	41.5476
Dispersión de la mínima distancia	2.57783	3.93935
Dispersión promedio	10.1889	10.3869
Dispersión de la mínima suma	16.3031	17.6548
Dispersión del diferencial	8.62596	5.05029

Por último consideremos un tercer ejemplo euclidiano, donde el conjunto de elementos tiene una gran cardinalidad, específicamente $n = 500$, al igual que en los ejemplos anteriores la ubicación de los puntos es aleatoria y se muestra en la Figura 10.

Para este ejemplo se han seleccionado arbitrariamente dos subconjuntos distintos de $m = 80$ elementos, que generan los ejemplos 3a y 3b, que se observan en la Figura 11.

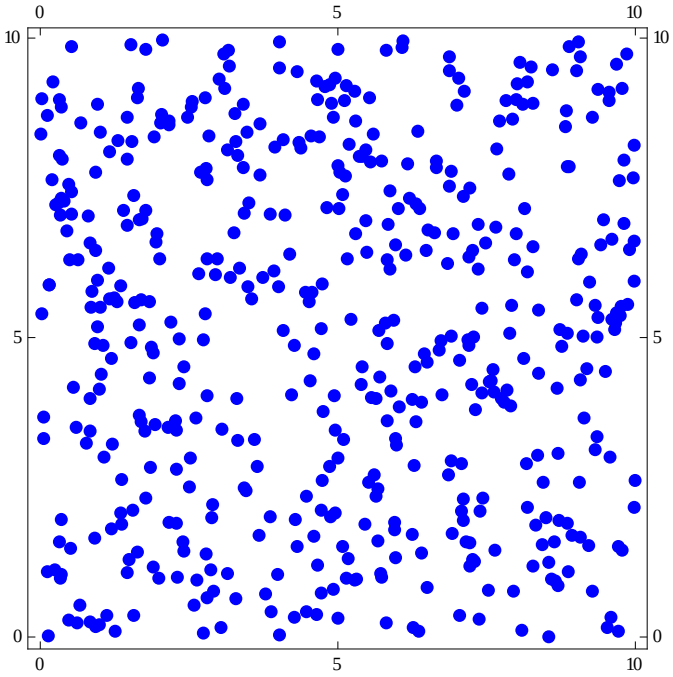
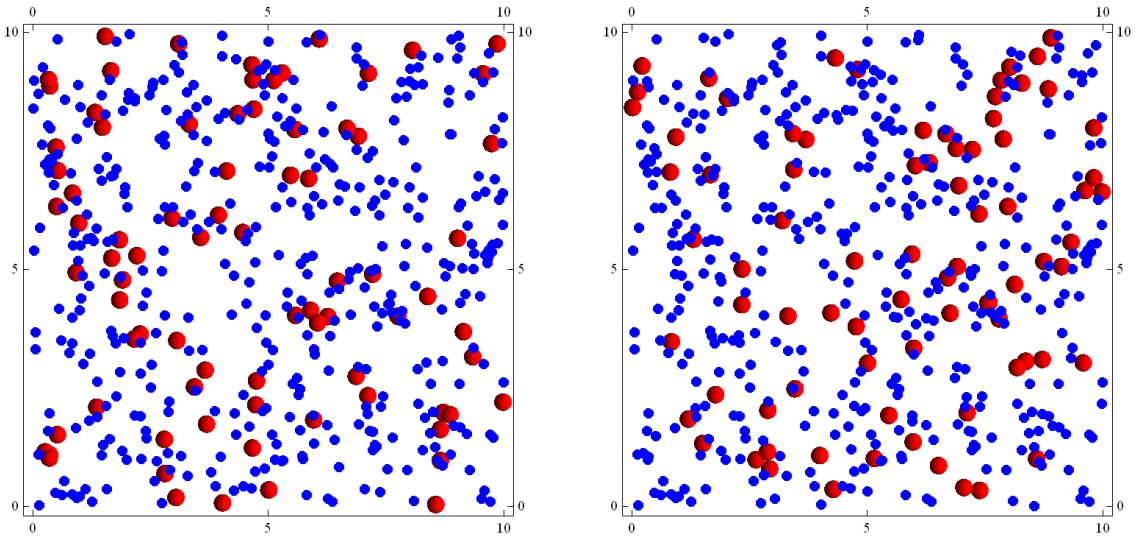


Figura 10. Ubicación de 500 elementos del ejemplo 3

Para este tercer ejemplo, todas las medidas de la diversidad calculadas, reportadas en la Tabla 5, indican que el subconjunto del ejemplo 3a es más diverso que el subconjunto del ejemplo 3b. Así, en este caso, con todas las medidas de diversidad se concluye que un conjunto es más diverso que el otro.



EJEMPLO 3 a. EJEMPLO 3 b.

Figura 11. Dos subconjuntos seleccionados arbitrariamente del conjunto de elementos del ejemplo 3

Tabla 5. Medidas de la diversidad para los dos subconjuntos del ejemplo 3.

MEDIDA	EJEMPLO 3a.	EJEMPLO 3b.
Dispersión de la suma	16885.3	16134.6
Dispersión de la mínima distancia	0.063908	0.0294764
Dispersión promedio	211.066	201.683
Dispersión de la mínima suma	313.512	296.206
Dispersión del diferencial	304.524	288.803

Una conclusión que se obtiene a partir de estos ejemplos, es que no hay unanimidad entre las medidas consideradas para decidir qué conjunto es más diverso que otro, aunque a veces varias medidas, o incluso todas pueden coincidir en este juicio. A partir de aquí se vuelve interesante analizar si es que existe algún tipo de correlación entre este tipo de medidas, se da contestación a esta pregunta en la sección 2.3

1.3.7. Complejidad en la evaluación de las medidas de diversidad

En cuanto a la complejidad de evaluar la diversidad por medio de estas medidas podemos decir lo siguiente, como para el cálculo de la diversidad con las medidas de la suma, de la mínima distancia y de la dispersión promedio sólo se requiere de las distancias entre los elementos seleccionados, entonces su cálculo es de complejidad $\mathcal{O}(m^2)$. En cambio el cálculo de la diversidad con las medidas de la mínima suma y de la dispersión diferencial es de complejidad $\mathcal{O}(m^3)$, ya que para cada elemento del conjunto seleccionado se deben efectuar aproximadamente m^2 sumas.

Es decir, evaluar la diversidad de un conjunto dado a través de las ecuaciones (1.4), (1.5), (1.7), (1.6) y (1.8) no presenta ninguna dificultad en cuanto a tiempo computacional, ya que se está midiendo la diversidad sólo en función de los elementos seleccionados, y no se toma en cuenta los elementos no seleccionados. Pero una forma más precisa de medir la diversidad podría involucrar también a los elementos no seleccionados, en la literatura se reportan otras medidas de diversidad que incluyen cálculos de este tipo, como en (3), pero con las cuales el sólo cálculo de la diversidad en un conjunto ya conocido se vuelve un problema complejo, entre estas tenemos la medida del *p*-centro y la medida de Weitzman, que se describen en las secciones 1.3.8 y 1.3.9, y cuyas complejidades de cálculo son de $\mathcal{O}(m\,n)$ y $\mathcal{O}(2^n)$, respectivamente.

1.3.8. La medida de dispersión del p-centro

Esta medida de dispersión puede expresarse de la siguiente forma:

$$div(M) = \max_{1 \le i \le n} \left\{ \min_{j \in M, i \neq j} d_{ij} \right\} \tag{1.9}$$

En contraste con las medidas anteriores la medida del *p*-centro no se calcula únicamente con los elementos seleccionados, sino que requiere de todo el conjunto original. Nótese que en este caso lo que interesa es minimizar la más grande de estas distancias minimales, por lo que podría considerarse la ecuación (1.9) cambiada de signo para que el problema sea de maximización. La evaluación de la ecuación (1.9) requiere de $\mathcal{O}(m\,n)$ operaciones, por lo cual si $m \ll n$, la evaluación de la diversidad sería más costosa que en los casos anteriores.

1.3.9. La medida de dispersión de Weitzman

Esta medida puede expresarse de la forma:

$$div(M) = \max_{j \in M} \{div(V - \{j\}) + D(j, V - \{j\})\} \tag{1.10}$$

Dónde: $D(j, W) = \min_{k \in W} d_{jk}$

En la literatura, (3), se reporta que esta medida es más precisa para la determinación de la diversidad ya que considera todos los posibles subconjuntos de elementos y por tanto engloba todas las relaciones entre elementos seleccionados, no seleccionados y de unos con otros, pero esta característica también implica que no haya una forma eficiente de evaluar la diversidad a través de la ecuación (1.10), ya que su complejidad de cálculo es del orden $\mathcal{O}(2^n)$.

Capítulo 2.

El Problema de la Diversidad Máxima

Una vez que se ha determinado el significado de las "distancias" o relaciones interelemento d_{ij}, cómo éstas pueden ser evaluadas, y las diferentes maneras de estimar la diversidad existente en un conjunto dado, lo siguiente es establecer el problema de optimización que busca determinar el subconjunto con diversidad maximal. A tal problema se lo denomina en la literatura de investigación de operaciones como el *Problema de la Diversidad Máxima*, y consiste en seleccionar un subconjunto de elementos de un conjunto dado de tal manera que una medida de la diversidad o la dispersión presente en el subconjunto seleccionado sea maximizada, véase por ejemplo (10). En la literatura se describen muchos modelos para lograr este fin, así como también se exploran sus múltiples aplicaciones prácticas, como en (4), (5), (6) y (7); en la generalidad de ellos se considera al número de elementos a seleccionar como un valor especificado de antemano, en cambio, en esta investigación consideramos también la posibilidad de que la cardinalidad del subconjunto a seleccionar se obtenga producto de la optimización, de esta manera el número de elementos seleccionados también se vuelve una variable de decisión del problema.

En lo que sigue se revisan los modelos clásicos del problema de la diversidad máxima y sus formulaciones de programación matemática. Se explora la relación que existe entre éstos y se realizan experimentos computacionales para resolver sus formulaciones en programación entera binaria. Se presenta y analiza la experiencia computacional con instancias de diferentes tamaños y características.

Los diferentes modelos que se han propuesto para tratar este problema de optimización combinatoria requieren de una forma de medir la diversidad, por ejemplo con cualquiera de las ecuaciones (1.4), (1.5), (1.7), (1.6), (1.8), (1.6) y (1.10), u otras, las cuales son típicamente una función de distancia en el espacio en el que yacen los objetos, o sus características. La definición de esta distancia entre elementos es personalizada de acuerdo a las aplicaciones específicas, por ejemplo con la establecida en la ecuación (1.3), como se describe en (6). Generalmente el desarrollo de un modelo u otro se ha dado de acuerdo a sus aplicaciones, como por ejemplo:

ubicación de unidades logísticas, problemas sociales, preservación ecológica, control de la polución, diseño de productos, inversión de capitales, gestión de la fuerza de trabajo, diseño de currículum e ingeniería genética, entre otras.

Probablemente el modelo más estudiado de esta clase de problemas es el Problema de la Diversidad Máxima (Maximum Diversity Problem) al que se suele referir como MDP por sus siglas en inglés, en el cual se maximiza la suma de las distancias entre los elementos seleccionados, es decir se maximiza la medida de diversidad de la suma establecida en la ecuación (1.4).

En la literatura el MDP es también conocido con otras denominaciones, como el modelo de la Diversidad Max-Sum como es establecido en (23), como el problema de la Máxima Dispersión (Maximum Dispersion) tal como se lo define en (24), como el problema de la jorga con máximo peso en las aristas (Maximum Edge Weight Clique Problem) definido en (25), como el problema del subgrafo de máximo peso en las aristas (Maximum edge-weighted subgraph) en (8), o como el problema del *k*-subgrafo denso (Dense *k*-subgraph) en (5).

Otro de los problemas que han sido muy documentados en la literatura es el denominado Problema de la Diversidad Max-Min, definido en (4), en el cual se maximiza la mínima distancia entre los elementos del subconjunto seleccionado, es decir, el problema de optimización que resulta de minimizar la ecuación (1.5). Para resolver estos problemas difíciles de optimización combinatoria se han propuesto muchas heurísticas, como las establecidas en (6), y algoritmos basados en meta-heurísticas, como puede verse en (11).

Pero la propia definición de diversidad no es única, ni sencilla, y generalmente es un término cargado de subjetividad, así que además del modelo Max-Sum y Max-Min en la literatura se reportan otros modelos para el Problema de la Diversidad Máxima, de acuerdo con la forma de calcular la diversidad de un conjunto. En particular, recientemente se han propuesto tres modelos adicionales para maximizar la diversidad en el contexto de modelos equitativos, en particular en (18), estos modelos son denominados el Problema de la Máxima Mínima Suma (Max-MinSum), el Problema de la Máxima Dispersión Promedio (Max-Mean) y el Problema de la Mínima Dispersión Diferencial (Min-Diff).

El problema Max-Mean es el problema de optimización que consiste en maximizar la ecuación (1.6), y una de sus principales características, que lo vuelve diferente al resto de modelos

de diversidad reportados en la literatura, es que el número de elementos a seleccionar también es una variable de decisión. Por otro lado, el problema Min-Diff consiste en minimizar la medida de dispersión del diferencial descrita en la ecuación (1.8), y el problema Max-MinSum consiste en maximizar la medida de dispersión de la mínima suma definida en la ecuación (1.7).

Las otras medidas de diversidad definidas en las ecuaciones (1.9) y (1.10) darían origen a otros problemas de optimización, pero no se abordan aquí porque utilizando estas medidas, el solo hecho de medir la diversidad de un subconjunto dado es un problema que tiene gran complejidad, lo que haría al problema de optimización difícil de tratar y daría lugar a otro tema de investigación.

2.1. Formulaciones y Modelos de Programación Matemática

Dado un conjunto $V = \{1, 2, \cdots, n\}$, y la distancia inter-elemento d_{ij} entre cada pareja de elementos de V, el problema es seleccionar un subconjunto $M \subset V$, de cardinalidad $m < n$, que tenga la máxima diversidad:

$$\max_{M \subset V} f_1(M) = div(M) \tag{2.1}$$

La manera en que medimos la diversidad en la ecuación (2.1) permite construir las formulaciones de los diferentes problemas de optimización enunciados antes. Así tenemos las siguientes:

2.1.1. El problema Max-Sum

El problema Max-Sum consiste en:

$$\max_{M \subset V, |M| = m} \sum_{i < j, i, j \in M} d_{ij}$$

Introduciendo las variables binarias:

$$x_i = \begin{cases} 1 \text{ si el elemento } i \text{ es seleccionado} \\ 0 \text{ si no} \end{cases}; 1 \leq i \leq n$$

Entonces, este problema puede ser formulado como un problema de programación cuadrática binaria:

$$\max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j \tag{2.2}$$

$$s.t. \sum_{i=1}^{n} x_i = m \tag{2.3}$$

$$x_i \in \{0,1\};\ 1 \leq i \leq n \tag{2.4}$$

Es claro que esta formulación tiene la estructura de un modelo de programación cuadrática binario (MIQCP), pero la mayoría de los métodos exactos de solución que se proponen en la literatura para resolver este tipo de problemas están basados en la linealización del mismo, y luego con la aplicación de los métodos para resolver problemas lineales enteros 0-1 encontrar el resultado.

El problema puede ser linealizado introduciendo nuevas variables binarias z_{ij} obteniendo la siguiente formulación lineal con variables binarias (10):

$$\max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} z_{ij} \tag{2.5}$$

$$s.t. \sum_{i=1}^{n} x_i = m \tag{2.6}$$

$$z_{ij} \geq x_i + x_j - 1;\ z_{ij} \leq x_i;\ z_{ij} \leq x_j;\ 1 \leq i < j \leq n \tag{2.7}$$

$$x_i,\ z_{ij} \in \{0,1\};\quad 1 \leq i < j \leq n \tag{2.8}$$

2.1.2. El problema Max-Min

El problema Max-Min puede ser formulado como:

$$\max_{M \subset V, |M|=m} \left\{ \min_{i<j, i,j \in M, |M|=m} d_{ij} \right\}$$

Con transformaciones simples, este problema puede ser linealizado y planteado como un problema de programación entera con variables binarias, a través de la siguiente formulación, establecida en (10).

$$\max y \tag{2.9}$$

$$s.t. \sum_{i=1}^{n} x_i = m \tag{2.10}$$

$$z_{ij} \geq x_i + x_j - 1;\ z_{ij} \leq x_i;\ z_{ij} \leq x_j;\ 1 \leq i < j \leq n \tag{2.11}$$

$$y \leq d_{ij} + \left(1 - z_{ij}\right) b_{ij};\ 1 \leq i < j \leq n \tag{2.12}$$

$$x_i,\ z_{ij} \in \{0,1\};\ 1 \leq i < j \leq n \tag{2.13}$$

En la ecuación (2.12) los parámetros b_{ij} son cotas convenientemente elegidas.

2.1.3. El problema Min-Diff

Para formular el problema Min-Diff introducimos, para cada nodo $i \in V$, el valor $d(i)$ como:

$$d(i) = \sum_{j \in M, j \neq i} d_{ij}$$

De esta manera $d(i)$ puede ser interpretado como la dispersión total asociada entre el elemento i y el resto del subconjunto seleccionado. Una forma de garantizar una diversidad equitativa sería minimizar la diferencia entre la más grande dispersión total y la menor dispersión total.

Así, el problema Min-Diff puede ser formulado como:

$$\min_{M \subset V, |M|=m} \{max_{i \in M}\, d(i) - min_{i \in M} d(i)\}$$

Al usar las variables binarias, el modelo de programación matemática del problema Min-Diff es el siguiente:

$$\min \left\{ \max_{i:\, x_i=1} \sum_{j: j \neq i} d_{ij} x_j - \min_{i:\, x_i=1} \sum_{j: j \neq i} d_{ij} x_j \right\}$$

$$s.t. \quad \sum_{i=1}^{n} x_i = m$$

$$x_i \in \{0,1\}; \quad 1 \leq i \leq n$$

Como se muestra en (18), este modelo puede ser linealizado introduciendo la variable $t = \{max_{i \in M}\, d(i) - min_{i \in M} d(i)\}$, obteniéndose la siguiente formulación lineal:

$$min_{t,r,s,x}\, t \tag{2.14}$$

$$s.t. \quad t \geq r - s; \;\; 1 \leq i \leq n \tag{2.15}$$

$$r \geq \sum_{i \in M} d(i) - U_i(1 - x_i) + M^-(1 - x_i); \quad 1 \leq i \leq n \tag{2.16}$$

$$s \leq \sum_{i \in M} d(i) - L_i(1 - x_i) + M^+(1 - x_i); \quad 1 \leq i \leq n \tag{2.17}$$

$$\sum_{i=1}^{n} x_i = m; \; x_i \in \{0,1\}; \;\; 1 \leq i \leq n \tag{2.18}$$

En las ecuaciones (2.16), (2.17) U_i y L_i son cotas superior e inferior convenientemente escogidas para $d(i)$, y M^-, M^+ son constantes grandes.

2.1.4. El problema Max-MinSum

El problema Max-MinSum fue establecido por primera vez en (18), y se define con base en la medida de la diversidad de la mínima suma, descrita en la ecuación (1.7) como sigue:

$$\max_{M\subset V,|M|=n}\left\{\min_{i\in M}\sum_{j\in M,j\neq i}d_{ij}\right\}$$

Introduciendo variables binarias, la formulación de programación matemática del problema es:

$$\max_{x\in\{0,1\}^n}\left\{\min_{i:\,x_i=1}\sum_{j:j\neq i}d_{ij}x_j\right\} \tag{2.19}$$

$$s.t.\ \sum_{i=1}^{n}x_i=m;\ x_i\in\{0,1\},\quad 1\leq i\leq n \tag{2.20}$$

Igual que los modelos anteriores, el modelo Max-MinSum puede ser linealizado y planteado como un problema de programación entera con variables binarias, obteniendo la siguiente formulación:

$$\max\ r \tag{2.21}$$

$$s.t.\ r\ \leq\ \sum_{j:j\neq i}^{n}d_{ij}x_j-L_i(1-x_i)+M^+(1-x_i);\quad 1\leq i\leq n \tag{2.22}$$

$$\sum_{i=1}^{n}x_i=m;\ x_i\in\{0,1\};\quad 1\leq i\leq n \tag{2.23}$$

Como antes, en esta formulación M^+ es una constante positiva grande y L_i son cotas inferiores convenientemente elegidas.

2.1.5. El problema Max-Mean

Este modelo surge de la optimización de la medida de la diversidad descrita en la ecuación (1.6), y al contrario de los problemas anteriores la cardinalidad de M es también una variable de decisión. El problema puede describirse como:

$$\max_{M\subset V,|M|\geq 2}\frac{\sum_{i<j,i,j\in M}d_{ij}}{|M|}$$

Genéricamente hablando, este problema trata de maximizar la diversidad promedio. Una formulación de programación matemática con variables binarias es entonces la establecida por las ecuaciones:

$$\max \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_i x_j}{\sum_{i=1}^{n} x_i} \tag{2.24}$$

$$s.t. \quad \sum_{i=1}^{n} x_i \geq 2 \tag{2.25}$$

$$x_i \in \{0,1\}, \quad 1 \leq i \leq n \tag{2.26}$$

En este problema la función objetivo (2.24) es el promedio de la suma de las distancias entre los elementos seleccionados, la restricción (2.25) indica que por lo menos dos elementos deben seleccionarse. Tal como se presenta en (18), es un problema de optimización binaria fraccional, pero puede ser linealizado utilizando nuevas variables binarias, así el problema es formulado por las ecuaciones (2.27) a (2.32):

$$\max \quad \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}z_{ij} \tag{2.27}$$

$$s.t. \quad y - z_i \leq 1 - x_i; \;\; z_i \leq y; \;\; z_i \leq x_i; \;\; z_i \geq 0; \;\; 1 \leq i \leq n \tag{2.28}$$

$$y - z_{ij} \leq 2 - x_i - x_j; \; z_{ij} \leq y; \;\; z_{ij} \leq x_i; \; z_{ij} \leq x_j; \; z_{ij} \geq 0; \; 1 \leq i < j \leq n \tag{2.29}$$

$$\sum_{i=1}^{n} x_i \geq 2 \tag{2.30}$$

$$\sum_{i=1}^{n} z_i = 1 \tag{2.31}$$

$$x_i \in \{0,1\}; \; 1 \leq i \leq n \tag{2.32}$$

Nótese que no se puede resolver el problema Max-Mean aplicando un método de solución para cualquiera de los otros problemas, a menos que lo apliquemos repetidamente para todos los valores posibles de $m = |M|; m = 2, 3, \ldots, n$. Sorprendentemente, como se verá en la sección

4.1.2, hallar la solución del problema Max-Mean con métodos exactos a través de la estrategia de resolver los $(n-1)$ problemas de tipo Max-Sum requiere mucho menos tiempo que resolver directamente la formulación (2.27)-(2.32) de este problema.

Para finalizar esta sección se puede destacar un hecho importante para las aplicaciones de esta investigación. Consideremos el problema de seleccionar equipos de personas, según muchos estudios recientes que se reportan en la literatura, como en (1) y (13), la eficiencia con la que grupos de personas resuelven problemas es un resultado de la diversidad del grupo. Los dos primeros problemas, el Max-Sum y el Max-Min, se enfocan en objetivos basados en la eficiencia global, con el fin de tener a los elementos escogidos "lejos" unos de otros, lo cual podría resultar en una gran inequidad del sistema resultante por perseguir exclusivamente la eficiencia. Entonces, si estamos conscientes de que también, además de eficiencia, se debe perseguir la equidad, podemos considerar los problemas Max-MinSum, Min-Diff y Max-Mean, que sí consideran de alguna manera el concepto de la justicia en la distribución de los elementos seleccionados, así en el problema Max-Mean no estamos interesados en maximizar la eficiencia total del grupo de personas, sino en maximizar la eficiencia promedio de ellas, y en los problemas Max-MinSum y Min-Diff al introducir la dispersión total asociada entre cada elemento i y el resto del subconjunto seleccionado se asegura de alguna manera la equidad en la solución.

2.2. Complejidad Computacional

Es conocido que el problema Max-Sum y el problema Max-Min son fuertemente NP-duros, como se demuestra en (10), (23) y (26). Recientemente, también se ha demostrado en (18) que el problema Max-Mean es fuertemente NP-duro si las distancias inter-elemento toman valores positivos y negativos; y que en cambio, los problemas Min-Diff y Max-MinSum son fuertemente NP-duros independientemente de los valores de las distancias inter-elemento, como se muestra en la Propiedad 3, Propiedad 4 y Propiedad 5. En esta investigación se demostró la Propiedad 6, que se encuentra en la sección 2.4, la cual indica que si estas distancias son no negativas, simétricas, y además se satisface la desigualdad triangular, entonces la diversidad $div(M)$ para cualquier $M \subset V$ es siempre menor que la diversidad de $M \cup \{k\}$ para cualquier $k \notin M$, luego, una solución con $m < n$ elementos no puede ser óptima en el problema Max-Mean, de ahí que el óptimo en este caso es seleccionar todos los n elementos.

Propiedad 1 (10)

El problema Max-Sum es fuertemente NP-duro.

Propiedad 2 (4)

El problema Max-Min es fuertemente NP-duro.

Propiedad 3:

Si los coeficientes d_{ij} no tienen restricciones en el signo, entonces el problema Max-Mean es fuertemente NP-duro.

Demostración:

Para la demostración consideramos como nuestro problema $\mathcal{P}_0$, al problema de la Máxima Jorga, el cual es conocido que es $NP - completo$, como puede verse en (27).

Sea $G = (V, E)$ un grafo no dirigido, donde V es el conjunto de nodos y E es el conjunto de aristas, donde $|V| = n$ nodos.

Un subconjunto de nodos $C \subseteq V$ se denomina una jorga de G si $\forall\, v_1, v_2 \in C \subseteq V$, existe una arista $(v_1, v_2) \in E$

La versión de decisión del problema de la máxima jorga consiste en chequear si existe una jorga C de tamaño al menos k en G.

Consideremos la siguiente instancia del problema Max-Mean:

$$\max_{x\in\{0,1\}^n,\ x\neq 0} g(x) = \frac{-n^2 \sum_{(i,j)\notin E, i<j} x_i x_j + \sum_{(i,j)\in E, i<j} x_i x_j}{\sum_{i=1}^{n} x_i}$$

Donde:

$$d_{ij} = \begin{cases} 1 & si\ (i,j) \in E \\ -n^2 & si\ (i,j) \notin E \end{cases}$$

Sea $x^* \in \{0,1\}^n, x^* \neq 0$. Definimos un subgrafo inducido $C(V_c, E_C) \subseteq G$, donde:

$$V_c = \{i \in \{1, \dots, n\}: x_i^* = 1\}$$

Si C no es una jorga: $x_i^* = 1, x_j^* = 1$ implica que $(i,j) \notin E$, entonces $g(x^*) < 0$.

Por el contrario si C es una jorga: $\forall\ (i,j)$ tal que $x_i^* = 1, x_j^* = 1$ se tiene $(i,j) \in E$, y entonces:

$$g(x^*) = \frac{|C|(|C|-1)}{2|C|} = \frac{|C|-1}{2} \geq 0$$

Por lo tanto, el grafo G contiene una jorga de tamaño al menos k si y sólo si:

$$\max_{x \in \{0,1\}^n,\ x \neq 0} g(x) \geq \frac{k-1}{2}$$

■

Propiedad 4 (18):

El problema Min-Diff es fuertemente NP-duro.

Propiedad 5 (18)

El problema Max-MinSum es fuertemente NP-duro.

2.3. Correlación entre los problemas

Aunque todos los modelos planteados buscan determinar el subconjunto de diversidad máxima, al ser el concepto de diversidad interpretado de diferente manera en cada uno de ellos, no se puede deducir a priori si existe o no algún tipo de correlación entre sus soluciones. Es decir, si para un modelo un subconjunto es el más diverso, que tan probable será que también lo sea para otro modelo. Este análisis tiene sentido para los problemas Max-Sum, Max-Min, Min-Diff y Max-MinSum, ya que el problema Max-Mean tiene una estructura muy diferente por el hecho de que en éste el número de elementos a seleccionar no es una restricción, sino que es resultado del proceso de optimización.

En la literatura se encuentran pocos resultados sobre la correlación entre las soluciones de los problemas Max-Sum y Max-Min, como en (28), en donde se reporta un análisis de correlación para una instancia con $n = 7$ elementos y $m = 5$ elementos a seleccionar. El resultado es una correlación de 52% entre todas las $\binom{n}{m}$ soluciones factibles, lo cual podría ser muy bajo como para considerar cierta equivalencia entre los dos problemas. Pero este análisis es muy particular, aquí se presenta un análisis más exhaustivo de la correlación entre las soluciones de los diferentes problemas.

En esta sección siguiente se realiza en primer lugar un análisis de la correlación que existe entre los problemas Max-Sum, Max-Min, Max-MinSum y Min-Diff con varias instancias de tamaño mediano. Luego se presenta un estudio sobre la relación entre el problema Max-Sum y el problema Max-Mean, que es directa. Con este propósito se realizaron los siguientes experimentos numéricos:

Considerando dos escenarios posibles para la naturaleza de las distancias inter-elemento d_{ij}, el primero asumiendo no negatividad y el segundo sin restricciones en el signo, se generaron aleatoriamente en cada escenario 30 instancias con $|V| = n = 20\ y\ |M| = m = 5$. Para cada instancia se exploraron todos los $\binom{20}{5}$ subconjuntos posibles de cardinalidad 5 tomados del conjunto de 20 elementos, es decir para cada ejemplo se generaron 15,504 subconjuntos. Y para cada uno de estos subconjuntos generados se calculó el valor de las cuatro medidas de diversidad (1.4), (1.5), (1.7) y (1.8) que se proponen.

Se analizaron tres parámetros: peor valor, promedio, y mejor valor. De esta manera se obtuvieron 30 valores para cada uno de estos parámetros y para cada escenario, valores con los cuales se realizó posteriormente un análisis estadístico de correlación.

Además, para efectos de analizar la influencia de la naturaleza de los d_{ij} sobre la correlación de los modelos se consideraron dos escenarios para la generación aleatoria de las instancias, que fueron los siguientes:

- Escenario I: Los d_{ij} seleccionados aleatoriamente de una distribución U[0,20]
- Escenario II: Los d_{ij} seleccionados aleatoriamente de una distribución U[-10,10]

La razón para escoger estos dos tipos de escenarios es que tratamos de analizar si esta correlación posible entre los diferentes modelos depende de la estructura de la matriz de distancias inter-elemento d_{ij}, para lo cual consideramos estas dos situaciones generales: en el primer caso, el Escenario I está constituido con valores de d_{ij} no negativos, y en el segundo caso en el Escenario II consideramos valores positivos y negativos.

El análisis de correlación mostró interesantes conclusiones sobre el comportamiento de los modelos con las cuatro medidas de dispersión propuestas en cada uno de los escenarios, constatándose que en el primer escenario los modelos Max-Sum, Max-MinSum y Max-Min están más correlacionados en sus parámetros valor promedio y mejor valor, tal como se observa en la Tabla 6, mientras que en el caso de considerar los valores de d_{ij} sin restricción en el signo, es decir en el escenario II, la conclusión es que existe únicamente correlación significativa en estos parámetros entre el modelo Max-Sum y el modelo Max-MinSum, como consta en la Tabla 7.

Por otro lado, no se observan correlaciones significativas en los dos escenarios entre en modelo Min-Diff y los otros modelos, por lo que se puede concluir que este problema es muy diferente que el resto. Tampoco se observan valores significativos para ninguno de los cuatro modelos respecto al parámetro peor valor, pero hay que observar que para fines prácticos este parámetro (el peor resultado) no es de mucha utilidad, mientras que los otros parámetros, el mejor valor y valor promedio, si son de interés práctico.

A pesar de estos valores observados en los coeficientes de correlación, que pueden parecer no muy altos al comparar los modelos Max-Sum, Max-Min y Max-MinSum, se debe resaltar el hecho de que el número de veces en que las soluciones de estos modelos coincidieron en exactamente el mismo subconjunto M seleccionado es bastante alto, para ambos escenarios.

Así, para las 30 instancias que se probaron en cada uno de los escenarios, el número de veces en que las soluciones coincidieron exactamente en la misma solución se muestra en las Tabla 8 y Tabla 9, se observa que no hay diferencias significativas para este hecho entre los dos escenarios.

Tabla 6. Correlaciones entre los mejores, promedio y peores resultados en 30 instancias con $d_{ij} \in$ U[0, 20], n =20, m= 5

		Max-Sum	Max-Min	Max-MinSum	Min-Diff
PEOR	Max-Sum	1	0.5081555	0.3069869	-0.3454331
	Max-Min	0.5081555	1	0.5805362	-0.3007265
	Max-MinSum	0.3069869	0.5805362	1	-0.4447183
	Min-Diff	-0.3454331	-0.3007265	-0.4447183	1
PROMEDIO	Max-Sum	1	0.5197526	0.9562924	0.02085051
	Max-Min	0.5197526	1	0.6927929	-0.63859032
	Max-MinSum	0.95629241	0.6927929	1	-0.263086
	Min-Diff	0.02085051	-0.6385903	-0.263086	1
MEJOR	Max-Sum	1	0.644131969	0.78511593	0.043505127
	Max-Min	0.64413197	1	0.66352037	-0.004005883
	Max-MinSum	0.78511593	0.663520371	1	-0.027300016
	Min-Diff	0.04350513	-0.004005883	-0.02730002	1

Tabla 7. Correlación entre el mejor, el peor y el resultado promedio en 30 instancias con $d_{ij} \in U[-10, 10]$, n =20, m= 5

		Max-Sum	Max-Min	Max-MinSum	Min-Diff
PEOR	Max-Sum	1	-0.1510322	0.627419	-0.2525618
	Max-Min	-0.1510322	1	0.3605734	-0.2215364
	Max-MinSum	0.627419	0.3605734	1	-0.4193284
	Min-Diff	-0.2525618	-0.2215364	-0.4193284	1
PROMEDIO	Max-Sum	1	0.6871436	0.9670816	-0.2386394
	Max-Min	0.6871436	1	0.7525095	-0.5559718
	Max-MinSum	0.9670816	0.7525095	1	-0.4688571
	Min-Diff	-0.2386394	-0.5559718	-0.4688571	1
MEJOR	Max-Sum	1	0.4130649	0.77181628	-0.15287194
	Max-Min	0.4130649	1	0.13861263	-0.1440507
	Max-MinSum	0.7718163	0.1386126	1	-0.03317477
	Min-Diff	-0.1528719	-0.1440507	-0.03317477	1

Tabla 8. Número de veces en las cuales las soluciones de los modelos coincidieron entre las 30 instancias probadas.

Escenario I

	Max-Sum	Max-Min	Max-MinSum	Min-Diff
Max-Sum	-	6	13	0
Max-Min	-	-	6	1
Max-MinSum	-	-	-	1

Tabla 9. Número de veces en las cuales las soluciones de los modelos coincidieron entre las 30 instancias probadas.

Escenario II

	Max-Sum	Max-Min	Max-MinSum	Min-Diff
Max-Sum	-	3	10	0
Max-Min	-	-	6	0
Max-MinSum	-	-	-	0

Si queremos tener una idea geométrica de las soluciones que arrojan los cuatro modelos podemos construir instancias Euclidianas para observar cómo están repartidos los elementos del

conjunto original y cómo quedan esparcidos los elementos seleccionados, la Figura 12 corresponde a las soluciones óptimas (puntos en rojo) con los cuatro modelos considerados, el ejemplo corresponde a una instancia Euclidiana con $n = 25$ y $m = 7$ y las posiciones de los puntos que se observan en la figura han sido generadas aleatoriamente en un cuadrado 100 X 100. En este caso los resultados ya no se pueden obtener por exploración exhaustiva del espacio de soluciones factibles ya que el número de soluciones factibles en este problema es $\binom{25}{7} = 480{,}700$, para obtener las soluciones óptimas se resolvieron las formulaciones lineales planteadas en las secciones 2.1.1, 2.1.2, 2.1.3 y 2.1.4, para lo cual se utilizó el solver Cplex 12.0.

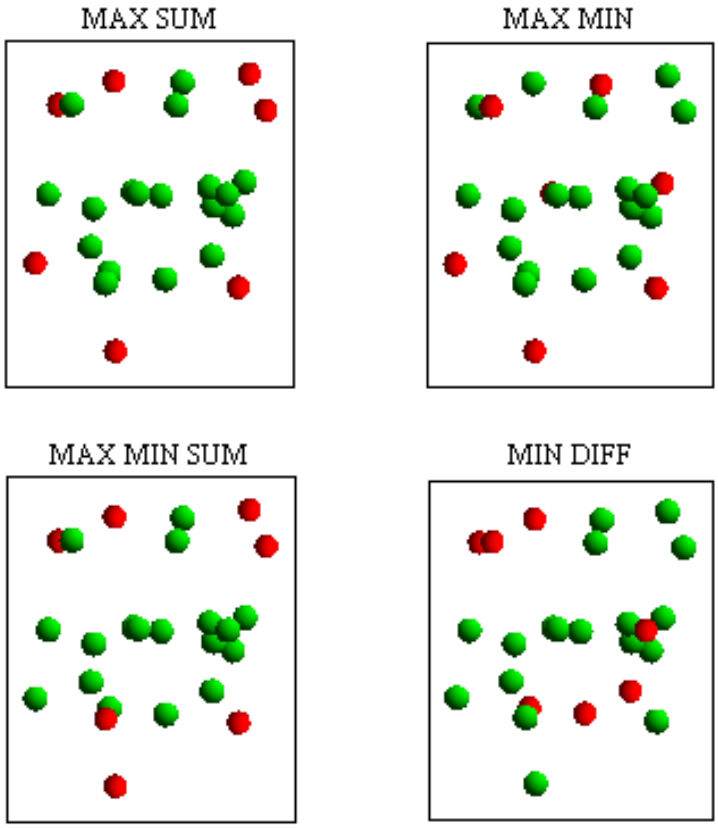


Figura 12. Soluciones de los modelos Max-Sum, Max-Min, Max-MinSum y Min-Diff para una instancia euclidiana con *n*=25, *m*=7

A partir de este análisis podemos concluir que si bien es cierto los cuatro modelos del problema de la diversidad máxima investigados tienen cierto grado de correlación y coinciden algunas veces en la solución óptima, también muchas veces dan soluciones muy diferentes, y la

solución obtenida por un modelo puede tener pobres resultados en otro modelo. De esta manera podemos decir que los modelos no son equivalentes y por tanto se requiere aplicar el modelo más conveniente en función del problema concreto que se está abordando, analizando cuál es el modelo de medida de diversidad que más se adapta a la realidad dada. Desde ese punto de vista tiene sentido plantear métodos de solución adaptados a cada modelo específico, tanto exactos como heurísticos. En la sección 4.1 se analiza la estrecha relación que existe entre el problema Max-Sum y el problema Max-Mean, lo cual permitirá diseñar métodos eficientes para resolver este último problema.

2.4. Propiedades del Problema del Máximo Promedio Max-Mean

En esta sección se demuestra una importante propiedad del problema Max-Mean, la cual será crucial en el diseño los problemas de prueba con los que se efectuarán los experimentos computacionales de los algoritmos diseñados para resolverlo.

Como se introdujo en la sección 2.1.5, el modelo Max-Mean del problema de la diversidad máxima consiste en:

$$\max \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_i x_j}{\sum_{i=1}^{n} x_i}$$

$$s.t. \quad \sum_{i=1}^{n} x_i \geq 2$$

$$x_i \in \{0,1\}, \quad 1 \leq i \leq n$$

Una propiedad interesante de este problema es la Propiedad 6, que indica que el modelo tiene solución trivial bajo determinadas características de las distancias inter-elemento d_{ij}.

Propiedad 6

El problema Max-Mean tiene la solución trivial $M = V$, si la relación inter-elemento d_{ij} es una métrica; es decir, si satisface las propiedades de no negatividad, transitividad y simetría.

Demostración:

De acuerdo a la ecuación (1.6), el problema Max-Mean consiste en seleccionar un subconjunto M de V tal que la diversidad $div(M)$ sea maximizada. Demostraremos que dada una instancia en la que d_{ij} son no negativas, simétricas y satisfacen la desigualdad triangular, la solución al problema Max-Mean es seleccionar todos los elementos, es decir: $M = V$.

Para todo $i, j \in M$ y dado $k \notin M$ la desigualdad triangular establece que $d_{ij} \leq d_{ik} + d_{jk}$

Sumando sobre todas las posibles parejas de elementos en M obtenemos:

$$\sum_{\substack{i,j \in M \\ i<j}} d_{ij} \leq \sum_{\substack{i,j \in M \\ i<j}} \left(d_{ik} + d_{jk}\right)$$

Pero el lado derecho de la última expresión es equivalente a $(|M| - 1)$ veces $\sum_{i \in M} d_{ik}$,

Si representamos con $m = |M|$, entonces:

$$\sum_{\substack{i,j \in M \\ i<j}} d_{ij} \leq \sum_{\substack{i,j \in M \\ i<j}} \left(d_{ik} + d_{jk}\right) = (m-1) \sum_{i \in M} d_{ik} < m \sum_{i \in M} d_{ik}$$

Dividiendo para m se tiene:

$$\frac{1}{m} \sum_{\substack{i,j \in M \\ i<j}} d_{ij} < \sum_{i \in M} d_{ik}$$

Sumando el término $\sum_{\substack{i,j \in M \\ i<j}} d_{ij}$ a ambos lados de la última desigualdad:

$$\frac{m+1}{m} \sum_{\substack{i,j \in M \\ i<j}} d_{ij} < \sum_{\substack{i,j \in M \\ i<j}} d_{ij} + \sum_{i \in M} d_{ik} = \sum_{\substack{i,j \in M \cup \{k\} \\ i<j}} d_{ij}$$

Y finalmente dividiendo para $(m + 1)$:

$$div(M) = \frac{1}{m}\sum_{\substack{i,j\in M\\ i<j}} d_{ij} < \frac{1}{m+1}\sum_{\substack{i,j\in M\cup\{k\}\\ i<j}} d_{ij} = div(M \cup \{k\})$$

Con lo que se demuestra que cualquiera que sea el elemento que se añada a un subconjunto seleccionado, la divergencia siempre será mayor, por tanto efectuando el procedimiento iterativamente terminan seleccionándose todos los elementos y el conjunto de diversidad maximal es el propio V.

■

Observación:

Aunque en (18) se demuestra que el problema Max-Mean es NP-duro cuando d_{ij} son irrestrictas en el signo, si consideramos sólo distancias no negativas y simétricas, pero se relaja la desigualdad triangular la solución ya no es necesariamente la solución trivial $M = V$. Como observamos en el siguiente ejemplo:

Sea el conjunto $V = \{1,2,3,4\}$ y matriz de distancias d_{ij}:

$$d = \begin{pmatrix} 0 & 20 & 18 & 1 \\ 20 & 0 & 20 & 2 \\ 18 & 20 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}$$

La solución óptima del problema Max-Mean es escoger el subconjunto $M^* = \{1,2,3\}$, con valor de la función objetivo $div(M^*) = 19.33$,

En cambio si seleccionamos todos los elementos $M = V$, el valor de la función objetivo es $div(V) = 15.5$.

En el Capítulo 4 se tendrá en cuenta esta propiedad para generar problemas de prueba adecuados para el desarrollo de los experimentos computacionales respectivos.

Capítulo 3.

Procedimientos Metaheurísticos en Optimización Combinatoria

El carácter computacionalmente difícil del problema de la diversidad máxima, determinado por el carácter NP-duro de sus diferentes variantes, hace que para resolver eficientemente casos de tamaños medianos o grandes, se tengan que utilizar técnicas heurísticas o aproximadas.

Los métodos descritos en este capítulo se denominan algoritmos metaheurísticos, o sencillamente heurísticos. El término heurístico deriva de la palabra griega "heuriskein" que significa encontrar o descubrir, y generalmente se usa en el ámbito de la optimización para describir una clase de algoritmos para la resolución de problemas considerados difíciles.

3.1. Métodos heurísticos simples

En el contexto científico la optimización es el proceso que se realiza para encontrar la mejor solución para un determinado problema. En un problema de optimización existen diferentes soluciones, denominadas soluciones factibles, el criterio para discriminar entre ellas está dictaminado por la llamada función objetivo. De forma más precisa, un problema de optimización se puede caracterizar como un problema en el cuál se deben encontrar los valores de unas variables de decisión para los que una determinada función objetivo alcanza su valor máximo o mínimo. El valor de las variables en ocasiones está sujeto a unas restricciones. Algunas veces los problemas de optimización son relativamente fáciles de resolver, como es el caso de los problemas de programación lineal, pero para una gran cantidad de aplicaciones prácticas los problemas de optimización son muy difíciles de resolver.

La idea intuitiva de un problema "difícil de resolver" queda reflejada en el término NP-duro, que puede consultarse en el ANEXO 2, utilizado en el contexto de la complejidad computacional. En términos muy superficiales podemos decir que un problema de optimización difícil es aquel para el que no podemos garantizar encontrar la mejor solución posible, es decir la óptima, en un tiempo razonable. Y como en la práctica se verifica la existencia de una gran

variedad de problemas difíciles que necesitan ser resueltos de forma eficiente, se han tenido que desarrollar procedimientos para encontrar rápidamente buenas soluciones aunque no fueran las óptimas. Estos métodos, en los que la rapidez del proceso es tan importante como la calidad de la solución obtenida, se denominan heurísticos o aproximados, de esta manera, en contraposición a los métodos exactos que proporcionan una solución óptima del problema, los métodos heurísticos se limitan a proporcionar en un tiempo razonable una buena solución del problema no necesariamente óptima. En (29) se recogen varias definiciones diferentes de algoritmo heurístico, entre las que se destaca la siguiente:

> "Un método heurístico es un procedimiento para resolver un problema de optimización mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución."

Aunque hemos mencionado el caso de la resolución de un problema difícil, existen otras razones para utilizar métodos heurísticos, entre las que podemos destacar:

- El problema es de una naturaleza tal que no se conoce ningún método exacto para su resolución.
- Aunque existe un método exacto para resolver el problema, su uso es computacionalmente muy costoso.
- El método heurístico es más flexible que un método exacto, permitiendo, por ejemplo, la incorporación de condiciones de difícil modelización.
- El método heurístico se utiliza como parte de un procedimiento global que garantiza el óptimo de un problema, bien para proporcionar una buena solución inicial de partida, o en un paso intermedio del procedimiento, como por ejemplo las reglas de selección de la variable a entrar en la base en el método Simplex.

En contraposición con otros métodos de resolución de propósito general, en los que el procedimiento es en gran medida independiente del problema abordado, los algoritmos heurísticos dependen mucho del problema concreto para el que se han diseñado; es decir, las técnicas e ideas aplicadas a la resolución de un problema son específicas de éste y aunque, en general, pueden ser trasladadas a otros problemas, han de particularizarse en cada caso.

3.2. Indicadores de calidad de un algoritmo heurístico

Al resolver un problema de forma heurística debemos medir la calidad de los resultados puesto que, como ya hemos mencionado, la optimalidad no está garantizada. En esta sección se recogen los principales métodos que se usan para medir la calidad y eficiencia de un algoritmo y poder determinar su valía frente a otros. En general un buen algoritmo heurístico debe de tener las siguientes propiedades:

- Eficiente. El esfuerzo computacional debe ser realista para obtener la solución.
- Bueno. La solución encontrada debe de estar, en promedio, cerca del óptimo.
- Robusto. La probabilidad de obtener una mala solución debe ser baja.

Para medir la calidad de un heurístico existen diversos procedimientos, entre los que se encuentran los siguientes:

3.2.1. Comparación con la solución óptima

Aunque normalmente se recurre al algoritmo aproximado por no existir un método exacto para obtener el óptimo, o por ser éste computacionalmente muy costoso, en ocasiones puede ser que se disponga de un procedimiento que proporcione el óptimo para un conjunto limitado de ejemplos (usualmente de tamaño reducido). Este conjunto de ejemplos puede servir para medir la calidad del método heurístico.

Para ello se calcula, para cada uno de los ejemplos, la desviación porcentual de la solución heurística frente a la óptima, determinando posteriormente el promedio de dichas desviaciones. Si llamamos c_h al valor de la función objetivo obtenida por medio del algoritmo heurístico y c_{opt} al valor de la función objetivo en la solución óptima de un ejemplo dado, en un problema de minimización o de maximización, la desviación porcentual, también conocida como *GAP*, de la palabra inglesa “brecha”, viene dada por la expresión:

$$GAP = \frac{|c_h - c_{opt}|}{c_{opt}} \, 100\%$$

3.2.2. Comparación con una cota

En ocasiones el óptimo del problema no está disponible ni siquiera para un conjunto limitado de ejemplos. Un método alternativo de evaluación consiste en comparar el valor de la solución que proporciona el heurístico con una cota del problema (inferior si es un problema de minimización y superior si es de maximización). Obviamente la bondad de esta medida dependerá de la bondad de la cota (cercanía de ésta al óptimo), por lo que, de alguna manera, tendremos que tener información de lo buena que es dicha cota. En caso contrario la comparación propuesta no tiene demasiado interés.

3.2.3. Comparación con un método exacto truncado

Un método enumerativo como el de Ramificación y Acotación explora una gran cantidad de soluciones, aunque sea únicamente una fracción del total, por lo que los problemas de grandes dimensiones pueden resultar computacionalmente inabordables con estos métodos. Sin embargo, podemos establecer un límite de iteraciones (o de tiempo) máximo de ejecución para el algoritmo exacto. También podemos saturar un nodo en un problema de maximización cuando su cota inferior sea menor o igual que la cota superior global más un cierto α (análogamente para el caso de minimizar). De esta forma se garantiza que el valor de la mejor solución proporcionada por el procedimiento no dista más de α del valor óptimo del problema. En cualquier caso, la mejor solución encontrada con estos procedimientos truncados proporciona una cota con la que contrastar el heurístico.

3.2.4. Comparación con otros métodos heurísticos

Este es uno de los métodos más empleados en problemas difíciles (NP-duros) sobre los que se ha trabajado durante mucho tiempo y para los que se conocen algunos buenos métodos heurísticos. Al igual que ocurre con la comparación con las cotas, la conclusión de dicha comparación está en función de la bondad del heurístico escogido. En el marco de esta investigación se compara la heurística propuesta con heurísticas previas que son eficientes en la resolución de problemas similares.

3.2.5. Análisis del peor caso

Uno de los métodos que durante un tiempo tuvo bastante aceptación es aquel que consistía en analizar el comportamiento en el peor caso del algoritmo heurístico; esto es, considerar los ejemplos que sean más desfavorables para el algoritmo y acotar analíticamente la máxima desviación respecto del óptimo del problema. Lo mejor de este método es que acota el resultado del algoritmo para cualquier ejemplo; sin embargo, por este mismo motivo, los resultados no suelen ser representativos del comportamiento medio del algoritmo. Además, el análisis puede ser muy complicado para los heurísticos más sofisticados.

3.3. Clasificación de los métodos heurísticos: constructivos y de búsqueda local

Existen muchos métodos heurísticos, y de naturaleza muy diferente, por lo que es complicado dar una clasificación completa. Además, muchos de ellos han sido diseñados para un problema específico sin posibilidad de generalización o aplicación a otros problemas similares. Sin embargo, generalmente es aceptado categorizar a los heurísticos en: métodos constructivos y métodos de búsqueda local.

3.3.1. Métodos Constructivos:

Los métodos constructivos son procedimientos iterativos que, en cada paso, añaden un elemento a la solución parcial en construcción, hasta completar una solución. Usualmente estos métodos son deterministas y están basados en seleccionar, en cada iteración, el elemento con mejor evaluación. Estos métodos son muy dependientes del problema que resuelven, y literalmente ensamblan paso a paso una solución del problema.

3.3.2. Métodos de Búsqueda Local:

A diferencia de los métodos anteriores, los procedimientos de búsqueda o mejora local comienzan con una solución del problema, construida por algún método, y la mejoran progresivamente. En resumen estos métodos consisten en pasar, en cada iteración, desde una solución a otra solución con mejor valor por medio de una operación básica denominada

"movimiento". El método finaliza cuando, para una solución, no existe un movimiento que permita pasar a una solución accesible que la mejore. Y al igual que ocurría con los métodos constructivos, estos algoritmos son muy dependientes del problema que resuelven. Más formalmente se pueden describir estos métodos introduciendo las siguientes definiciones:

Definición: Sea X el conjunto de soluciones del problema combinatorio. Cada solución $x \in X$ tiene un conjunto de soluciones asociadas $N(x) \subseteq X$, que se denomina *entorno* o *vecindad* de x.

Definición: Dada una solución x, se denomina *movimiento* a cualquier operación básica por la cual podemos obtener directamente una solución de su entorno, $x' \in N(x)$, a partir de x.

Entonces, un método de búsqueda local parte de una solución inicial x_0 , calcula su entorno $N(x_0)$ y escoge una nueva solución con mejor valor en la función objetivo, $x_1 \in N(x_0)$. Es decir, realiza el movimiento m_1 que aplicado a x_0 produce como resultado una mejor solución x_1. Este proceso se aplica iterativamente hasta que ya no existe un elemento del entorno que mejora el valor de la función objetivo, tal como se esquematiza en la Figura 13.

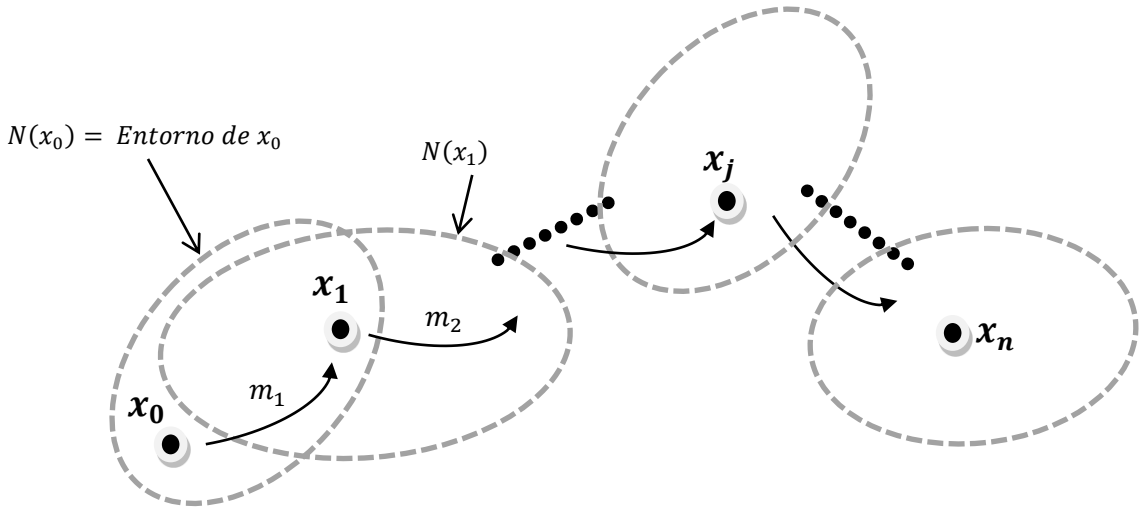


Figura 13. Esquema de funcionamiento de los procedimientos de Búsqueda Local

Así, un procedimiento de búsqueda local queda completamente determinado al especificar que es un movimiento, lo cual equivale a especificar que es un entorno o vecindad, y el criterio de selección de una solución dentro del entorno. Esta definición de movimiento o entorno,

depende en gran medida de la estructura del problema a resolver, así como de su función objetivo. También se pueden definir diferentes criterios para seleccionar una nueva solución del entorno. Uno de los criterios más simples consiste en tomar la solución con mejor evaluación de la función objetivo, siempre que la nueva solución sea mejor que la actual. Este criterio, conocido como voraz, permite ir mejorando la solución actual mientras se pueda. El algoritmo se detiene cuando la solución no puede ser mejorada. La solución encontrada será por tanto un óptimo local respecto al entorno definido.

El óptimo local alcanzado no puede mejorarse mediante el movimiento definido. Sin embargo, el método empleado no permite garantizar, de ningún modo, que sea el óptimo global del problema. Más aún, dada la "miopía" de la búsqueda local, es de esperar que en general no lo sea.

3.4. Métodos Combinados

En los apartados anteriores se han caracterizado los métodos constructivos, que obtienen una solución del problema, y los métodos de mejora que, a partir de una solución inicial, tratan de obtener nuevas soluciones con mejor valor. Es evidente que ambos métodos pueden combinarse, tomando los segundos como solución inicial la obtenida con los primeros. En esta sección se analizan algunas variantes y mejoras sobre tal esquema.

Como se ha visto, una de las limitaciones más importantes de los métodos heurísticos es la denominada miopía provocada por seleccionar, en cada paso, la mejor opción. Como una opción alternativa, resulta muy provocativo elegir al azar el elemento a seleccionar; sin embargo, es evidente que el tomar una opción al azar como norma puede conducirnos a cualquier resultado, por lo que parece más adecuado recurrir a algún procedimiento sistemático que conjugue la evaluación con el azar. Dos de los procedimientos más utilizados en la literatura son los siguientes.

3.4.1. Procedimientos Aleatorizados

Una modificación en los algoritmos de construcción consiste en sustituir la elección voraz por una elección al azar de entre un conjunto de buenos candidatos. Así, en cada paso del

procedimiento, se evalúan los elementos que pueden ser añadidos y se selecciona el subconjunto de los mejores. La elección se realiza al azar sobre el subconjunto de buenos candidatos.

Existen varias maneras de establecer el subconjunto de mejores candidatos. En un problema de maximización, en donde cuanto mayor es la evaluación de una opción más "atractiva" resulta, podemos destacar:

- Establecer un número fijo *k* para el subconjunto de mejores candidatos e incluir los *k* mejores.
- Establecer un valor umbral e incluir en el conjunto todos los elementos cuya evaluación esté por encima de dicho valor. Incluir siempre el mejor de todos.
- Establecer un porcentaje respecto del mejor, e incluir en el subconjunto todos aquellos elementos cuya evaluación difiere, de la del mejor en porcentaje, en una cantidad menor o igual que la establecida.

En todos los casos la elección final se realiza al azar de entre los preseleccionados.

Una estrategia alternativa a la anterior consiste en considerar las evaluaciones como pesos y utilizar un método probabilístico para seleccionar una opción. Así, si $v_1, v_2, \dots, v_k$ son los posibles elementos a añadir en un paso del algoritmo, se calculan sus evaluaciones $e_1, e_2, \dots, e_k$, y se les asigna un intervalo del siguiente modo:

Elemento	**Evaluación**	**Intervalo**
v_1	e_1	$[0, e_1[$
v_2	e_2	$[e_1, e_1 + e_2[$
…	…	…
v_k	e_k	$\left[\sum_{i=1}^{k-1} e_i, \sum_{i=1}^{k} e_i\right]$

Luego se genera un número a al azar entre 0 y $\sum_{i=1}^{k} e_i$, y se selecciona el elemento correspondiente al intervalo que contiene a a.

Al algoritmo constructivo modificado, tanto con la opción primera como con la segunda, se lo denomina Algoritmo Constructivo Aleatorizado. Puede ser que este algoritmo produzca una solución de peor calidad que la del algoritmo original; sin embargo, dado que el proceso no es determinista, cada vez que lo realicemos sobre un mismo ejemplo obtendremos resultados diferentes. Esto permite definir un proceso iterativo consistente en ejecutar un número prefijado de veces (MAX_ITER) el algoritmo y quedarnos con la mejor de las soluciones obtenidas en todas las iteraciones. Obviamente, cada una de dichas soluciones puede mejorarse con un algoritmo de búsqueda local. El siguiente procedimiento incorpora el algoritmo de mejora a dicho esquema.

Algoritmo combinado aleatorizado

Inicialización

Obtener una solución con el algoritmo constructivo aleatorizado.

Sea c^* el coste de dicha solución.

Hacer $i = 0$

Mientras ($\boldsymbol{i < MAX\_ITER}$)

Obtener una solución $x(i)$ con el algoritmo constructivo aleatorizado.

Aplicar el algoritmo de búsqueda local a $x(i)$.

Sea $x^*(i)$ la solución obtenida y $S^*(i)$ su valor.

Si ($S^*(i)$ mejora a c^*) Hacer $c^* = S^*(i)$ y guardar la solución actual

$i = i + 1$

En cada iteración el algoritmo construye una solución y después trata de mejorarla. Así, en la iteración i, el algoritmo construye la solución $x(i)$ con valor $S(i)$ y posteriormente la mejora obteniendo $x^*(i)$ con valor $S^*(i)$. Hay que notar que $x^*(i)$ puede ser igual a $x(i)$ si el algoritmo de mejora local no encuentra ningún movimiento que mejore la solución.

Después de un determinado número de iteraciones es posible estimar el porcentaje de mejora obtenido por el algoritmo en su fase de mejora local y utilizar esta información para

aumentar la eficiencia del procedimiento. En concreto, al construir una solución se examina su valor y se puede considerar que la fase de mejora local mejoraría la solución en un porcentaje similar al observado en promedio. Si el valor resultante queda alejado del valor de la mejor solución encontrada hasta el momento, podemos descartar la solución actual y no realizar la fase de mejora local del algoritmo, con el consiguiente ahorro computacional.

3.4.2. Métodos Multi - Arranque

Los métodos Multi-Arranque, también llamados Multi-Start, generalizan el esquema anterior. Tienen dos fases: la primera en la que se genera una solución y la segunda en la que la solución es típicamente, pero no necesariamente, mejorada. Cada iteración global produce una solución, usualmente un óptimo local, y la mejor de todas es la salida del algoritmo.

Algoritmo Multi-Arranque

Mientras (Condición de parada)

Fase de Generación

Construir una solución.

Fase de Búsqueda

Aplicar un método de búsqueda para mejorar la solución construida

Actualización

Si la solución obtenida mejora a la mejor almacenada, actualizarla.

Dada su sencillez de aplicación, estos métodos han sido muy utilizados para resolver gran cantidad de problemas. En el contexto del problema de la máxima diversidad, se pueden encontrar últimamente algunos trabajos tanto teóricos como aplicados. En (9), Lozano y otros presentan una metaheurística voraz iterada bastante simple que genera una sucesión de soluciones iterando sobre una heurística constructiva voraz usando fases de construcción y destrucción que permite resolver instancias grandes del problema de la máxima diversidad Max-Sum descrito en la sección 2.1.1, mientras que en (30) Aringhieri propone un algoritmo de Multi-

arranque aleatorio que tiene un buen rendimiento al resolver el mismo problema Max-Sum. Las primeras aplicaciones en el ámbito de la optimización combinatoria consistían en métodos sencillos de construcción, completa o parcialmente aleatorios, y su posterior mejora con un método de búsqueda local. Sin embargo el mismo esquema permite sofisticar el procedimiento basándolo en unas construcciones y/o mejoras más complejas. Numerosas referencias y aplicaciones se pueden encontrar en (31).

Uno de los artículos que contiene las ideas en que se basa el método de Búsqueda Tabú, véase (17), también incluye aplicaciones de estas ideas para los métodos de multi-arranque. Básicamente se trata de almacenar la información relativa a soluciones ya generadas y utilizarla para la construcción de nuevas soluciones. Es en este contexto en que se introducen las estructuras de memoria reciente y frecuente.

Un problema para diseñar un buen procedimiento de búsqueda basada en multi- arranque es cómo decidir si es preferible implementar un procedimiento de mejora sencillo que permita realizar un gran número de iteraciones globales o, alternativamente, aplicar una rutina más compleja que mejore significativamente unas pocas soluciones generadas. Un procedimiento sencillo depende fuertemente de la solución inicial pero un método más elaborado consume mucho más tiempo de computación y, por tanto, puede ser aplicado pocas veces, reduciendo la exploración del espacio de soluciones. Esta limitación de la estrategia voraz es el punto de partida de los procedimientos metaheurísticos basados en búsqueda local: evitar el quedar atrapados en un óptimo local lejano del global. Para lo cual se hace necesario utilizar movimientos que empeoren la función objetivo. Sin embargo esto plantea dos problemas. El primero es que al permitir movimientos de mejora y de no mejora, el procedimiento se puede ciclar, revisitando soluciones ya vistas, por lo que habría que introducir un mecanismo que lo impida. El segundo es que hay que establecer un criterio de parada ya que un procedimiento de dichas características podría iterar indefinidamente. En este sentido son los procedimientos metaheurísticos los que incorporan mecanismos sofisticados para solucionar eficientemente ambas cuestiones, así como también para tratar, en la medida de lo posible, de dirigir la búsqueda de forma inteligente. En la sección 3.5 se introducen los conceptos referentes a estas técnicas.

3.5. Metaheurísticas

Si bien todos estos métodos han contribuido a ampliar nuestro conocimiento para la resolución de problemas reales, los métodos constructivos y los de búsqueda local constituyen la base de los procedimientos denominados Metaheurísticos, término que fue introducido por Fred Glover en 1986, en (29), para referirse a ellos. El término metaheurísticas se obtiene de anteponer a la palabra heurística el prefijo meta, que significa “más allá" o “a un nivel superior". Las concepciones actuales de lo que es una metaheurística están basados en las diferentes interpretaciones de lo que es una forma inteligente de resolver un problema.

Así, estos métodos fueron desarrollados con el propósito de obtener mejores resultados que los alcanzados por los métodos heurísticos tradicionales, y en general es aceptada en la comunidad científica la siguiente definición (31):

> “Los procedimientos Metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos”

De esta manera, los procedimientos Metaheurísticos se sitúan conceptualmente por encima de los heurísticos en el sentido que guían el diseño de éstos. Por ello, al enfrentarnos a un problema de optimización, podemos escoger cualquiera de estos métodos para diseñar un algoritmo específico que lo resuelva aproximadamente.

Actualmente existe un gran desarrollo y crecimiento de estos métodos. En esta investigación usamos ciertas metaheurísticas que se han consolidado últimamente en el tratamiento de problemas difíciles, los métodos GRASP, véase (32), junto con otros de reciente desarrollo como el de Búsqueda en Vecindades Variables, planteados en (16), y Rencadenamiento de Trayectorias, que se pueden ver en (15).

3.5.1. Metaheurística GRASP

El método GRASP (acrónimo del inglés "Greedy Randomized Adaptative Search Procedure") es una metaheurística de desarrollo relativamente reciente, introducida a finales de la década del 80 del siglo XX por Feo y Resende, en (32). De manera general se puede describir la operación de GRASP de la siguiente manera:

En una primera fase se aplica un procedimiento para construir una "buena" solución factible, la cual después en una segunda fase es mejorada por una técnica de búsqueda local. Estas dos fases son repetidas de manera iterativa durante cierto número máximo de iteraciones, recordando la mejor solución explorada. En su forma clásica la fase constructiva de GRASP se realiza mediante un procedimiento glotón adaptativo; es decir, un método de construcción entrenado para considerar la aleatoriedad y que se adapta en cada paso de la construcción.

Se dice que un heurístico con base en GRASP se adapta porque en cada iteración se actualizan los beneficios obtenidos al añadir el elemento seleccionado a la solución parcial. Es decir, la evaluación que se tenga de añadir un determinado elemento a la solución en la iteración j, no coincidirá necesariamente con la que se tenga en la iteración $j + 1$.

El heurístico GRASP es aleatorizado porque no selecciona el mejor candidato según la función voraz adaptada sino que, con el objeto de diversificar y no repetir soluciones en dos construcciones diferentes, se construye un lista con los mejores candidatos de entre los que se toma uno al azar.

Al igual que ocurre en muchos métodos, las soluciones generadas por la fase de construcción de GRASP no suelen ser óptimos locales. Dado que la fase inicial no garantiza la optimalidad local respecto a la estructura de entorno en la que se esté trabajando (notar que hay selecciones aleatorias), se aplica un procedimiento de búsqueda local como Post-procesamiento para mejorar la solución obtenida

Experimentalmente, se ha demostrado que la distribución de la muestra generalmente tiene un valor en promedio que es inferior al obtenido por un procedimiento determinista, sin embargo, la mejor de las soluciones encontradas generalmente supera a la del procedimiento determinista con una alta probabilidad, como se demuestra en (15).

Las implementaciones GRASP generalmente son robustas en el sentido de que es difícil el encontrar ejemplos patológicos en donde el método funcione arbitrariamente mal. El enorme éxito de este método se puede constatar en la gran cantidad de aplicaciones que han aparecido en los últimos años. En (33) se comentan cerca de 200 trabajos en los que se aplica GRASP a la resolución de problemas concretos.

De acuerdo a la propuesta original de Feo y Resende (34), un método GRASP general responde al algoritmo descrito en la Figura 14.

$dim =$ dimensión del problema
Para $k = 1, \dots, \max\_iter$ hacer:
    Fase de Construcción:
        $solución = \emptyset$
        Evaluar el aporte o el costo de cada candidato individual
        Si $dimensión(solución) \neq dim$:
        Construir la LCR: Lista de Candidatos Restringida
        Seleccionar aleatoriamente un elemento e de LCR
        $solución = solución \cup \{e\}$
    Fin de la fase de Construcción
    Fase de búsqueda local:
        Si $solución$ no es optimal localmente:
        Hallar $s' \in$ la vecindad de $(solución)$
        De tal forma que se obtenga una mejora de la función objetivo
        $solución = s'$
    Fin de la fase de búsqueda local
    $valor = Mejor\ valor\ \{solución, valor\}$
Fin $(***For***)$
Devuelva $valor$

Figura 14. Esquema de un algoritmo GRASP general

Durante la fase de construcción se desarrolla la lista de candidatos restringida, LCR, de la siguiente manera:

1. Se determinan todos los elementos factibles de ingresar a la solución actual
2. Se determina una función que describa el aporte que tendría en la solución la inclusión de cada uno de los elementos factibles, esto da la dimensión glotona al método (*Greedy*).
3. Se escoge los $\alpha\%$ mejores elementos factibles, esta es la lista LCR
4. Aleatoriamente se escoge un elemento de LCR para añadirlo a la solución actual, esto da la dimensión probabilística (*Randomized*) del método.

Hay que resaltar el hecho de que en cada paso de la fase de construcción, un nuevo elemento es incorporado a la solución actual, y la lista de candidatos restringida LCR debe ser actualizada con base en el aporte de los nuevos elementos factibles a la solución actual, lo cual da el carácter adaptativo al método (*Adaptative*).

La eficiencia del método depende grandemente del valor de α que se use, por lo que este parámetro debe ser bien calibrado en los experimentos numéricos. Por definición α debe tomar valores en el intervalo $[0,1]$. Con $\alpha = 1$ el método se vuelve de búsqueda totalmente aleatoria, pues cualquier elemento tendría probabilidad de incorporarse a la solución en cada paso del algoritmo, mientras que con $\alpha = 0$ el método se aproxima a un procedimiento totalmente glotón, pues solo el mejor elemento sería incorporado en cada paso a la solución. Una modificación importante que ha dado buenos resultados es cambiar dinámicamente la dimensión de la lista de candidatos restringida por medio de alguna estrategia, es decir aumentar o disminuir el valor de α de acuerdo a la conveniencia de diversificar o intensificar la búsqueda, respectivamente, en ese caso se habla de un algoritmo GRASP reactivo.

Otra modificación importante que se reporta últimamente en la literatura, como en (35), a la metaheurística GRASP es la siguiente: las construcciones originales basadas en GRASP primero evalúan cada elemento candidato a seleccionarse por la función voraz (*Greedy*) que mide su aporte a la solución que está construyéndose, y de ahí se construye la Lista de Candidatos Restringida RCL, de la cual se selecciona finalmente un elemento aleatoriamente (*Randomized*) para ser incorporado a la solución. Ese es el orden de las acciones a ejecutarse en un algoritmo clásico GRASP, sin embargo, estudios más recientes como el reportado en (35) han demostrado

que un diseño alternativo en el cual primero aplicamos la aleatorización (*Randomized*) y luego la acción voraz (*Greedy*) puede ser más eficiente para obtener mejores resultados que si se aplicara primero la función voraz y luego la aleatorización como en un algoritmo GRASP clásico. La heurística que se propone en esta investigación, denominada GRASP3, y que se desarrolla en la sección 4.3, está construida con base a esta nueva filosofía.

Últimamente se ha propuesto el uso del método GRASP en combinación con la metodología de Rencadenamiento de Trayectorias (*Path Relinking* en su terminología en inglés), con lo cual se han desarrollado algoritmos muy eficientes para resolver una amplia gama de problemas de optimización combinatoria, tal como se relata en (36). Usamos esta hibridación de la metodología GRASP con Rencadenamiento de trayectorias como una fase de post optimización en el algoritmo propuesto.

3.5.2. Rencadenamiento de Trayectorias

El término Rencadenamiento de Trayectorias se refiere a una técnica de intensificación de la búsqueda en regiones atractivas del espacio de soluciones, que fue originalmente propuesta por Glover en el contexto de la metaheurística de búsqueda TABU, como se puede leer en (17). La técnica consiste en utilizar una población de buenas soluciones que han sido creadas previamente con algún método, por ejemplo con GRASP, para luego intentar crear mejores soluciones. El objetivo del Rencadenamiento de trayectorias no es construir una solución a partir de otras, sino más bien crear un conjunto de soluciones, vecinas unas de otras, que conecten dos soluciones de buena calidad, creando así un camino o una trayectoria entre estas dos soluciones. Usualmente cada una de las soluciones del camino formado se mejora con un procedimiento de búsqueda local.

Más específicamente, dada una solución denominada solución inicial, x_s, y una solución denominada solución guía, x_t, se denomina un camino de x_s a x_t a la sucesión de soluciones: $x_s = x(0),\ x(1),\ x(2), \dots, x(r-1), x(r) = x_t$, donde $x(i+1)$ es obtenida al introducir en $x(i)$ un atributo que reduce la distancia de $x(i)$ hasta la solución guía x_t. Es decir, en el camino originado, cada nuevo elemento que se genera en la sucesión se parece más a la solución guía.

Algunas estrategias que se pueden considerar para el desarrollo de la técnica son, según (36):

- Forward: La peor entre las soluciones x_s y x_t se pone como el origen y la otra es la solución guía;
- Backward: La mejor entre las soluciones x_s y x_t se pone como el origen y la otra es la solución guía;
- Backward y Forward: Se exploran las dos trayectorias diferentes, de x_s a x_t y de x_t a x_s, es decir intercambiando sus roles;

En algunas implementaciones se ha considerado explorar el entorno de las soluciones intermedias para dar más posibilidad al descubrimiento de buenas soluciones. Detalles sobre el método pueden encontrarse en (17).

En (28) se propone el uso de Rencadenamiento de trayectorias en el contexto de GRASP, aunque aquí su significado es diferente ya que las soluciones no han estado unidas por ningún camino previo. Así, una vez generada una colección de soluciones mediante las fases de construcción y mejora de GRASP, se seleccionan parejas (o subconjuntos) de soluciones para unirlas mediante este procedimiento. A partir de una solución se realiza una búsqueda local para llegar a la otra (o a una combinación de las otras en el caso de subconjuntos).

La hibridación de GRASP y Rencadenamiento de trayectorias se muestra muy prometedora, por lo que ha despertado gran interés entre la comunidad de investigación de operaciones y de ciencias de la computación, una muestra de ello es que algunos autores la han usado para tratar con éxito los problemas clásicos de la diversidad máxima, como se puede ver en (28).

En la heurística GRASP3 que se propone en esta investigación, y que se muestra en la sección 4.3, usamos la estrategia Backward y Forward para generar las trayectorias entre cada pareja del conjunto de soluciones élite.

3.5.3. Búsqueda en Vecindades Variables

La Búsqueda en Vecindades Variables (*Variable Neighborhood Search*, VNS por sus siglas) es una metaheurística de desarrollo reciente, propuesta por Hansen en (16) para resolver problemas de optimización combinatoria, cuya idea básica es cambiar sistemáticamente la vecindad, por medio de un procedimiento aleatorio o determinístico, dentro de un procedimiento

de búsqueda local. Este cambio de vecindad puede favorecer enormemente la eficiencia de los algoritmos pues diversifica la búsqueda y así la búsqueda no queda viciada por algún sesgo en el tipo de vecindad seleccionada, por lo que VNS constituye una herramienta simple y poderosa para mejorar la eficiencia de los algoritmos de búsqueda.

La idea original fue considerar distintas estructuras de entornos y cambiarlas sistemáticamente para escapar de los mínimos locales. El VNS básico obtiene una solución del entorno de la solución actual, ejecuta una búsqueda monótona local desde ella hasta alcanzar un óptimo local, que remplaza a la solución actual si ha habido una mejora y modifica la estructura de entorno en caso contrario. Una variante de esta estrategia básica es la búsqueda descendente por entornos variables (VND), que aplica una búsqueda monótona por entornos cambiando de forma sistemática la estructura de entornos cada vez que se alcanza un mínimo local.

La ventaja de utilizar varias estructuras de entornos, como se plantea en VNS, radica en el hecho de que un óptimo local para un determinado entorno, no tiene por qué serlo para otro, por lo que la búsqueda podrá continuar hasta obtener una solución que sea un buen óptimo local para el problema. VNS ha sido hibridizado con éxito con otras metaheurísticas como Búsqueda Tabú o Scatter Search, y en particular con GRASP en su fase de búsqueda local, como en (33). En nuestra investigación utilizamos VNS en la segunda fase del algoritmo GRASP3.

La filosofía del procedimiento VNS es la siguiente: se empieza representando por $N_k,\ k = 1, 2, \dots, k_{max}$ al conjunto de estructuras de vecindades preseleccionadas, de tal manera que $N_k(x)$ representa el conjunto de soluciones, vecinas de x, en la $k - ésima$ vecindad de x. Estas vecindades N_k pueden ser inducidas a partir de una o más métricas introducidas en el espacio de soluciones considerado. Entonces la VNS se basa en tres principios generales:

1. Si x^* es un óptimo local con respecto a una estructura de vecindad, no tiene necesariamente que serlo para otra estructura de vecindad diferente.
2. Un óptimo global es un óptimo local respecto a todas las estructuras de vecindad.
3. En muchos problemas combinatorios, los óptimos locales respecto a una o varias estructuras de vecindad están relativamente cerca unos de otros.

Con VNS se deben tener en cuenta estos tres principios a la hora de diseñar un algoritmo, pudiendo efectuar los cambios de vecindad de manera determinista o estocástica.

En el Capítulo 4 se utilizan estas metaheurísticas: GRASP, VNS y PR, para diseñar algoritmos híbridos que permitan resolver de manera eficiente el problema de la diversidad máxima en su variante Max-Mean

Capítulo 4.

Diseño de heurísticas eficientes para resolver el problema Max-Mean.

Dado que los problemas considerados en el Capítulo 2 son de tipo NP-duro, y que en particular el problema Max-Mean en su formulación (2.27)-(2.32) no puede ser resuelto en tiempos razonables para problemas con $n > 25$ nodos, según se demuestra en la sección 4.1.2, es claro que se requiere diseñar heurísticas para resolver instancias de tamaños medianos o grandes.

Aunque en la literatura no han sido propuestos métodos de solución heurísticos o exactos para el problema Max-Mean, si se han desarrollado métodos de solución eficientes para los problemas clásicos de la diversidad máxima, en particular para el problema Max-Sum y el problema Max-Min, que han sido intensamente estudiados, como en (11), (30) y (9). Estos procedimientos pueden ser adaptados gracias a la relación que existe entre estas variantes del problema de la diversidad máxima y el problema Max-Mean, relación que se determina en este capítulo.

En (30) se establece que la mayoría de algoritmos desarrollados para resolver el problema Max-Sum son demasiado sofisticados y no explotan la estructura simple que muchas veces tienen los problemas combinatorios de este tipo, y se plantean algoritmos simples que sorprendentemente superan a otros sofisticados algoritmos propuestos en la literatura, demostrando que a veces los algoritmos más simples pueden ser mejores. Considerando esta filosofía, desarrollamos un algoritmo basado en la metaheurística GRASP que explota las características del problema Max-Mean, al cual hibridizamos con otras técnicas modernas de intensificación, como el Rencadenamiento de trayectorias (Path Relinking, PR), y la búsqueda en vecindades variables (Variable Neighborhood Search, VNS). Luego de los experimentos numéricos este algoritmo resultó ser más eficiente en la solución de problemas medianos y grandes.

Para comparar la eficiencia de nuestra heurística, en las secciones 4.2.1 y 4.2.2 seleccionamos algunos de los algoritmos más eficientes reportados en la literatura, que han sido

desarrollados para resolver otros problemas de la diversidad máxima, y los adaptamos para resolver el problema Max-Mean.

Para los experimentos computacionales generamos dos tipos de problemas de prueba que son adecuados para el análisis del desempeño de los algoritmos desarrollados para resolver el problema Max-Mean.

4.1. Relación entre el problema Max-Sum y el problema Max-Mean

Se puede obtener la solución óptima del problema Max-Mean de una manera indirecta si resolvemos el modelo Max-Sum para todos los valores posibles de m; es decir, para $m = 2, 3, \dots, n$, y luego dividimos la solución resultante para el correspondiente valor de m. Entonces, el mejor valor de estos $(n-1)$ valores obtenidos es el óptimo del modelo Max-Mean. En esta sección de muestran propiedades interesantes del problema Max-Mean obtenidas por experimentos computacionales, con base en las cuales se desarrollarán heurísticas eficientes para su solución.

Para el diseño de los experimentos consideramos la generación de problemas de prueba (instancias) con características adecuadas al problema Max-Mean, así, de acuerdo con la Propiedad 6, una de esas características deseadas es que las instancias contengan valores de d_{ij} positivos y negativos, otra característica deseada es tener al menos dos tipos diferentes de instancias con el fin de observar el rendimiento de las heurísticas en diferentes circunstancias.

4.1.1. Problemas de prueba

En la literatura se reportan conjuntos de problemas de prueba usados para experimentos computacionales del problema de la diversidad máxima en sus variantes clásicas Max-Min y Max-Sum, en particular en el sitio Web del proyecto OPTSICOM, establecido en (37), se puede descargar la colección MMDPLIB que contiene instancias de dos tipos: Geométricas, compuestas de matrices con n =10, 15, 30, 100, 250 y 500, cuyos valores fueron calculados como distancias euclidianas desde puntos generados aleatoriamente en un plano; y, Random, compuestas de matrices con números enteros aleatorios.

Como para el problema Max-Mean no tiene sentido considerar ejemplos de prueba cuyas d_{ij} sean no negativas o que satisfagan las propiedades de una métrica, en el análisis de los

métodos que se desarrollan en esta investigación se consideran conjuntos de problemas de prueba correspondientes a dos nuevos tipos de instancias donde, como se explicó antes, las distancias toman valores sin restricción de signo. El objetivo de generar dos tipos diferentes de instancias radica en comprobar la eficiencia de los algoritmos en diferentes estructuras de datos.

Estas dos clases de instancias las denominamos: de Tipo I y de Tipo II, y se generaron como sigue:

TIPO I: Este conjunto de datos consiste de 60 matrices simétricas con números aleatorios en el intervalo $[-1,1]$ y generados a partir de una distribución uniforme. Estos números podrían representar, por ejemplo los grados de afinidad o de no afinidad entre individuos en una red social. En aplicaciones prácticas este tipo de estructuras de datos se originarían al estimar los valores de los coeficientes d_{ij} por medio de ecuaciones como las descritas en la sección 1.2.

Se generaron 10 problemas de prueba de este tipo para cada uno de los tamaños: $n = 20,\ 25, 30, 35, 150\ y\ 500$, con el fin de ejemplificar el funcionamiento de los métodos de solución que se proponen en problemas de tamaños pequeño, mediano y grande.

TIPO II: Este conjunto de datos consiste también de 60 matrices simétricas, pero cuyos coeficientes se generaron como números aleatorios con distribución uniforme en el conjunto $[-1,-0.5] \cup [0.5,1]$. Siguiendo con el ejemplo anterior, estos coeficientes podrían representar grados de afinidad entre individuos en un grupo social, pero en el que existe un ambiente de polarización, y hay mucha afinidad o bien mucha no afinidad entre los individuos, existiendo poca indiferencia en sus relaciones, ya que no se generan valores en el intervalo $[-0.5, 0.5]$.

Igual que antes, se generaron 10 problemas de prueba de este tipo para cada valor de $n = 20,\ 25, 30, 35, 150\ y\ 500$, para ejemplificar el rendimiento de los métodos de solución propuestos en problemas de tamaño pequeño, mediano y grande.

También es importante explicar la forma en que estos valores fueron distribuidos en las matrices de distancia, y puesto que se busca garantizar la aleatoriedad, la ubicación de los números generados también fue aleatoria. En la Figura 15 se observa la estructura con la que se generaron los ejemplos de prueba del conjunto de instancias de tipo I, y en la Figura 16 se observa el esquema mediante el cual se generaron los ejemplos de prueba del conjunto de instancias de tipo II.

$$d = \begin{pmatrix} 0 & d_{12} & d_{13} & d_{14} & \cdots & d_{1n} \\ & 0 & d_{23} & d_{24} & \cdots & d_{2n} \\ & & 0 & d_{34} & \cdots & d_{3n} \\ & & & 0 & \ddots & \vdots \\ & & & & 0 & d_{n-1,n} \\ & & & & & 0 \end{pmatrix}$$

$\left[\left[\frac{n(n-1)}{2}\right]\right]$ elementos d_{ij} generados aleatoriamente en $[-1,1]$ y ubicados al azar entre $\frac{n(n-1)}{2}$ posiciones posibles

Figura 15. Estructura de los ejemplos de prueba tipo I

$$d = \begin{pmatrix} 0 & d_{12} & d_{13} & d_{14} & \cdots & d_{1n} \\ & 0 & d_{23} & d_{24} & \cdots & d_{2n} \\ & & 0 & d_{34} & \cdots & d_{3n} \\ & & & 0 & \ddots & \vdots \\ & & & & 0 & d_{n-1,n} \\ & & & & & 0 \end{pmatrix}$$

$\left[\left[\frac{n(n-1)}{4}\right]\right]$ elementos d_{ij} generados aleatoriamente en $[-0.5,1]$ y ubicados aleatoriamente entre las $\frac{n(n-1)}{2}$ posiciones posibles, los $n - \frac{n(n-1)}{2}$ elementos restantes generados aleatoriamente en $[0.5,1]$

Figura 16. Estructura de los ejemplos de prueba tipo II

Con estas instancias de pruebas así creadas buscamos simular características posibles en un problema en el que los valores de d_{ij} son positivos y negativos, por ejemplo en el caso de un grupo social con afinidades y no afinidades entre individuos, y también tener conjuntos de datos diferentes para registrar el desempeño de los algoritmos diseñados cuando éstos se ejecutan en instancias de uno u otro tipo.

Estas instancias de prueba han sido publicadas en el sitio web del proyecto OPTSICOM y están disponibles en (38) para otros investigadores que desean usar los datos.

4.1.2. Características del problema Max-Mean resuelto a través del problema Max-Sum

Es evidente que se puede obtener la solución óptima del problema Max-Mean de una manera indirecta si resolvemos el modelo Max-Sum para todos los valores posibles de m; es decir, para $m = 2, 3, \dots, n$, y luego dividimos las soluciones resultantes para el correspondiente valor de m. Entonces, el mejor valor de estos $(n-1)$ valores obtenidos es el óptimo del modelo Max-Mean. Es decir, si $Z^*_{Max-Sum(m)}$ es el valor óptimo de la función objetivo del problema Max-Sum con m elementos a seleccionar, y $Z^*_{Max-Mean}$ es el valor óptimo de la función objetivo del problema Max-Mean, entonces:

$$Z^*_{Max-Mean} = \max_{m\in\{2,\dots,n\}} \left\{ \frac{Z^*_{Max-Sum(m)}}{m} \right\}$$

En la Figura 17 se muestra el resultado de resolver el problema Max-Mean de esta manera indirecta, para un ejemplo de prueba de tipo I de tamaño $n = 50$, resolviendo de manera exacta 49 problemas de tipo Max-Sum, cada uno de ellos en su formulación lineal (2.2)-(2.4) por medio del optimizador Cplex. En el eje y consta el valor óptimo de la función objetivo del problema Max-Sum dividido para el respectivo valor de m, donde m toma todos sus valores posibles: 2, 3, …, 50.

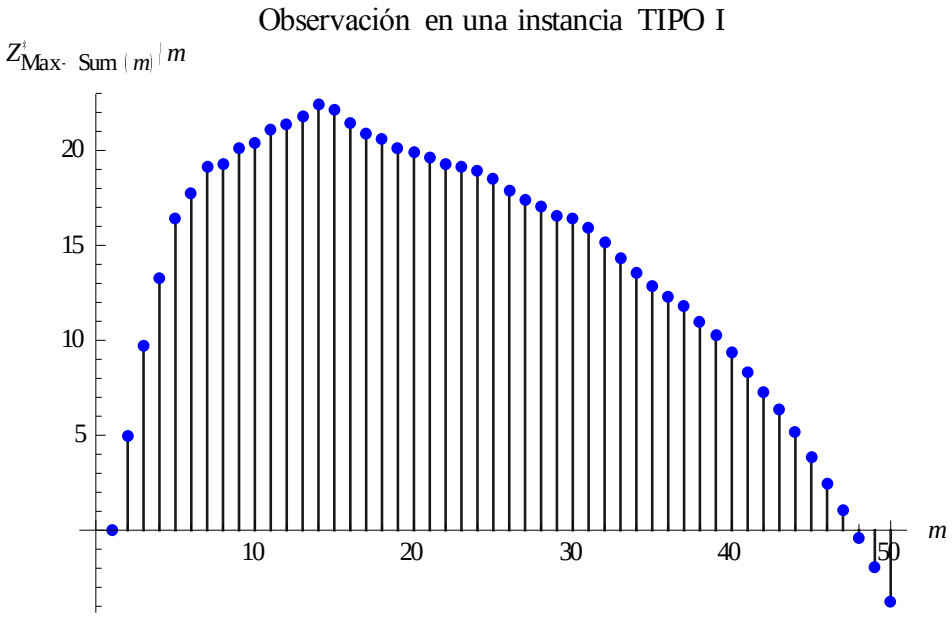


Figura 17. Solución del problema Max-Mean a través de las soluciones del modelo Max-Sum

En la Figura 17 se puede observar que el valor óptimo de cada problema Max-Sum, dividido para m, se incrementa monótonamente cuando m aumenta de 2 a 14, y luego este valor decrece también de manera monótona para el resto de valores de m (desde $m = 15$ a $m = 50$). Así el óptimo del problema Max-Mean se alcanza en este ejemplo para m=14.

Este mismo patrón, de la evolución de los valores del óptimo del problema Max-Sum divididos para m, en forma de una función cuasi-cóncava, pudo ser observado en otros experimentos numéricos que se desarrollaron para diferentes valores de n, para otros problemas de prueba de tipo I, tal como se observa en la Figura 18. En el Capítulo 3 consideramos este patrón como una característica importante a incluir en el diseño de un algoritmo GRASP eficiente para resolver el problema Max-Mean.

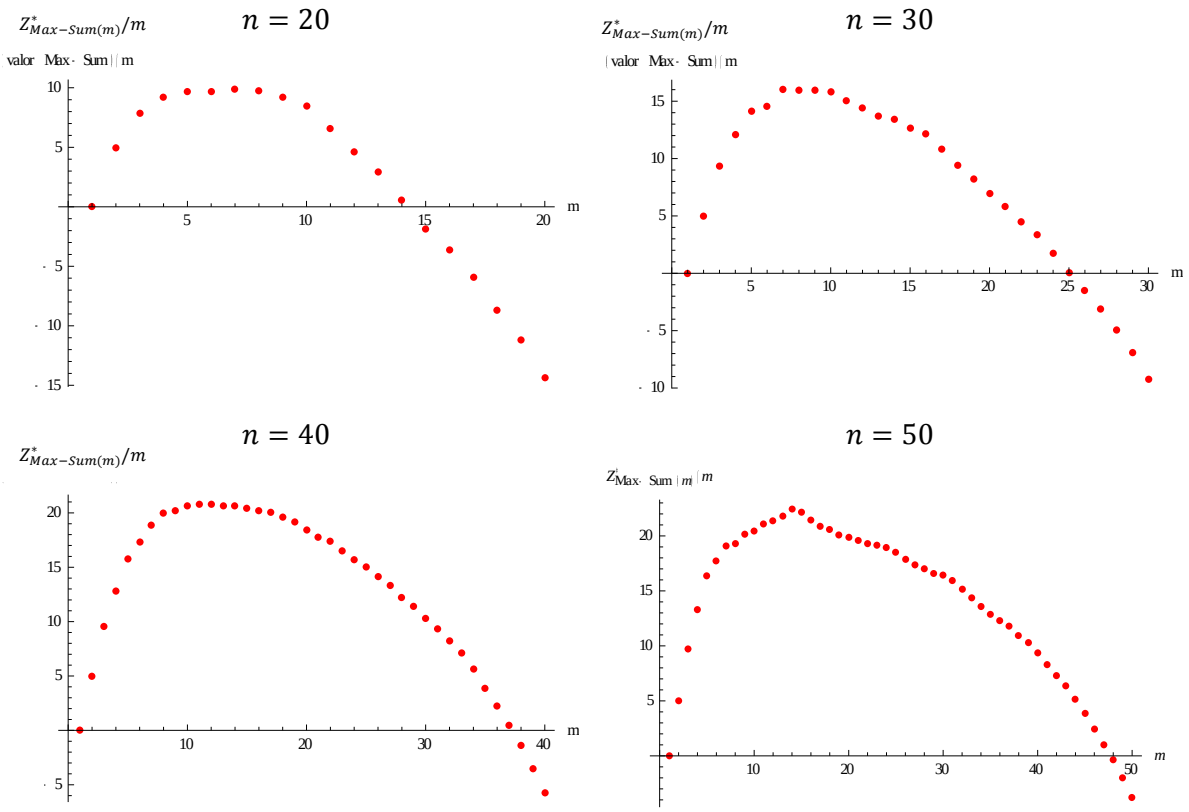


Figura 18: Forma cuasi-cóncava de los valores óptimos del problema Max-Sum divididos para *m*

Esta característica observada además de proporcionarnos información muy útil para la exploración del espacio de soluciones en el diseño de una heurística eficiente para resolver el problema, también nos indica otro camino para obtener la solución del problema Max-Mean, esto es resolver los $(n-1)$ problemas Max-Sum, a través de su formulación lineal (2.2)-(2.4), para $m = 2, 3, \dots, n$, y luego dividimos la solución resultante para el correspondiente valor de m. Entonces, el mejor valor de estos $(n-1)$ valores obtenidos es el óptimo del modelo Max-Mean. Parecería obvio que esta estrategia requeriría más tiempo que resolver directamente un solo problema lineal dado por la formulación (2.27)-(2.32).

Haciendo un análisis rápido de la talla de estos problemas, podemos darnos cuenta que el número de variables y restricciones en la formulación de cada uno de los problemas Max-Sum es del mismo orden que en la formulación del problema Max-Mean, como se observa en la Tabla 10, y en la Tabla 11, en las que se presentan los resultados para el número de variables y restricciones, se puede observar que no hay grandes diferencias en cuanto a la talla de estos dos problemas, nótese que en el caso del Max-Sum, estos números se refieren a las tallas de cada uno de los problemas Max-Sum a resolverse, y como tienen que resolverse $(n-1)$ problemas de ellos podría pensarse que esta estrategia requiere más tiempo, curiosamente la situación es totalmente diferente, como se observa en el siguiente experimento.

En el próximo experimento computacional se resuelve de manera exacta el problema Max-Mean utilizando las dos estrategias, más específicamente se comparan las soluciones óptimas exactas obtenidas con Cplex cuando se resuelve directamente la formulación lineal del problema Max-Mean (2.27)-(2.32) y cuando se lo resuelve indirectamente a través de procesar $(n-1)$ veces la formulación del problema Max-Sum (2.2)-(2.4) para todos los posibles valores de m entre 2 y n.

Tabla 10. Talla de las formulaciones lineales de los problemas Max-Sum y Max-Mean

	PROBLEMA	
	Max-Sum	Max-Mean
Número de variables	$\frac{n^2}{2}+\frac{1}{2}n$	$\frac{n^2}{2}+\frac{3}{2}n+1$
Número de restricciones	$\frac{3}{2}n^2-\frac{3}{2}n+1$	$2n^2+n+2$

A continuación se describe el experimento numérico desarrollado: Para esta comparación usamos las 60 instancias de tamaño pequeño con $n = 20, 25\ y\ 30$, tanto de TIPO I como de TIPO II definidas en la sección 4.1.1.

La Tabla 12 muestra, para cada método y cada tamaño del problema, el valor promedio de la función objetivo $(Valor)$ en la solución óptima, el promedio del número de elementos que terminan siendo seleccionados en la solución óptima (m), y el tiempo CPU promedio en segundos (CPU), ND significa que el valor no está disponible porque no se obtuvo la solución en un tiempo de 5 horas.

Tabla 11. Talla de los problemas Max-Sum y Max-Mean para diferentes valores de n

	Número de variables		Número de restricciones	
n	Max-Sum	Max-Mean	Max-Sum	Max-Mean
15	120	136	316	467
16	136	153	361	530
17	153	171	409	597
18	171	190	460	668
19	190	210	514	743
20	210	231	571	822
21	231	253	631	905
22	253	276	694	992
23	276	300	760	1083
24	300	325	829	1178
25	325	351	901	1277
26	351	378	976	1380
27	378	406	1054	1487
28	406	435	1135	1598
29	435	465	1219	1713
30	465	496	1306	1832

Tabla 12. Soluciones del problema Max-Mean con Cplex12

		TIPO I		TIPO II	
n		**Max -Mean**	**Max - Sum (*n*-1) veces**	**Max -Mean**	**Max - Sum (*n*-1) veces**
20	***CPU* (s)**	50.334	14.662	66.714	19.164
	Valor	1.443	1.443	1.898	1.898
	$\boldsymbol{m}$	7.400	7.400	7.500	7.500
25	***CPU* (s)**	694.606	41.826	1995.100	59.581
	Valor	1.732	1.732	2.207	2.207
	$\boldsymbol{m}$	9.800	9.800	9.600	9.600
30	***CPU* (s)**	> 5 horas	102.303	> 5 horas	182.176
	Valor	ND	1.875	ND	2.383
	$\boldsymbol{m}$	ND	10.700	ND	10.800

Los resultados que se muestran en la Tabla 12 claramente indican que Cplex sólo permite resolver problemas pequeños en tiempos moderados. En particular con la formulación lineal (2.27)-(2.32) solo se pueden resolver instancias de tamaño $n < 30$ para los problemas de prueba de tipo I y de tipo II, y para $n = 30$ no se pudo obtener el resultado en 5 horas de procesamiento.

Sorprendentemente, el modelo Max-Sum aplicado $(n-1)$ veces permite resolver instancias de tamaños más grandes en menor tiempo, y se pudo obtener la solución para $n = 30$ en 102.30 segundos en promedio, y para $n = 35$ en 719.51 segundos en los problemas de tipo I, en los problemas de tipo II esto requirió de más tiempo. Se puede concluir que si se quiere resolver el problema Max-Mean de manera exacta es preferible usar la estrategia de resolver $(n-1)$ veces el modelo Max-Sum ya que consistentemente dio mucho menor tiempo en todos los experimentos. Esto se debe probablemente al hecho de que la relajación continua del problema Max-Sum provee mejores cotas que las que provee la relajación continua del problema Max-Mean.

En lo que sigue se realiza una prueba estadística mediante la cual se comprueba que en general los problemas de tipo II son más difíciles de resolver que los de tipo I, ya que el tiempo que requirió su procesamiento en los dos procedimientos fue significativamente mayor en los problemas de tipo I. Este comportamiento sistemático se observa resumido en la Tabla 13, los tiempos de procesamiento que requirió cada uno de los problemas de tipo I y tipo II, cuando se

aplica el procedimiento de resolver los $(n-1)$ problemas Max-Sum se muestran en la Tabla A 1 del ANEXO 1.

Se realizó una prueba estadística, basada en el test t-student para muestras apareadas, para comprobar la hipótesis que el tiempo promedio de procesamiento en problemas de tipo I es menor que el tiempo de procesamiento promedio en problemas tipo II, aplicando la prueba de hipótesis a las 30 parejas de tiempos de procesamiento se obtuvieron los resultados resumidos en la Tabla 13.

De igual manera se realizó la prueba estadística para los tiempos de ejecución al resolver directamente la formulación lineal del problema Max-Mean, los tiempos detallados de procesamiento para cada uno de los problemas se presentan en la Tabla A 2 del ANEXO 1, mientras que los resultados de la aplicación de la prueba se resumen en la Tabla 14.

El nivel de significancia en todos los casos fue inferior al 5%, por tanto concluimos que las instancias de tipo II son más difíciles que las de tipo I, y permitirán discriminar de mejor manera entre los algoritmos heurísticos que se diseñen.

Tabla 13. Tiempos de procesamiento de problemas Tipo I Vs. Tiempos de problemas Tipo II en el método de resolver $(n-1)$ problemas Max-Sum

Prueba de hipótesis para medias con muestras apareadas			
	cpu tipo I - cpu tipo II		
	n = 20	**n = 25**	**n = 30**
Tamaño muestral	10	10	10
Media Muestral	-4.7302	-17.7551	-79.8735
Desviación estándar muestral	5.360580396	12.8170327	79.66914337
Hipótesis Nula sobre la diferencia	0	0	0
Hipótesis alternativa sobre la diferencia	<> 0	<> 0	<> 0
Error estándar de la media	1.695164363	4.05310162	25.19359523
Estadístico t-Test	-2.7904	-4.3806	-3.1704
Valor p	0.0210	0.0018	0.0114
Hipótesis nula al 5% de significancia	Rechazar	Rechazar	Rechazar

Tabla 14. Tiempos de procesamiento de problemas Tipo I Vs. Tiempos de problemas Tipo II en el método de resolver directamente la formulación del problema Max-Mean

Prueba de hipótesis para medias con muestras apareadas		
	cpu tipo I - cpu tipo II	
	n = 20	**n = 25**
Tamaño muestral	10	10
Media Muestral	-17.3801	-1300.4939
Desviación estándar muestral	28.0405773	249.849174
Hipótesis Nula sobre la diferencia	0	0
Hipótesis alternativa sobre la diferencia	<> 0	<> 0
Error estándar de la media	8.86720913	79.009246
Estadístico t-Test	-1.9900	-16.4600
Valor p	0.0482	< 0.0001
Hipótesis nula al 5% de significancia	Rechazar	Rechazar

En la Figura 19 y en la Figura 20 se observa la evolución de los valores óptimos de los problemas Max-Sum que se generan con cada valor de $m = 2, 3, \ldots, 30$ para los 10 problemas de prueba de tipo I de tamaño $n = 30$, se puede observar el mismo patrón aproximadamente cuasi cóncavo, en todos los problemas, propiedad que explotamos en la siguiente sección para diseñar una heurística eficiente para resolver este problema.

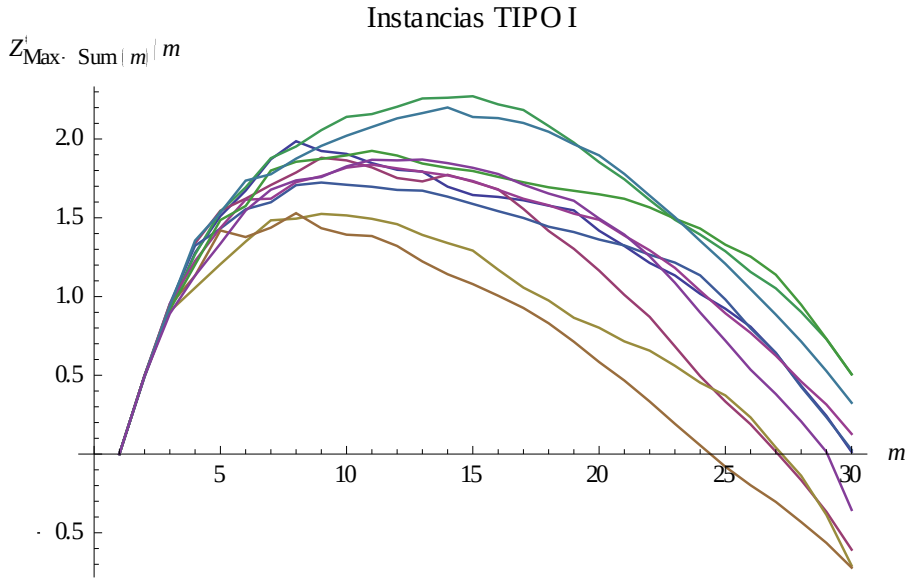


Figura 19. Evolución de los valores óptimos del problema Max-Sum divididos para su correspondiente valor de *m* para los 10 ejemplos de prueba de tipo I

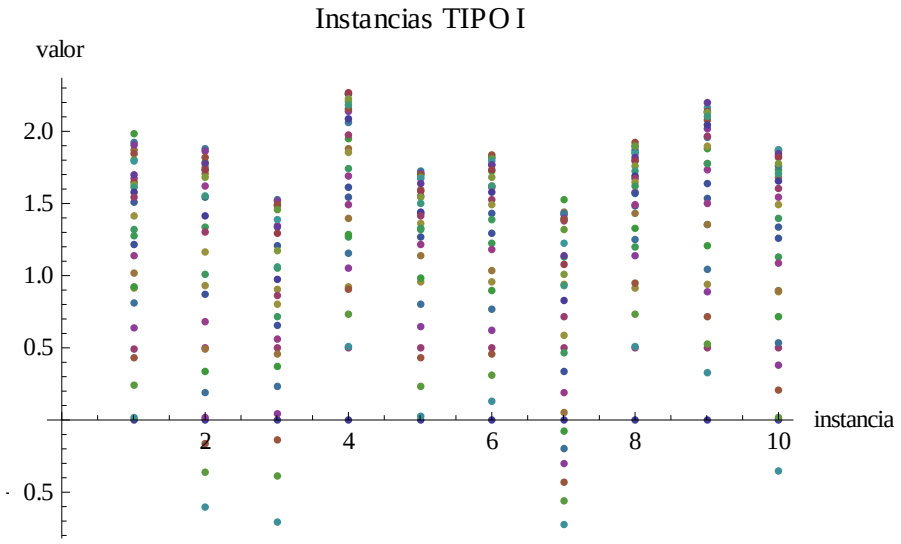


Figura 20. Valores óptimos de todos los problemas Max-Sum divididos por *m* para cada valor de *m* para cada uno de los 10 ejemplos de prueba de tipo I

En la Figura 21 y Figura 22 se observa la misma situación para los problemas de prueba de tipo II.

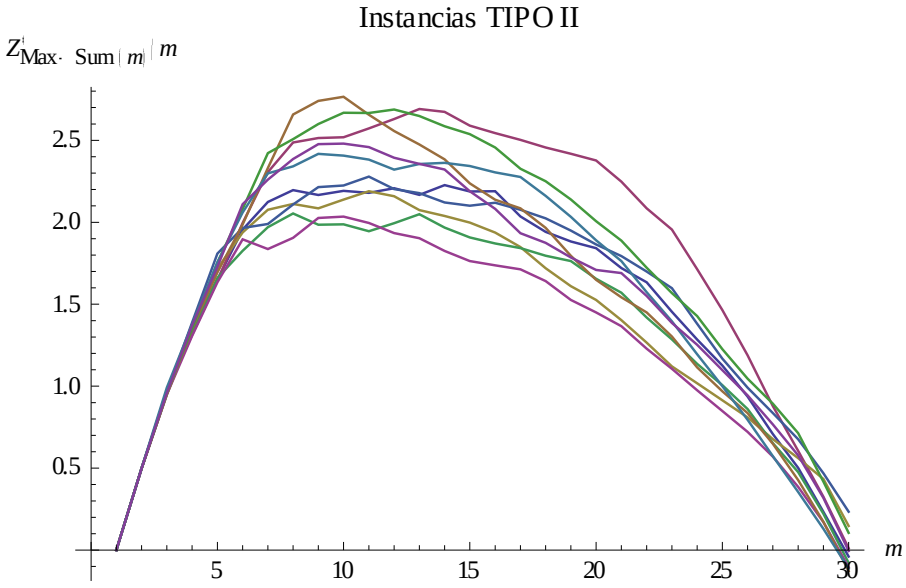


Figura 21. Evolución de los valores óptimos del problema Max-Sum divididos para su correspondiente valor de *m* para los 10 ejemplos de prueba de tipo II

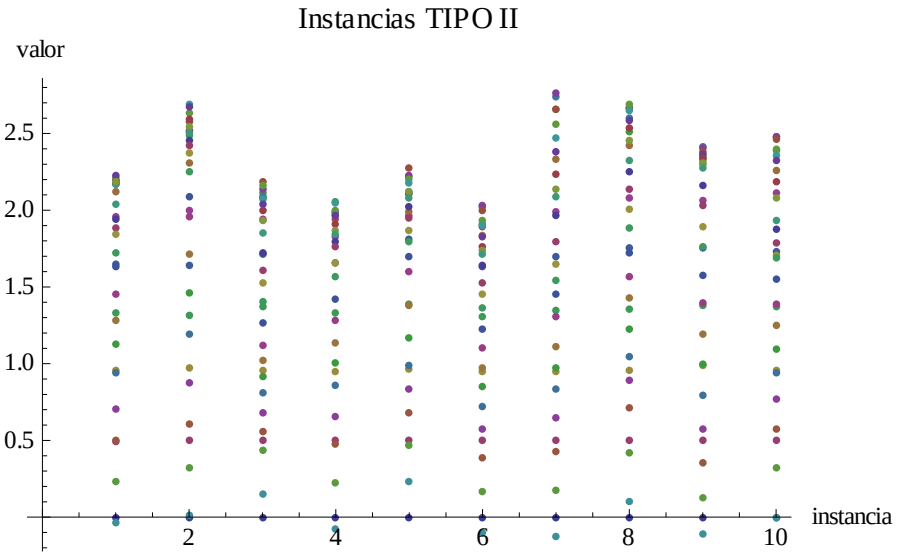


Figura 22. Valores óptimos de todos los problemas Max-Sum divididos por *m* para cada valor de *m* para cada uno de los 10 ejemplos de prueba de tipo II

4.2. Diseño de Heurísticas previas adaptadas al problema Max-Mean

Dado que los problemas considerados en el capítulo 3 son de tipo NP-duro, y que en particular el problema Max-Mean en su formulación (2.27)-(2.32) no puede ser resuelto en tiempos razonables para problemas con $n > 25$ nodos, es claro que se requiere diseñar heurísticas para resolver problemas de tamaños medianos o grandes. Aunque en la literatura no han sido propuestos métodos de solución heurísticos o exactos para el problema Max-Mean, sí se han desarrollado métodos de solución eficientes para los problemas de la diversidad máxima clásicos, en particular para el problema Max-Sum y el problema Max-Min, que han sido intensamente estudiados. También otros autores han propuestos heurísticas eficientes para resolver otras variantes, como el problema Max-MinSum, véase (18). En esta sección adaptamos estas heurísticas para la solución del problema Max-Mean.

4.2.1. Heurística GRASP 1

En (18) Prokopyev y otros proponen un algoritmo GRASP para el problema Max-MinSum descrito en la sección 2.1.4, en esta sección describimos como este algoritmo puede ser modificado para resolver el problema Max-Mean, a este método adaptado lo denominamos heurística GRASP1.

En su algoritmo Prokopyev considera una solución parcial M_k, con k elementos seleccionados. Cada fase de construcción del algoritmo GRASP inicia seleccionando aleatoriamente un elemento cualquiera, el cual genera el conjunto inicial M_1. Luego, en cada iteración, el método determina una lista de candidatos L con los elementos que pueden ser añadidos a la solución en construcción; es decir, $L = \{1, 2, \dots, n\} \setminus M_k$. Para cada elemento i en L, el método calcula $\Delta f^k(i)$, que es la contribución marginal del elemento i a la solución en construcción si éste es añadido a M_k para obtener M_{k+1}.

Como es obligatorio en las construcciones GRASP, se encuentra una lista restringida de candidatos RCL con los mejores elementos de L. En particular, el método ordena los elemento en L de acuerdo a su contribución marginal y forma RCL con sus α primeros elementos, donde α es un número entero aleatoriamente seleccionado en el intervalo $[1, |L|]$. A continuación se selecciona (de acuerdo a una distribución uniforme) un elemento i^* en RCL y se lo añade a la solución parcial, obteniéndose $M_{k+1} = M_k \cup \{i^*\}$. Cada fase de construcción termina cuando es alcanzado el número preestablecido de elementos a seleccionar, es decir, cuando $|M_k| = m$.

En la Figura 23 se muestra un esquema del seudo código de este algoritmo.

1. Aleatoriamente seleccionar un elemento i^* en $V = \{1, 2, \dots, n\}$
2. Poner $M_1 = \{i^*\}$ y $k = 1$
3. Sea m el número de elementos a seleccionar desde el conjunto V

Mientras $(k < m)$:

4. Calcular $L = \{1, 2, \dots, n\} \setminus M_k$
5. Calcular $\Delta f^k(i) \ \ \forall i \in L$
6. Ordenar de mayor a menor los elementos en L de acuerdo a su valor $\Delta f^k(i)$
7. Seleccionar aleatoriamente $\alpha \in [1, |L|]$
8. Construir RCL con los primeros α elementos de L
9. Seleccionar aleatoriamente un elemento i^* en RCL
10. $M_{k+1} = M_k \cup \{i^*\}$
11. $k = k + 1$

Figura 23. Fase de construcción GRASP

Se adaptó este método, propuesto en el contexto del problema Max-Sum, a la resolución del problema Max-Mean, de la siguiente manera: dada una solución parcial M_k, su valor de diversidad es calculado de acuerdo con la ecuación (1.6):

$$div(M_k) = \frac{\sum_{i<j,i,j\in M_k} d_{ij}}{k}$$

Entonces, el valor de la diversidad promedio del nuevo conjunto $M_{k+1} = M_k \cup \{i^*\}$ puede calcularse recursivamente como sigue:

$$\begin{aligned} div(M_{k+1}) &= \frac{\sum_{i<j,i,j\in M_{k+1}} d_{ij}}{k+1} \\ &= \frac{\sum_{i<j,i,j\in M_k} d_{ij} + \sum_{j\in M_k} d_{i^*j}}{k+1} \\ &= \frac{k\ div(M_k)}{k+1} + \frac{\sum_{j\in M_k} d_{i^*j}}{k+1} \end{aligned}$$

Por lo tanto, consideramos en el algoritmo propuesto la siguiente expresión para el cálculo de los aportes individuales de cada candidato a entrar en la solución:

$$\Delta f^k(i) = div(M_{k+1}) - div(M_k) = \frac{-\ div(M_k)}{k+1} + \frac{\sum_{j\in M_k} d_{ij}}{k+1}$$

Pero, como el algoritmo desarrollado en (18) está diseñado para resolver el problema Max-MinSum, y en este problema el número de elementos a seleccionar, $\boldsymbol{m}$, es fijo, y en cambio en el problema Max-Mean $\boldsymbol{m}$ es también una variable de decisión, se adaptó este algoritmo, seleccionando aleatoriamente un valor para $\boldsymbol{m}$ en cada construcción GRASP; así obtenemos soluciones para todos los valores posibles de $\boldsymbol{m}$ cuando el método se ejecuta para un número relativamente grande de iteraciones del algoritmo.

Se completa la descripción de la fase de construcción de GRASP1 para resolver el problema Max-Mean, remplazando el paso 3 del seudo código de la Figura 23 por la siguiente instrucción:

"Seleccionar aleatoriamente un número entero m en el intervalo $[2, m]$"

De esta forma después de que una solución M ha sido construida, se realiza la fase de mejora, con el fin de encontrar un óptimo local del problema.

Esta fase básicamente consiste en un mecanismo de intercambio en el cual un elemento seleccionado, es decir elemento de M, es reemplazado por un elemento no seleccionado del conjunto $N \setminus M$. El método selecciona aleatoriamente ambos elementos y los intercambia si el valor de la función objetivo mejora con este intercambio; sino sucede esto, el intercambio es descartado. La fase de mejora termina luego de $maxiter$ intercambios consecutivos en los que no se haya detectado una mejora. En nuestros experimentos computacionales se fijó $maxiter$ en 100.

A todo este procedimiento que consiste de la construcción más mejora, lo denominamos heurística GRASP1a. Como se verá más adelante, este método no es muy eficiente en comparación con otros que se desarrollan en la investigación, obviamente el método resulta bastante sencillo y no muy preciso, pero sorprendentemente, como se verá también en la sección 5.5, con su utilización se obtuvieron mejores soluciones que utilizando el paquete computacional comercial EVOLVER, que usa algoritmos genéticos para la búsqueda de la solución.

Una forma alternativa de adaptar el algoritmo de la Figura 23, para resolver el problema Max-Mean, consiste en enumerar todos los posibles valores de m. En otras palabras, en una iteración cualquiera, construimos la primera solución con $m = 2$ nodos, luego, la segunda solución con $m = 3$, y así sucesivamente hasta que en el paso $n - 1$ construimos una solución con todos los elementos seleccionados, de esta manera en esta iteración disponemos de $(n - 1)$ soluciones construidas con base en el algoritmo de la Figura 23, con nuestra estrategia guardamos la mejor de ellas. En la próxima iteración se reinicia el valor de m y se realiza nuevamente la construcción de una solución con $m = 2$, luego con $m = 3$, etc., de la misma manera que la anterior iteración. Se realiza este procedimiento para un número máximo de iteraciones preestablecido. A este procedimiento lo denominamos método GRASP1b. Luego de que se ha construido una solución M se realiza, igual que antes, la fase de mejora, la cual consiste del mecanismo de intercambio señalado en el método GRASP1a.

4.2.2. Heurística GRASP 2

Una vez diseñados los métodos GRASP1a y GRASP1b, como adaptaciones de los métodos propuestos en (18) para resolver el problema Max-MinSum, consideramos la adaptación de otra heurística para la resolución del problema estudiado. En este caso recurrimos a la heurística reportada en la literatura como la más eficiente para resolver el problema Max-Sum, que fue propuesta por Duarte y otros en (11). Estos autores desarrollan un algoritmo basado en la metaheurística GRASP, denominada GRASP_C2, para resolver el problema Max-Sum que, como se establece en la sección 4.1, está íntimamente relacionado con el problema Max-Mean. En esta investigación se modificó tal algoritmo para adaptarlo a la solución del problema Max-Mean, a estos algoritmos los denominamos GRASP2a y GRASP2b.

La heurística se describe de la siguiente manera: dada una solución parcial M_k, el valor de su diversidad en el problema Max-Sum es calculado de acuerdo a la ecuación (1.4):

$$div_{sum}(M_k) = \sum_{i<j, i,j \in M_k} d_{ij}$$

Los autores en (11) introducen la distancia entre un elemento i^* y una solución parcial M_k, representada con $d_s(i^*, M_k)$, para calcular recursivamente el valor de la dispersión del conjunto ampliado $M_{k+1} = M_k \cup \{i^*\}$, definida como:

$$\begin{aligned} div_{sum}(M_{k+1}) &= \sum_{i<j; i,j \in M_{k+1}} d_{ij} \\ &= \sum_{i<j; i,j \in M_k} d_{ij} + \sum_{j \in M_k} d_{i^* j} \\ &= div_{sum}(M_k) + d_s(i^*, M_k) \end{aligned}$$

Basados en estos elementos, proponen un método GRASP constructivo, que denominan GRASP_C2 para resolver el problema Max-Sum. En este algoritmo la lista restringida de candidatos, RCL, es calculada con aquellos elementos $i \in L$ para los cuales $d_s(i, M_k)$ es mayor que un cierto valor de umbral. Más específicamente:

$$RCL = \{i \in L : d_s(i, M_k) \geq div_{sum\_min}(M_k) + \alpha(\overline{div})\} \qquad (3.1)$$

Donde:

$$\overline{div} = div_{sum\_max}(M_k) - div_{sum\_min}(M_k)$$

$$div_{sum\_max}(M_k) = \max_{i \in L} d_s(i, M_k) \text{ ; y,}$$

$$div_{sum\_min}(M_k) = \min_{i \in L} d_s(i, M_k)$$

El valor de α ha sido puesto en 0.5, de acuerdo a lo que proponen los autores, otra estrategia que puede utilizarse es que α tome valores variables de acuerdo a la necesidad de intensificar la búsqueda en ciertas regiones (estableciendo un valor de α pequeño) o de diversificar la búsqueda con vecindades más amplias (poniendo α grande), con lo cual se tendría un algoritmo de tipo GRASP reactivo. Igual que lo que se hizo en la sección 4.2.1, este algoritmo puede ser modificado para adaptarlo a la resolución del problema Max-Mean de la siguiente manera:

1. Seleccionar m aleatoriamente al inicio de cada construcción;
2. Dividir el valor obtenido en la construcción para el valor de m generado.

En la Figura 24 se muestra el seudo código asociado.

1. Seleccionar aleatoriamente un elemento i^* en $V = \{1, 2, \dots, n\}$
2. Poner $M_1 = \{i^*\}$ y $k = 1$
3. Seleccionar m aleatoriamente en el intervalo $[2, m]$

Mientras $(k < m)$:

4. Calcular $L = \{1, 2, \dots, n\} \setminus M_k$
5. Calcular $d_s(i, M_k), \forall i \in L,\ div_{sum\_min}(M_k)$ y $div_{sum\_max}(M_k)$
6. Construir la lista RCL con los elementos que satisfagan la ecuación (3.1)
7. Seleccionar aleatoriamente un elemento i^* en RCL
8. $M_{k+1} = M_k \cup \{i^*\}$
9. $k = k + 1$

Figura 24. Fase de construcción del algoritmo GRASP_C2 adaptado al problema Max-Mean

Una vez realizada la fase de construcción en el algoritmo GRASP_C2 que hemos adaptado a la resolución del problema Max-Mean se realiza la fase de post procesamiento basada en

búsqueda local. Esta fase también realiza intercambios como en los métodos GRASP1a y GRASP1b. Sin embargo, para realizar una búsqueda local similar a la propuesta de (11), en lugar de seleccionar aleatoriamente los dos elementos para intercambiarlos, el método se enfoca en seleccionar los elementos con la menor contribución al valor de la función objetivo en la solución actual y trata de intercambiarlo con elemento no seleccionado. Específicamente, para cada $i \in M$ calculamos:

$$d_s(i, M) = \sum_{j \in M} d_{ij}$$

Y consideramos el elemento $i^* = \min_{i \in M}\{d_s(i, M)\}$. Luego, el método explora el conjunto de elementos no seleccionados en busca del primer intercambio de i^* que mejora el valor de M.

Obsérvese que cuando un movimiento de mejora es identificado, éste se realiza sin examinar los elementos restantes en $V \setminus M$. Esta fase de mejora se lleva a cabo siempre y cuando la solución se haya mejorado.

A la fase de construcción adaptada que se mostró en la Figura 24, más esta fase de búsqueda local, es lo que denominamos heurística GRASP2a.

De manera similar al planteamiento del método GRASP1b, se plantea el método GRASP2b que consiste en uso del algoritmo de la Figura 24 para todos los posibles valores de m, construyendo una solución para cada valor, y luego aplicando la fase de mejora con el procedimiento de búsqueda local descrito para GRASP2a. A este nuevo procedimiento constructivo más la fase de búsqueda local denominamos método GRASP2b, tal como se muestra en la Figura 25.

Mejor solución=0;

1. Para $m = 2,3, \dots, n$ Hacer:
   2. Seleccionar aleatoriamente un elemento i^* en $V = \{1, 2, \dots, n\}$
   3. Poner $M_1 = \{i^*\}$ y $k = 1$
   4. Seleccionar m aleatoriamente en el intervalo $[2, m]$

Mientras $(k < m)$:

   5. Calcular $L = \{1, 2, \dots, n\} \setminus M_k$
   6. Calcular $d_s(i, M_k), \forall i \in L,\ div_{sum\_min}(M_k)$ y $div_{sum\_max}(M_k)$
   7. Construir la lista RCL con los elementos que satisfagan la ecuación (3.1)
   8. Seleccionar aleatoriamente un elemento i^* en RCL
   9. $M_{k+1} = M_k \cup \{i^*\}$
   10. $valor = \frac{(k)valor + \sum_{j \in M_k} d_{i^* j}}{k+1}$
   11. $k = k + 1$
   12. Si $valor$ > mejor solución, mejor solución = $valor$; $M = M_m$

   FASE DE MEJORA: Mientras $mejora = Verdadero$:

   13. Calcular $i^* = \min_{i \in M}\{d_s(i, M)\}$
   14. Si para algún $i^{**} \in V \setminus M$: $valor + \frac{\sum_{j \in M}\left(d_{i^{**} j} - d_{i^* j}\right)}{m} < valor$,

       $valor = valor + \frac{\sum_{j \in M}\left(d_{i^{**} j} - d_{i^* j}\right)}{m}, M = (M \cup \{i^{**}\}) - \{i^*\}$

       Si no $mejora = Falso$

Figura 25. Algoritmo GRASP 2*b*

4.3. Heurística GRASP3

En esta sección se propone una nueva heurística, para obtener soluciones de mejor calidad para el problema Max-Mean. Esta heurística consiste de una fase de construcción GRASP, con una fase de búsqueda local basada en la metodología de búsqueda en vecindades variables (Variable Neighborhood Search). A este método lo denominamos heurística GRASP3, que posteriormente es mejorado con la incorporación de una fase de post procesamiento, basada en la metodología de Rencadenamiento de Trayectorias (Path Relinking), que denominamos heurística GRASP3+PR.

4.3.1. Fase de Construcción de GRASP3

Con base en el comportamiento de los valores óptimos para el problema Max-Sum observado en las Figura 18, Figura 19 y Figura 21, se diseñó un nuevo método constructivo de tipo GRASP, en el cual añadimos nuevos elementos a la solución parcial bajo construcción siempre que el valor de la función objetivo del problema Max-Mean mejore, y cuando se observa un decrecimiento de este valor, se detiene la fase de construcción. De esta manera m no se establece aleatoriamente como en los métodos GRASP1a y GRASP2a, o se explora para todos sus valores posibles, como en GRASP1b y GRASP 2b, sino que el método selecciona por sí mismo, utilizando un mecanismo inteligente, el valor de m que parece más adecuado al problema.

En lugar de una típica construcción GRASP, establecida en (15), en la cual para construir la Lista de Candidatos Restringida, *RCL*, primero cada elemento candidato a seleccionarse es evaluado por una función voraz (Greedy) y luego un elemento es seleccionado aleatoriamente, se utiliza un diseño alternativo, en concordancia con lo propuesto en estudios recientes, que se expone en (35), mediante lo cual primero aplicamos la aleatorización y luego la parte voraz, permitiendo obtener mejores resultados.

Así, en nuestro método constructivo para el problema Max-Mean seguimos esta última estructura, primero elegimos aleatoriamente candidatos para el conjunto RCL y luego evaluamos cada candidato de acuerdo a la función voraz, seleccionando el mejor candidato.

En términos matemáticos, dada una solución parcial M_k con k elementos seleccionados, la lista de candidatos CL está formada por los $(n-k)$ elementos no seleccionados. La lista de candidatos restringida, RCL, contiene una fracción $\alpha\ (0 < \alpha < 1)$ de los elementos de CL seleccionados aleatoriamente, donde α es un parámetro que debe elegirse adecuadamente, generalmente por experimentos computacionales.

Luego, cada elemento $i \in RCL$ es evaluado de acuerdo al cambio que produce en la función objetivo; es decir, se calcula:

$$eval(i) = div(M_k \cup \{i\}) - div(M_k)$$

Donde $div(\cdot)$ es la función de diversidad del promedio definida en la ecuación (1.6).

Después, el método selecciona el mejor candidato i^* en RCL si esto mejora la solución parcial actual; es decir, si $eval(i^*) > 0$, y lo añade a la solución parcial, $M_{k+1} = M_k \cup \{i^*\}$; en caso contrario, si $eval(i^*) \leq 0$, el método para.

La Figura 26 muestra el seudo código de esta fase de construcción del método que denominamos heurística GRASP3.

1. Seleccionar aleatoriamente un elemento i^* en $V = \{1, 2, \dots, n\}$
2. Poner $M_1 = \{i^*\}$, $k = 1$ y $mejora = 1$.

Mientras $(mejora = 1)$:

3. Calcular $CL = \{1, 2, \dots, n\} \setminus M_k$
4. Construir RCL a partir de α $|CL|$ elementos aleatoriamente seleccionados de CL
5. Calcular $eval(i) = div(M_k \cup \{i\}) - div(M_k)$ $\forall i \in RCL$
6. Seleccionar el elemento i^* en RCL con el máximo valor de $eval$

Si ($eval(i^*) > 0$):

7. $M_{k+1} = M_k \cup \{i^*\}$
8. $k = k + 1$

Si no

9. $mejora = 0$

Figura 26. Fase de construcción del método GRASP3

4.3.2. Búsqueda Local en GRASP3

La construcción obtenida a través de un algoritmo con base en GRASP por lo general no permite obtener un óptimo local, y es costumbre aplicar luego de la construcción de una solución un método de búsqueda local. Como se mostró en la sección 4.2, los métodos previos de búsqueda local para los problemas de diversidad máxima se auto limitan a intercambiar elementos seleccionados con elementos no seleccionados, dejando constante el número m de objetos seleccionados. Ya que no se tiene esta restricción en el problema Max-Mean y admitimos

soluciones con cualquier valor de m, podemos considerar una clase vecindades más extensa y general.

Más específicamente, con base en la metodología de Descenso en Vecindades Variables (VND por sus siglas en inglés), descrita en (16), consideramos la combinación de tres tipos de vecindad en nuestro procedimiento de búsqueda local:

- Vecindad tipo N_1: Cuyos elementos se obtienen al remover un elemento de la solución actual, reduciendo así el número de elementos seleccionados en una unidad.
- Vecindad tipo N_2: Resultado de intercambiar un elemento seleccionado, con uno no seleccionado, manteniendo así constante el número de elementos seleccionados.
- Vecindad tipo N_3: Se obtiene al añadir un elemento no seleccionado en la solución actual, incrementando así el número de elementos seleccionados en una unidad.

El orden de exploración de las vecindades está dado para intentar, en lo posible, disminuir el número de elementos seleccionados, aumentando a su vez la diversidad, lo cual sucede cuando se encuentra una mejor solución al explorar N_1. Si esto no es posible, podemos conservar la cardinalidad del conjunto seleccionado con el compromiso de aumentar la diversidad, tal como sucede al explorar la vecindad N_2. Por último, al explorar N_3, estamos dispuestos a aumentar la cardinalidad del conjunto seleccionado en aras de aumentar su diversidad. Los experimentos numéricos confirman que este orden de seleccionar las vecindades es adecuado, como se observa en la sección 5.3.2.

Más específicamente: dada una solución, M_m, la búsqueda local primero intenta obtener una solución en N_1 que la mejore. Si esto sucede, y logramos hallar M'_{m-1} con $div(M'_{m-1}) > div(M_m)$, entonces aplicamos el movimiento y consideramos M'_{m-1} como la solución actual. Caso contrario, el método recurre a explorar la vecindad N_2 y busca el primer intercambio que mejora M_m. Si esto sucede, y logramos hallar M'_m con $div(M'_m) > div(M_m)$, entonces aplicamos el movimiento y consideramos M'_m como la solución actual. En cualquier caso, sin tener en cuenta si

hallamos la solución mejorada en N_1 o en N_2, en la próxima iteración el método inicia explorando nuevamente N_1 para mejorar la solución actual.

Si ni la vecindad N_1 ni la vecindad N_2 contienen una mejor solución que la solución actual, entonces finalmente recurrimos a explorar N_3. Si esta exploración es exitosa, y logramos hallar M'_{m+1} con $div(M'_{m+1}) > div(M_m)$, entonces aplicamos el movimiento y consideramos M'_{m+1} como la solución actual, y luego regresamos a explorar N_1 en la siguiente iteración. Caso contrario, estaríamos en una situación en la que ninguna de las vecindades contiene una mejor solución que la solución actual, y por tanto el método termina.

Además, para acelerar la búsqueda en estas vecindades, no realizamos la exploración de manera secuencial sobre los elementos de una vecindad específica, sino que evaluamos su potencial contribución a la solución parcial dela siguiente manera: Dada una solución M_m, calculamos la contribución de cada elemento seleccionado i, así como la contribución potencial de cada elemento no seleccionado i como:

$$d_s(i, M_m) = \sum_{j \in M_m} d_{ij}$$

Entonces, cuando exploramos N_1 para remover un elemento de M_m, buscamos los elementos seleccionados en el orden dado por d_s, donde el elemento con el valor más pequeño se prueba primero. Similarmente, cuando exploramos N_2 vamos probando los elementos seleccionados en el mismo orden pero los elementos no seleccionados en el "orden inverso"; es decir, primero consideramos los elementos no seleccionados con una gran contribución potencial a la solución parcial.

Finalmente, cuando exploramos N_3 los elementos no seleccionados, que se consideran para ser añadidos en la solución actual, son explorados de la misma manera que en N_2, en la cual el elemento con la mayor contribución potencial es considerado primero. La Figura 27 muestra el seudo código de esta fase de mejora del algoritmo.

1. Sea M_m la solución inicial, determinada en la fase de construcción
2. Calcular $d_s(i, M_m)$ para todo elemento $i \in V$.

Mientras $(improve = 1)$

3. Explorar los elementos seleccionados en el orden de d_s para transformarlos en no seleccionados
   Si existe un movimiento en N_1 que mejora la solución actual,
4. Realizar el primer movimiento de mejora (N_1)
5. Sea M'_{m-1} la solución actual
   Sino
6. Buscar para realizar los intercambios en el orden de d_s
   Si existe un movimiento en N_2 que mejora la solución actual,
7. Realizar el primer movimiento de mejora (N_2)
8. Sea M'_m la solución actual
   Sino
9. Buscar los elementos no seleccionados en el orden reverso de d_s
   Si existe un movimiento en N_3 que mejora la solución actual,
10. Realizar el primer movimiento de mejora (N_3)
11. Sea M'_{m+1} la solución actual
    Sino
12. $improve = 0$

13. Actualizar los valores de d_s

Figura 27. Algoritmo de búsqueda local para GRASP3

4.3.3. GRASP3+Rencadenamiento de Trayectorias

Como se resumió de manera general en la sección 3.5.1, el algoritmo de Rencadenamiento de Trayectorias (PR por sus siglas en inglés), fue descrito por primera vez en el marco del método de búsqueda tabú, en (17), como una estrategia de intensificación de la búsqueda en regiones prometedoras del espacio de soluciones de un problema de optimización combinatoria. El método de Rencadenamiento de trayectorias opera sobre un conjunto de soluciones, denominado "Conjunto Élite" (ES por sus siglas en inglés), que se construye con la aplicación de un método previo, que en general es una metaheurística (originalmente el método de búsqueda tabú). En nuestro caso, este método previo es la heurística GRASP3, con el cual se genera el conjunto élite

considerando tanto la calidad, como la diversidad, como las características deseables de sus elementos.

La estrategia que aplicamos para diseñar el nuevo algoritmo (GRASP3+PR) es la siguiente:

- Inicialmente ES es vacío, y se aplica GRASP3, para $b = |ES|$ iteraciones, con esto poblamos inicialmente el conjunto élite con estas soluciones obtenidas, ordenando las soluciones en ES con base en su valor objetivo desde la mejor, representada con x^1, a la peor, representada con x^b. Nótese que ES es por tanto un conjunto ordenado de soluciones.
- Una vez poblado el conjunto ES por b soluciones iniciales, en las siguientes iteraciones del algoritmo se determina cuando una nueva solución generada, x', califica o no para entrar a ES, y cuál solución que estaba en ES deberá salir, con el fin de que la cardinalidad de ES permanezca constante e igual a b.
- Específicamente, una nueva solución x' ingresa al conjunto ES si se cumple una de las siguientes condiciones:
  1. Si x' es mejor que x^1, esta solución ingresa al conjunto élite, esto indica que se privilegia la calidad como una condición esencial para pertenecer al conjunto ES.
  2. Si x' es peor que x^1 pero mejor que x^b y además es suficientemente diferente que las otras soluciones del conjunto ES, x' también ingresa al conjunto ES, esto con el fin de favorecer la diversidad del conjunto de soluciones élite. Ahora, para ver esto, considerar que x' es suficientemente diferente que cualquier otra solución en ES, si la distancia entre x' y ES, es mayor que cierto umbral que representaremos con dth; esto es, x' ingresa al conjunto ES, si se satisface la condición (3.2):

$$x' \text{ es mejor que } x^1, \text{ o, } x' \text{ es mejor que } x^b \text{ y } d(x', ES) \geq dth \qquad (3.2)$$

  Donde $d(x', ES)$ representa la distancia entre x' y ES, luego tenemos que definir qué es esta distancia y cómo debe ser calculada.
- Para conservar la cardinalidad de ES constante e igual a b, cuando se añade un elemento a este conjunto debemos remover alguno de los que ya estaban presentes. Y con el fin de privilegiar la calidad y la diversidad del conjunto ES, se remueve aquella solución de peor valor que está más cercana a x' en ES. Para declarar que una solución está cercana o

lejana a x' en ES, debemos considerar una medida de distancia entre esta solución y el conjunto de soluciones élite.

En consecuencia, una solución del problema Max-Mean x, puede ser interpretada como un vector binario con n coordenadas, $x = (x_1, x_2, \dots, x_n), \forall i \in \{1,2, \dots, n\}: x_i \in \{0,1\}$, donde la coordenada x_i toma el valor 1 si el elemento i está seleccionado en esa solución y 0 en caso contrario, con esta representación la distancia entre las dos soluciones puede ser calculada como se muestra en la ecuación (3.3):

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{3.3}$$

Es decir, $d(x, y)$ indica el número de elementos que no están simultáneamente en ambas soluciones x, y, o desde otro punto de vista, el número de elementos en los que difieren las dos soluciones.

Así entonces la distancia entre una solución x' y el conjunto de soluciones élite ES, $d(x', ES)$, puede ser calculada como la suma de las distancias entre x' y todos los elementos de ES.

$$d(x', ES) = \sum_{y \in ES} d(x', y) \tag{3.4}$$

Para establecer completamente la forma de evaluar la condición (3.2), se implementó el parámetro de umbral dth como un porcentaje φ del tamaño total de la población, n, representado como dth^* multiplicado por la cardinalidad del conjunto élite de soluciones, b; es decir:

$$dth = (\varphi\, n)\, b = dth^*\, b \tag{3.5}$$

La razón de la ecuación (3.5) es que $d(x', ES)$ es la suma de las distancias desde x' hasta cada elemento del conjunto ES, que tiene cardinalidad b. Si consideramos que dos soluciones son suficientemente diferentes cuando en promedio difieran en un porcentaje φ de sus elementos, de acuerdo a la ecuación (3.3) tenemos que $d(x', y) \geq \varphi\, n$, y entonces si se usa la fórmula (3.4) para la evaluación de la distancia de x' a ES, se tiene finalmente:

$$d(x', ES) \geq \varphi\ n\ b.$$

Así, se ha descrito de manera precisa la forma en que una nueva solución puede ingresar a ES y reemplazar a una solución del conjunto élite de soluciones.

- Cuando se hace esto para cierto número de iteraciones se tiene finalmente poblado el conjunto ES y entonces, posteriormente, se realiza la fase de intensificación, con el procedimiento de Rencadenamiento de Trayectorias propiamente dicho:

  Dadas dos soluciones x, $y \in ES$, el procedimiento de Rencadenamiento de trayectorias $PR(x, y)$ actúa sobre estas dos soluciones para obtener una mejor solución de la siguiente manera: Inicia con la primera solución x, denominada ***solución inicial***, y por medio de algún mecanismo de transformación gradualmente se la transforma en la solución final y, denominada ***solución guía***. Para lograr esto en cada iteración de la transformación se consideran dos movimientos:

  - Remover un elemento que está en x pero que no está presente en y; o,
  - Añadir un elemento que no está presente en x pero que si está presente en y.

  El método selecciona el mejor de estos movimientos, originando la primera ***solución intermedia*** de la trayectoria, representada con $x(1)$. Luego se consideran nuevamente los dos movimientos para $x(1)$: remover un elemento en $x(1)$ que no está presente en y, o añadir un elemento que no está presente en $x(1)$ pero que si está presente en y. El mejor de estos movimientos origina la segunda solución intermedia $x(2)$.

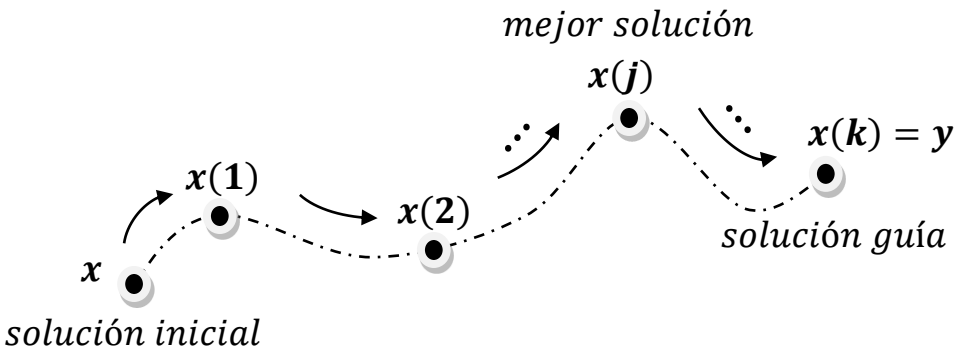


Figura 28. Esquema representativo del Rencadenamiento de Trayectorias

De esta manera, si operamos iterativamente, generamos la trayectoria de soluciones intermedias $x(1), x(2), \dots, x(j), \dots, x(k) = y$ hasta alcanzar a y, como se representa en la Figura 28. La salida de esta fase del algoritmo PR es la mejor solución intermedia de esta trayectoria, digamos $x(j)$, diferente de x y de y.

A continuación, se aplica a esta mejor solución encontrada la fase de mejora por el procedimiento de búsqueda local. En nuestra investigación realizamos el procedimiento PR para ir de x a y $\left(PR(x, y)\right)$ y también PR para ir de y a x $\left(PR(y, x)\right)$, de donde se obtiene una solución x', que será considerada como la mejor solución si su valor objetivo supera a la mejor solución de ES, como se observa en la Figura 29.

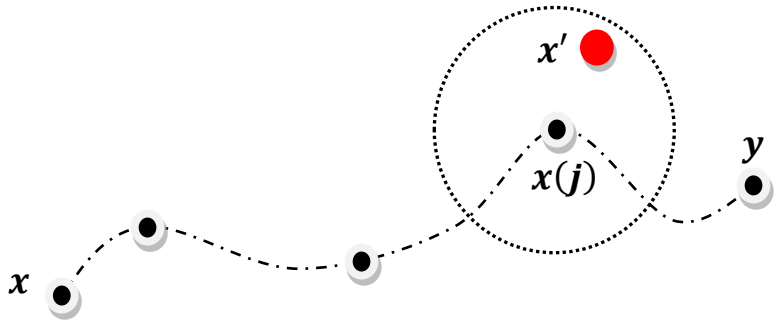


Figura 29. Esquema del procedimiento *PR* en su fase de búsqueda local

Finalmente, se resume la fase de Rencadenamiento de trayectorias que se propone en la Figura 30, en la que se muestra su seudo código.

Al algoritmo total que incluye las fases de construcción búsqueda local y Rencadenamiento de trayectorias denominamos GRASP3+PR.

1. Poner $Globaliter$ igual al número global de iteraciones elegida.
2. Aplicar el método GRASP3 (construcción más mejora) por $b = |ES|$ iteraciones para poblar el conjunto $ES = \{x^1, x^2, \dots, x^b\}$
3. Poner $iter = b + 1$

Mientras ($iter \leq Globaliter$):

4. Aplicar la fase de construcción GRASP ⇒ x
5. Aplicar la fase de búsqueda local de GRASP a $x \Rightarrow x'$

Si ($f(x') > f(x^1)$ o ($f(x') > f(x^b)$ y $d(x', ES) \geq dth$)):

6. Sea x^k la solución más cercana a x' en ES con $f(x') > f(x^k)$
7. Añadir x' a ES y remover x^k.
8. Actualizar el orden del conjunto ES (desde el mejor x^1 al peor x^b).

9. Sea $x^{best} = x^1$

Para ($i = 1$ hasta $i = b - 1$ y desde $j = i + 1$ hasta $j = b$)

10. Aplicar $PR(x^i, x^j)$ y $PR(x^j, x^i)$, sea x la mejor solución hallada
11. Aplicar la fase de búsqueda local de GRASP3 a $x \Rightarrow x'$

Si $\left(f(x') > f(x^{best})\right)$:

12. $x^{best} = x'$

13. Salida x^{best}

Figura 30. Seudo Código de la fase de Rencadenamiento de Trayectorias del algoritmo GRASP3+PR

4.3.4. Nuevas adaptaciones de los métodos GRASP1 y GRASP2 existentes

Podría pensarse que el método propuesto GRASP3+PR tiene una ventaja a priori sobre los métodos GRASP1 y GRASP2, en el hecho de que en estos últimos no se utiliza dentro de su diseño la característica observada en la Figura 18 y la Figura 19, que podría ser crucial en la eficiencia de los algoritmos desarrollados. Por este motivo se proponen dos adaptaciones adicionales a los métodos previos GRASP1 y GRASP2. Así, el principio aplicado en nuestro método constructivo GRASP3 descrito en la sección 4.3.1, que fue visualizado de la experimentación de la sección 4.1, puede usarse para adaptar los métodos previos diseñados por investigadores para tratar los problemas similares Max-MinSum y Max-Sum. En particular, en lugar de escoger un número *a priori* de elementos, o un número escogido aleatoriamente como en nuestra adaptación GRASP1a, podemos ir añadiendo secuencialmente, uno a la vez, elementos a la solución parcial bajo

consideración mientras el valor de la función objetivo Max-Mean mejore, y cuando este valor empiece a decrecer paramos la construcción. Al método constituido por esta fase de construcción, más la fase de mejora considerada en GRASP1a, lo denominamos método GRASP1c. Similarmente llamamos método GRASP2c a esta adaptación en la fase constructiva del método originalmente propuesto en (11) para resolver el problema Max-Sum.

Con esto tenemos las adaptaciones GRASP1a, GRASP1b y GRASP1c al método propuesto por (18) para resolver originalmente el problema Max-MinSum, y las adaptaciones GRASP2a, GRASP2b y GRASP2c al método propuesto por (11) para resolver el problema Max-Sum, modificadas para resolver el problema Max-Mean.

Capítulo 5.

Resultados numéricos con los problemas de prueba

Este capítulo contiene los resultados de una gran cantidad de experimentos computacionales que se realizaron en esta investigación para evaluar y calibrar los métodos planteados en el Capítulo 4.

En particular aquí se describen la calibración de los parámetros que intervienen en los métodos desarrollados, y la comparación de los resultados obtenidos utilizando nuestras heurísticas GRASP3 Y GRASP3+PR y los métodos GRASP1a, GRASP1b, GRASP1c, GRASP2a, GRASP2b y GRASP2c, adaptados para la resolución del problema Max-Mean de métodos extraídos del estado del arte para resolver otros problemas de la diversidad máxima. También se hace una comparación con los resultados que se obtienen al aplicar un software comercial de optimización basado en algoritmos genéticos, más específicamente se utilizó Evolver V.5.5[3], el cual se anuncia como un paquete computacional que permite resolver una amplia variedad de problemas de optimización usando algoritmos genéticos y que permite encontrar la mejor solución global a complejos problemas del mundo real, como se puede ver en (39).

Todas las pruebas fueron realizadas con los conjuntos de datos que se describen en la sección 4.1.1, generados teóricamente pero cuya estructura suponemos presente en conjuntos de datos correspondientes a grupos de personas. Estos conjuntos de datos de datos son denominados de Tipo I y de Tipo II:

- Tipo I: Este conjunto de datos consiste de 60 matrices simétricas con números aleatorios en el intervalo $[-1,1]$ generados con una distribución uniforme. Estas 60 matrices corresponden a problemas pequeños, medianos y grandes, más específicamente, se tienen 10 matrices para cada uno de los siguientes tamaños del problema: $n = 20, 25, 30, 35, 150\ y\ 500$.

- Tipo II: Este conjunto de datos consiste de 60 matrices simétricas con números aleatorios en el conjunto $[-1,-0.5] \cup [-0.5,-1]$ generados con una distribución

[3] Evolver fue originalmente desarrollado por Axcelis Inc. en 1990, ahora pertenece a Palisade Corporation.

uniforme y de acuerdo al esquema indicado en la Figura 16. Estas 60 matrices corresponden a problemas pequeños, medianos y grandes, más específicamente, se tienen 10 matrices para cada uno de los siguientes tamaños del problema: $n = 20, 25, 30, 35, 150\ y\ 500$.

Todos los algoritmos fueron implementados en Mathematica V.7[4], que es una plataforma de procesamiento numérico con un lenguaje de programación propio, que ha resultado muy útil para la implementación de este tipo de algoritmos, tal como se plantea en (40).

Este lenguaje de programación permitió una implementación eficiente de los métodos descritos, en particular, la implementación en Mathematica permite la interacción con otros programas necesarios para esta investigación como Cplex, y una rápida y fácil importación y exportación de datos, especialmente archivos planos, así como una gran potencia para generar gráficos de gran calidad y precisión.

Los experimentos computacionales se procesaron en una Laptop Intel Core 2, 1.4 GHz y 2GB de RAM.

5.1. Calibración del parámetro α

Para la fase constructiva de nuestra heurística GRASP3 existen varios parámetros que deben ser calibrados, en particular un valor muy importante es el parámetro α, pues éste determina la diversidad adecuada que se requiere para obtener buenas soluciones. Para una buena calibración del parámetro α, se ejecutó la fase constructiva del método GRASP3 por 100 iteraciones, para todos los posibles valores de α, desde $\alpha = 0.1$ hasta $\alpha = 0.9$, con un paso de 0.1, para cada ejemplo de prueba de tipo I de tamaño mediano $(n = 150)$, luego se elaboró el perfil de búsqueda del promedio de los mejores valores obtenidos, esto se observa en la Figura 31, en la cual el valor (value) está multiplicado por 10 para apreciar los detalles.

Como el mejor comportamiento se tiene para valores de α entre 0.3 y 0.7, en la Figura 32 se observa la evolución del promedio del mejor valor para esos valores de α. Es claro que el valor de α que da mejor rendimiento a la fase constructiva en instancias de Tipo I es $\alpha = 0.6$.

[4] **Mathematica** es una robusta herramienta especializada en análisis numérico y cálculo simbólico, que incorpora un potente lenguaje de programación propio y una interfaz externa que permite salidas a C, Fortran y TEX, es una marca registrada de Wolfram Research Inc.

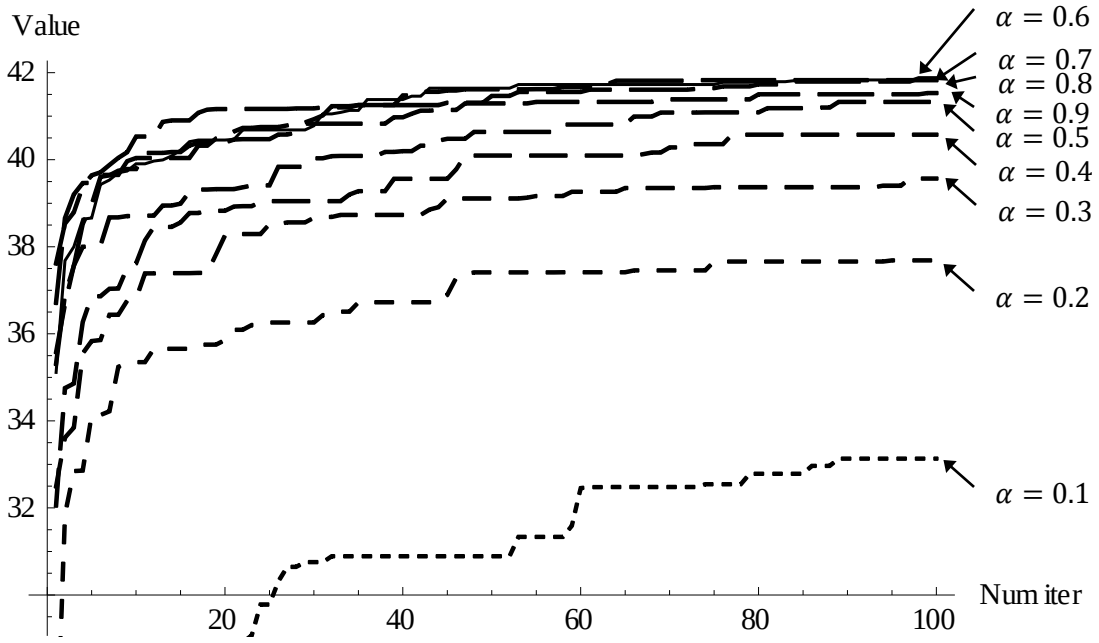


Figura 31. Perfil de búsqueda para las construcciones GRASP en problemas de prueba tipo I

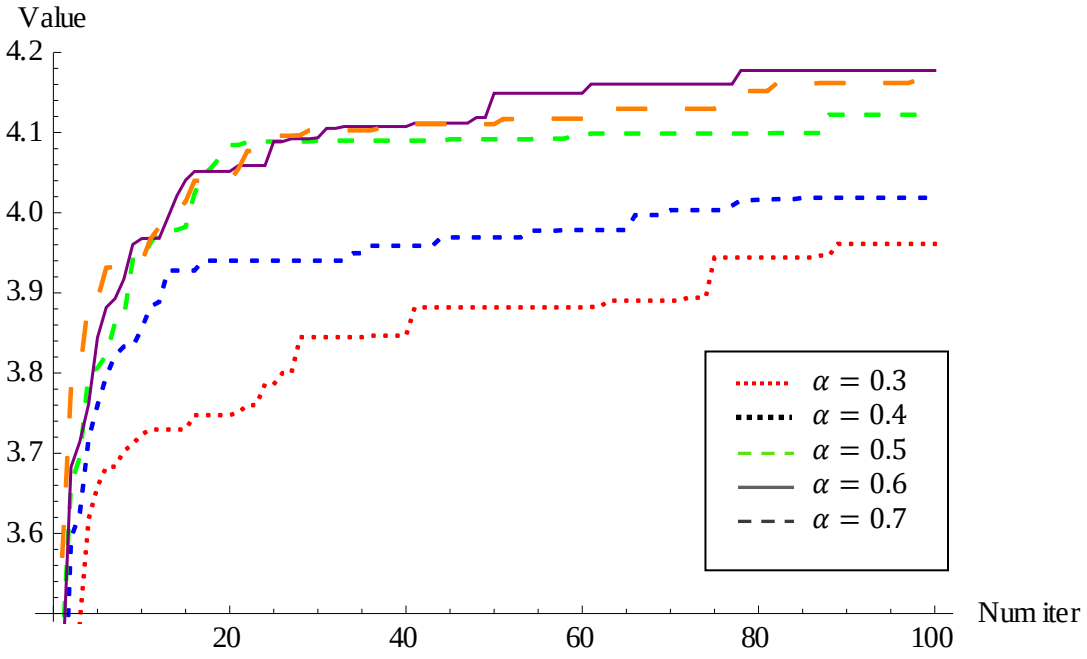


Figura 32. Perfil de búsqueda de la fase constructiva de GRASP 3 para los valores de α más significativos para ejemplos tipo I

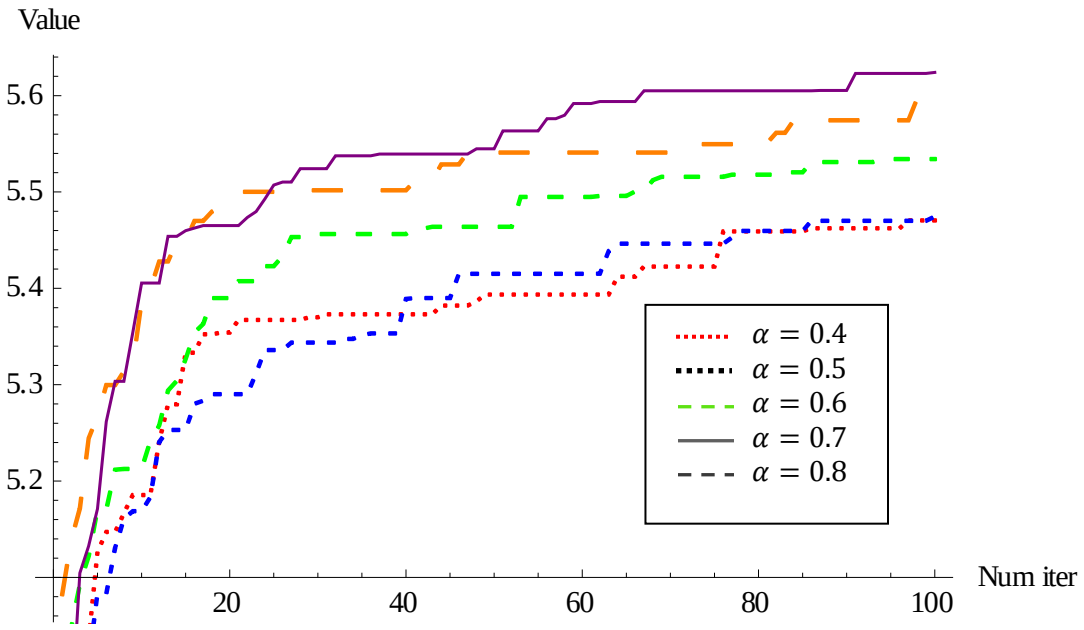


Figura 33. Perfil de búsqueda de la fase constructiva de GRASP 3 para distintos valores de α para ejemplos tipo II

En cambio en la Figura 33 se muestra el perfil de búsqueda de la fase de construcción de la heurística GRASP3 para los problemas de prueba de tipo II, en ella se observa que el mejor valor para α es $\alpha = 0.7$, con el que se obtuvieron consistentemente mejores resultados.

Así, se puede concluir que el valor de α que más se adecúa depende del tipo de problema a resolver, pero podría ser escogido en el intervalo $[0.6, 0.7]$.

5.2. Rendimiento de las heurísticas en problemas pequeños

En esta sección se realiza una comparación del rendimiento de las heurísticas adaptadas GRASP1α y GRASP2α, las desarrolladas en esta investigación GRASP3 y GRASP3+PR, y los óptimos exactos reportados para problemas pequeños. Los resultados se muestran en la Tabla 15.

Se usaron instancias pequeñas de tamaño $n = 30$, que son las más grandes para las que se pudo resolver con Cplex de manera exacta en tiempos razonables. Como el óptimo es conocido, una medida de la precisión de los métodos es la diferencia porcentual relativa respecto al óptimo (GAP), que se define como:

$$GAP = \frac{valor\ óptimo - valor\ encontrado}{valor\ óptimo} \times 100\%$$

En primer lugar se ejecutaron únicamente las fases constructivas de los algoritmos GRASP1a, GRASP2a y GRASP3, sin la fase de post procesamiento de mejora local, con el objetivo de observar la calidad de las soluciones construidas. En la Tabla 15 se presentan los resultados de los promedios para las 10 instancias de tamaño $n = 30$, en problemas de tipo I y II, se incluye el valor promedio de la función objetivo (Valor), el número promedio de elementos seleccionados (m), las veces que se alcanzó el óptimo (# veces óptimo), la diferencia relativa porcentual con el óptimo (GAP) promedio y el tiempo promedio que requirió el procesamiento computacional en segundos (Tiempo CPU). En todas las heurísticas se consideró el mismo número de iteraciones igual a 100. La razón por la cual no se presentan los resultados del procesamiento con las heurísticas GRASP1b, GRASP1c, GRASP2b y GRASP2c es que GRASP1a y GRASP2a son más eficientes, como se verá en la sección 5.3.1.

Tabla 15. Rendimiento de los métodos en problemas pequeños

		GRASP1a constructivo	**GRASP2a constructivo**	**GRASP3 constructivo**	**Cplex**
Tipo I	Valor	1.466543	1.52726	1.87351	1.874955
	m	11.4	13.1	10.8	10.7
	# veces óptimo	0	0	9	10
	GAP	22.307%	19.225%	0.084%	0%
	Tiempo CPU (s)	2.90896	1.22184	0.35444	102.303
Tipo II	Valor	1.863813	1.853205	2.377163	2.383
	m	13.1	14.1	10.3	10.8
	# veces óptimo	0	0	6	10
	GAP	21.807%	22.137%	0.397%	0%
	Tiempo CPU (s)	4.87984	1.142	0.3444	182.176

Se puede observar que únicamente aplicando la fase constructiva de GRASP3 ya se alcanza al óptimo exacto del problema el 90% de las veces, para las instancias de prueba de tipo I, y el 80% de las veces en las instancias de prueba de tipo II, y en un tiempo muy reducido (menos de un segundo), además en las instancias en que no se encontró el óptimo, el GAP es muy pequeño.

En la Tabla 15 consta el resumen de los resultados promedio, mientras que en la Tabla A 3 del ANEXO 1 se muestran los resultados detallados para cada uno de los problemas de prueba con los que se obtuvieron estos promedios.

En lo que sigue, se presentan los resultados de las fases constructivas de GRASP1a, GRASP2a y GRASP3 con el procedimiento de búsqueda local y su comparación con los valores óptimos encontrados con Cplex. En la Tabla 16 se muestran los resultados, se puede observar cómo la búsqueda local mejora sustancialmente los resultados obtenidos por el método constructivo en GRASP1a y GRASP2a, para las cuales se realizaron 100 iteraciones, mientras que GRASP3 acertó al óptimo en todos los casos y en un tiempo sumamente pequeño. En la Tabla A 4 del ANEXO 1 se observan los resultados particulares con cada uno de los problemas de prueba de tamaño $n = 30$ cuando se aplicaron las heurísticas completas.

En conclusión, se puede decir que en promedio las heurísticas GRASP1a y GRASP2a reportan resultados de la función objetivo equivalentes en calidad, pero la heurística GRASP1a requiere más del doble de tiempo para encontrar estos resultados en comparación con la heurística GRASP2a, por otro lado GRASP3 se muestra como la heurística más eficiente, permitiendo encontrar el óptimo en todos los casos en un tiempo muy reducido.

Tabla 16. Resultados de las heurísticas con búsqueda local para problemas pequeños $n = 30$

		Con búsqueda local			
		GRASP1a	**GRASP2a**	**GRASP3**	**Cplex**
Tipo I	Valor	1.842227	1.841831	1.874955	1.874955
	m	10.9	11.3	10.7	10.7
	# veces óptimo	4	5	10	10
	GAP	1.812%	1.926%	0%	0%
	Tiempo CPU (s)	7.9091	3.3088	0.4135	102.303
Tipo II	Valor	2.345667	2.35521	2.383	2.383
	m	11.4	11.4	10.8	10.8
	# veces óptimo	4	5	10	10
	GAP	1.682%	1.268%	0%	0%
	Tiempo CPU (s)	8.1357	3.1777	0.3634	182.176

La razón por la que únicamente se consideró las heurísticas GRASP1a y GRASP2a, y no las heurísticas GRASP1b, GRASP1c y GRASP2b, GRASP2c se la verá en la sección 5.3.

5.3. Solución de problemas medianos con GRASP1, GRASP2 y GRASP3.

La resolución de problemas pequeños mostrada en la sección 5.2 permitió comparar las soluciones obtenidas por medio de las heurísticas con la solución exacta, que solo se puede obtener en tiempos razonables para esas tallas de los problemas; sin embargo, no permiten observar cuál es un rendimiento en problemas medianos o grandes de tamaño real.

En esta sección se presentan los resultados de aplicar estos algoritmos y sus implementaciones en Mathematica a los problemas de prueba de tamaño mediano $(n = 150)$ y grande $(n = 500)$. También se demuestra que las estrategias GRASP1b, GRASP1c no resultan más eficientes que GRASP1a, y de manera similar GRAS2b y GRASP2c no son mejores que GRASP2a, esto quiere decir que para estos métodos es suficiente, dentro del proceso constructivo, seleccionar aleatoriamente el valor de m.

5.3.1. Comparación de los resultados y análisis de la eficiencia de los algoritmos

Como ya no es posible comparar con la solución óptima exacta de estos problemas, en lugar del GAP se reporta el porcentaje de desviación respecto a la mejor de las soluciones encontradas en los experimentos, valor representado en las tablas como desviación, y que es igual a:

$$Desviación = \frac{mejor\ solución - solución\ encontrada}{mejor\ solución} \times 100\%$$

La Tabla 17 resume los resultados promedio para los diez problemas de prueba de este tamaño, mientras que la Tabla A 5 y la Tabla A 6 del ANEXO 1 muestran los resultados detallados para cada uno de los diez problemas de prueba de tipo I y tipo II con los que se realizó el experimento. Estos resultados reportados en la Tabla 17 confirman lo que se observó al procesar los problemas de tamaño pequeño, la fase constructiva del algoritmo GRASP3 genera soluciones de mucha mejor calidad que las fases constructivas de las heurísticas GRASP1 y GRASP2, y además en menos tiempo, además se observa que para ambas clases de tipos de problema, la eficiencia de

los algoritmos es parecida respecto al mejor valor encontrado. Es interesante observar que las mejores soluciones obtenidas con la fase constructiva de GRASP3 presentan en promedio un número más pequeño de elementos seleccionados (m) que las mejores soluciones encontradas por las fases constructivas de GRASP1 y GRASP2, tanto en los problemas de prueba de tipo I como en los de tipo II.

Por otro lado, en la Tabla 17 se observa que, en cuanto a los métodos adaptados de heurísticas previas, GRASP1a tiene un mejor rendimiento que sus adaptaciones GRASP1b y GRASP1c, si bien es cierto GRASP1b proporciona un mejor valor de la función objetivo respecto a GRASP1a, esta mejora solo es de 8% pero a costa de un incremento en el tiempo de ejecución del 300%. En cambio GRASP1c presenta un deterioro en el valor de la función objetivo. Igual comportamiento se puede observar al comparar GRASP2a, GRASP2b y GRASP2c. Es decir las adaptaciones de tipo a de estos algoritmos tienen mejor rendimiento que las adaptaciones tipo b y c. En el caso de las adaptaciones tipo b recorrer todos los valores posibles de m hace a los métodos muy costosos en términos de tiempo computacional requerido, que no se compensa con la mejora que se obtiene en el valor en la función objetivo.

En cambio los métodos tipo c tienen la particularidad que convergen rápidamente a un mal valor de la función objetivo, como en estos métodos el valor de m se va escogiendo secuencialmente desde 2, 3, etc. hasta que se observe una desmejora de la función objetivo, por la estructura de la función adaptativa, esto ocurre rápidamente, parando el algoritmo constructivo en valores pequeños de m. Debido a los resultados de esta comparación, en lo sucesivo se usarán únicamente las adaptaciones GRASP1a y GRASP2a de los métodos constructivos, completados con su fase de búsqueda local, y descartamos a los métodos tipo b y c. En lo sucesivo denominaremos simplemente método GRASP1 en lugar de GRASP1a y método GRASP2 en lugar de GRASP2a.

En nuestro siguiente experimento comparamos los algoritmos completos, que incluyen la fase constructiva más búsqueda local para estos problemas de talla mediana ($n = 150$), en la Tabla 18 se muestran los resultados promedio alcanzados.

Tabla 17. Resultados fase constructiva para problemas de tamaño $n = 150$

Tipo		GRASP1a	GRASP1b	GRASP1c	GRASP2a	GRASP2b	GRASP2c	GRASP3
I	Valor	2.495	2.830	2.258	2.875	3.205	1.410	4.145
	m	54.8	56.9	14.7	68.2	62	7.1	41.8
	# veces mejor	0	0	0	0	0	0	10
	Desviación	39.8 %	31.7 %	45.5 %	30.7 %	22.6 %	65.9 %	0.0 %
	CPU (s)	329.9	990.5	107.6	26.9	48.2	12.6	4.7
II	Valor	3.360	3.748	3.086	3.82	4.252	1.816	5.499
	m	61.6	59.8	14.6	80.7	69	7.2	40.1
	# veces mejor	0	0	0	0	0	0	10
	Desviación	38.8 %	31.8 %	43.8 %	30.5 %	22.6 %	66.8 %	0.0 %
	CPU (s)	230.1	595.8	71.4	25.714	48.62	12.17	5.02

Tabla 18. Resultados de los métodos GRASP (construcción + búsqueda local) en los ejemplos de prueba con $n = 150$

		GRASP1	**GRASP2**	**GRASP3**
Tipo I	Valor	3.9078	3.9596	4.3236
	m	49.4	45.7	44
	# veces mejor	0	0	10
	Desviación	9.63%	8.41%	0%
	Tiempo CPU (s)	241.6	61.5	26.4
Tipo II	Valor	4.943	5.223	5.684
	m	54.6	52.2	46.1
	# veces mejor	0	0	10
	Desviación	13.06%	8.14%	0%
	Tiempo CPU (s)	250.0	80.1	22.9

Los resultados de la Tabla 18 confirman la superioridad de la heurística propuesta con respecto a los métodos previos. Específicamente, GRASP3 obtiene una desviación promedio respecto a la mejor encontrada de 0% en problemas de prueba de tipo I o de tipo II, mientras

GRASP1 presenta desviaciones respecto a las mejores soluciones de 9.63% y 13.06% y GRASP2 desviaciones de 8.41% y 8.14% en los problemas de prueba de tipo I y tipo II, respectivamente.

Los datos desagregados con los resultados individuales de cada problema se muestran en la Tabla A 7 y la Tabla A 8 del ANEXO 1.

Para finalizar este análisis, y poder concluir que la heurística propuesta GRASP3 es más eficiente que las adaptadas de métodos previos, se debe determinar si estos resultados aportan evidencia estadística suficiente para respaldar esta afirmación.

La idea es comprobar si los resultados obtenidos por los diferentes métodos corresponden a poblaciones (de soluciones) distintas, o si por el contrario estos resultados no aportan suficiente evidencia estadística para afirmarlo. En este caso el contraste de la hipótesis de que varias muestras proceden de una misma población o de poblaciones con la misma distribución no puede realizarse mediante el análisis de la varianza, al incumplirse el supuesto de independencia de las muestras, ya que en este caso se trata de muestras apareadas. Por lo tanto se puede utilizar la prueba no paramétrica de rangos de Friedman.

Así, para fundamentar la conclusión, se aplicó la prueba estadística de Friedman para muestras emparejadas a los datos desagregados de los experimentos. El nivel de significancia alcanzado en esta prueba es de 0.000, lo cual claramente indica que hay diferencias estadísticamente significativas en los resultados obtenidos por estos tres métodos.

Un análisis típico post-test consiste en calificar a los métodos bajo consideración de acuerdo a los “Rank values” promedio calculados con esta prueba de hipótesis estadística. De acuerdo con este ranking, el mejor método es la heurística GRASP3, con un Rank value de 2.93, seguido por el método GRASP2 con un Rank value de 1.93 y finalmente el método GRASP1 con un Rank value de solo 1.15.

5.3.2. Perfil de la exploración en vecindades variables por GRASP3

Nuestra búsqueda local en la heurística GRASP3 utiliza tres tipos de vecindades diferentes, generadas de acuerdo con la metodología de descenso en Vecindades Variables, descrita en la sección 3.5.3, estas vecindades son representadas por:

- N_1 (remover un elemento de la solución),
- N_2 (intercambiar un elemento de la solución con uno que no esté seleccionado), y
- N_3 (añadir un elemento a la solución). Así, un estudio interesante es medir la contribución de cada tipo de vecindad a la calidad de la solución final.

En la Figura 34 se presenta un gráfico de barras con el número promedio de veces, en los 10 ejemplos de prueba de tipo I y 10 ejemplos de prueba de Tipo II de tamaño $n = 150$, en que la exploración de la vecindad de cada tipo mejoró la solución actual.

En ambos casos (instancias de Tipo I y de Tipo II), se repite el mismo patrón, esto es que en un mayor número de veces la exploración de N_2 mejoró las soluciones en todos los ejemplos de prueba, pero también la exploración de las vecindades N_1 y N_3 contribuyó a mejoras en la solución final, N_3 mejoró en más veces a la función objetivo que N_1 en todos los ejemplos. En la Tabla 19 se presentan los estadísticos correspondientes a este número de mejoras por la exploración de cada tipo de vecindad.

Tabla 19. Estadísticos del número de veces que la exploración de cada vecindad mejoró la solución en ejemplos de prueba de Tipo I y Tipo II

		Tipo de vecindad		
		N_1	N_2	N_3
Promedio total	Tipo I	3.9275	11.08	6.97
	Tipo II	4.64	10.41	6.524
Desviación estándar	Tipo I	1.0286	1.5164	0.636
	Tipo II	0.60955	1.97859	0.96834

Pero curiosamente, si calculamos cuál fue el aporte promedio a la mejora de la función objetivo que proporcionó la exploración en cada uno de estos tipos de vecindades, se observa otro comportamiento. Los experimentos numéricos indican que las vecindades tipo N_1 y N_3 hicieron más aporte promedio que las vecindades de tipo N_2, como se observa en la Figura 35 y en la Tabla 20. Esto significa que aunque más veces se mejoró por visitar una vecindad tipo N_2, estas mejoras fueron, en promedio, saltos muy pequeños, en cambio menos veces se mejoró por explorar las vecindades tipo N_1, pero esas mejoras fueron en promedio saltos más grandes y por tanto de mejor calidad; es decir, fueron menos movimientos pero sumaron más valor.

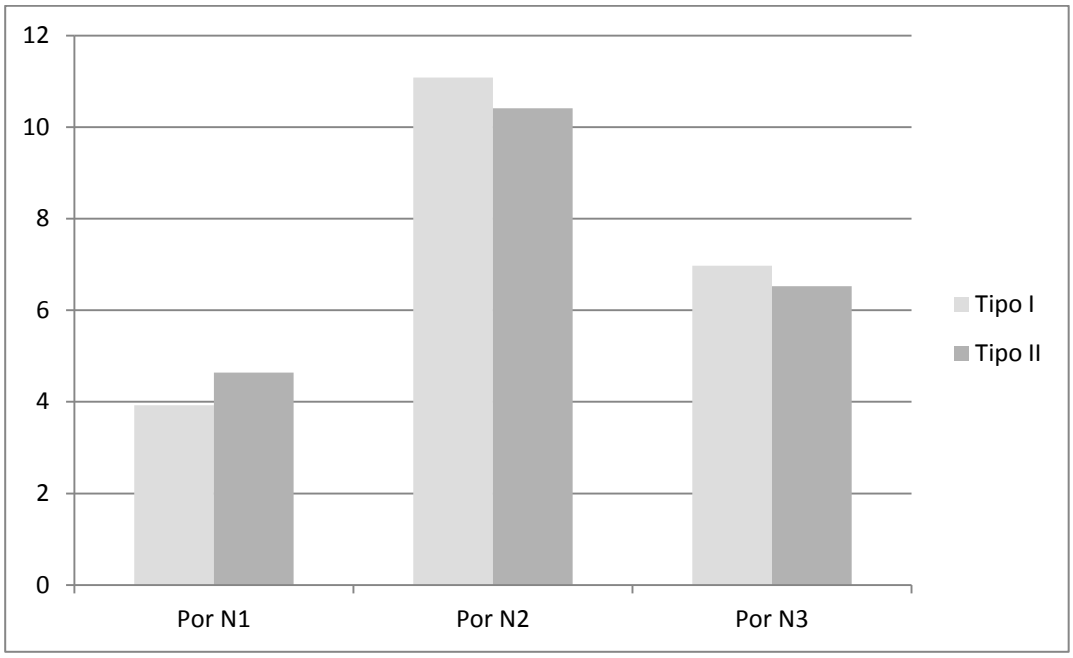


Figura 34. Número promedio de mejoras en cada tipo de vecindad en el algoritmo GRASP3 en los ejemplos de prueba de Tipo I y de Tipo II

Esto nos lleva a concluir, que en el problema Max-Mean es más eficiente y de mejor calidad un movimiento en el cuál se disminuye el número de elementos seleccionados que simplemente intercambiar un elemento seleccionado por otro no seleccionado dejando constante la cantidad de elementos seleccionados.

Tabla 20. Estadísticos del aporte a la mejora de la función objetivo por la exploración de cada tipo de vecindad

		Tipo de vecindad		
		Por N1	Por N2	Por N3
Promedio total	Tipo I	0.018837841	0.01181602	0.018171
	Tipo II	0.026249205	0.01641721	0.02397691
Desviación estándar	Tipo I	0.004336539	0.0010018	0.00186028
	Tipo II	0.002198756	0.00023477	0.0018417

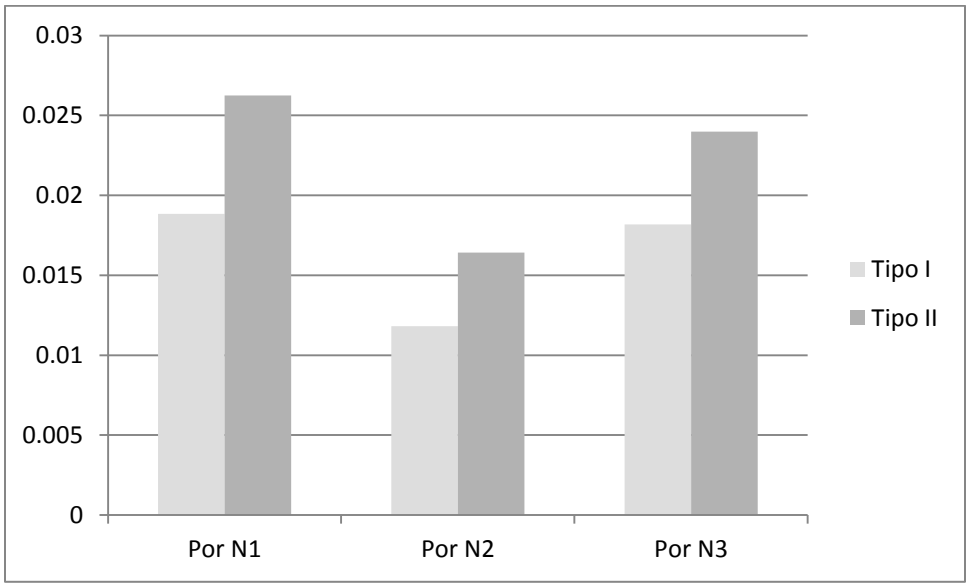


Figura 35. Aporte a la mejora de la función objetivo en valor por cada tipo de vecindad

5.4. GRASP3 con Rencadenamiento de Trayectorias para problemas grandes

En esta sección se describen los experimentos realizados con los 20 problemas de prueba grandes, de tamaño $n = 500$, y el uso de los algoritmos GRASP1, GRASP2, GRASP3 y GRASP3 con Rencadenamiento de Trayectorias. Se ha puesto el número global de iteraciones de tal forma que todos los métodos corran en tiempos CPU equivalentes.

5.4.1. Calibración de los parámetros

El tamaño del conjunto de soluciones élite en la fase de Rencadenamiento de Trayectorias, b, fue puesto en 20, que constituyó un número adecuado en los experimentos computacionales, pues valores más grandes no garantizaron la obtención de mejores soluciones y el tiempo de procesamiento se incrementó. La distancia umbral dth definida en la ecuación (3.5) se calibró considerando valores de φ entre 2% y 6%; es decir dth^* entre 10 y 30.

Para esta calibración del parámetro dth se realizaron experimentos computacionales con distintos valores de dth^* para cada uno de los problemas de prueba de tipo I y de tipo II de tamaño grande ($n = 500$), se calcularon las desviaciones respecto al mejor resultado obtenido

para cada uno de los ejemplos de prueba y se registró la desviación promedio para estos diferentes valores de dth^*.

Con la lista de parejas ordenadas $\{(dth^*, \%\text{ desviación promedio})/10 \leq dth \leq 30\}$ se construyeron los respectivos trazadores cúbicos (curvas Spline). En la Figura 36 se muestra el trazador para problemas tipo I y en la Figura 37 se muestra la curva para problemas tipo II, de ahí que usamos $dth^* = 25$ para problemas tipo I y $dth^* = 17$ para problemas tipo II como los valores sugeridos para este parámetro.

El valor del parámetro b en GRASP3 con Rencadenamiento de trayectorias fue puesto deliberadamente bajo, en 20, para que el tiempo de ejecución fuera equivalente a los tiempos de ejecución de GRASP1, GRASP2 y GRASP3. En GRASP3 el número de iteraciones se estableció en 250 para lograr esta equivalencia. Para obtener resultados con el mismo esfuerzo computacional en términos del tiempo CPU, GRASP3 se corrió con 250 iteraciones, mientras que GRASP3 con Rencadenamiento de trayectorias con 20 iteraciones.

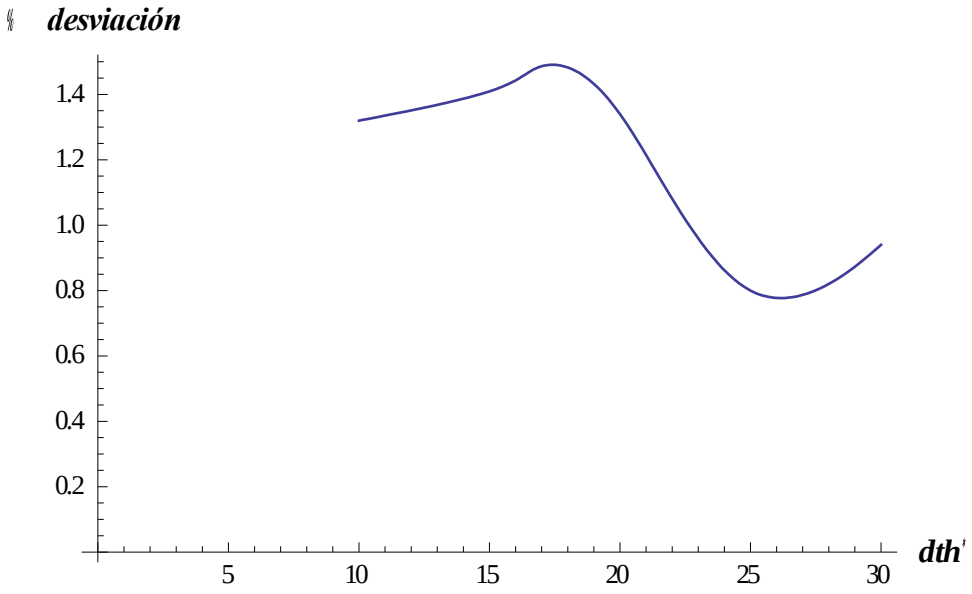


Figura 36. Curva de las desviaciones promedio respecto a los mejores resultados, obtenidos con Rencadenamiento de trayectorias usando distintos valores de dth^* para problemas tipo I

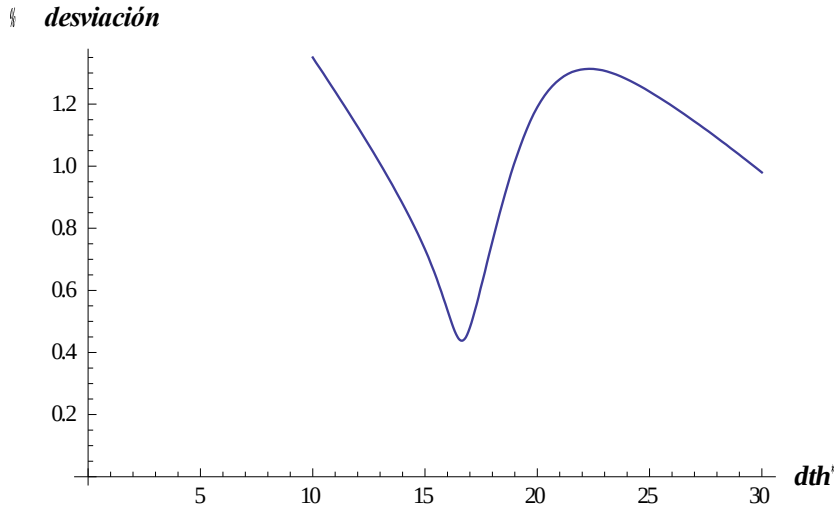


Figura 37. Curva de las desviaciones promedio respecto a los mejores resultados, obtenidos con Rencadenamiento de trayectorias usando distintos valores de dth^* para problemas tipo II

5.4.2. Resultados numéricos con instancias grandes

En la Tabla 21 se muestran el resumen de los resultados obtenidos en las instancias grandes al aplicar los cuatro algoritmos propuestos, los valores corresponden a los promedios alcanzados con cada uno de los 20 problemas de prueba, diez de tipo I y diez de tipo II.

Tabla 21. Comparación de los resultados obtenidos con GRASP1, GRASP2, GRASP3 y GRASP3 con Rencadenamiento de trayectorias en los ejemplos de prueba con *n* =500

		GRASP1	GRASP2	GRASP3	GRASP3+PR
Tipo I	Valor	6.6796	7.0163	7.71370405	7.78896586
	m	154.4	157.6	139.4	145.2
	# veces mejor	0	0	0	10
	Desviación	14.22%	9.91%	1.07%	0.00%
	Tiempo CPU (s)	984.56882	950.821574	717.337083	669.417148
Tipo II	Valor	8.898207	9.26865	10.2957	10.43732
	m	186.1	170.3	143.2	144.4
	# veces mejor	0	0	0	10
	Desviación	14.74%	11.18%	1.35%	0.00%
	Tiempo CPU (s)	736.79645	708.2459	662.422	679.641

Los resultados mostrados en la Tabla 21 están en total concordancia con los resultados obtenidos en los experimentos previos con los problemas de tamaño pequeño y mediano; es decir con $n = 15, 20, 25, 30, 35$ y 150. En particular se confirma que la heurística GRASP3 es más eficiente, pues consistentemente permite obtener mejores resultados, y en menor tiempo de ejecución, que GRASP1 y GRASP2. También, como se muestra en la última columna de la Tabla 21, la fase de Rencadenamiento de Trayectorias permitió mejorar los resultados de la heurística GRASP3 en todos los casos y para los dos tipos de ejemplos considerados.

En la Tabla A 9 y Tabla A 10 el ANEXO 1 se presentan los resultados detallados de la aplicación de estos cuatro algoritmos en cada uno de los problemas de prueba de tipo I y tipo II, respectivamente.

5.4.3. Pruebas estadísticas

De acuerdo a nuestra experimentación los algoritmos tienen un comportamiento similar al tratar tanto ejemplos de tipo I como ejemplos de tipo II.

La prueba de hipótesis de Friedman, aplicada a los resultados resumidos en la Tabla 21 exhibe un nivel de significancia p de 0.000, lo cual indica que hay diferencias significativas entre los cuatro métodos considerados. De acuerdo al Ranking proporcionado por este test, el mejor método es la heurística GRASP3+PR, con un rank value de 3.80), seguido por el método GRASP3, con un rank value de 3.20, GRASP2 , con un rank value de 2.00 y GRASP1 con un rank value de 1.00.

Considerando que GRASP3 y GRASP3+PR obtienen Rank values similares, también realizamos una comparación de los resultados obtenidos por ambos métodos con los dos test no paramétricos más conocidos para comparar datos apareados: el test de Wilcoxon y el test del Signo. Estos dos test permiten responder a la pregunta: Las dos muestras de datos apareados (en nuestro caso las soluciones obtenidas con GRASP3 y GRASP3+PR) ¿representan a dos poblaciones diferentes?. El nivel de significancia resultante p es de 0.019, lo cual permite concluir que sí hay evidencia estadística suficiente para concluir que los datos si provienen de poblaciones diferentes, y por tanto que los métodos GRASP3 y GRASP3+PR producen resultados distintos. De otro lado el test de los signos calcula el número de ejemplos en los cuales un algoritmo supera a otro. El nivel de significancia resultante p=0.012 indica que GRASP3+PR es claramente el ganador.

5.4.4. Perfil de búsqueda

Finalmente, para completar el análisis de la comparación de la eficiencia de los algoritmos que se diseñaron, se construyeron las gráficas del perfil de búsqueda de los algoritmos; es decir, cómo estas heurísticas fueron mejorando el valor de la función objetivo en función del tiempo de ejecución.

En la Figura 38 se representa el perfil de búsqueda de los cuatro métodos en instancias tipo I para una corrida de 800 segundos en todos los métodos, y en la Figura 39 se puede observar el detalle ampliado de este perfil de búsqueda en la vecindad de los mejores valores encontrados. Las figuras claramente muestran que GRASP3 logra rápidamente buenas soluciones, y que GRASP1 y GRASP2 requieren mucho tiempo para mejorar poco el valor de la función objetivo.

Por otro lado, en la ejecución de GRASP3+PR, la fase de rencadenamiento de trayectorias es ejecutada luego de que ha sido poblado el conjunto de soluciones élite, ES, y luego de realizar la fase de diversificación de ES, lo cual ocurre luego de aproximadamente 460 segundos, en promedio. Entonces inicia la fase de rencadenamiento de trayectorias propiamente dicha, al aplicar este procedimiento a cada pareja de soluciones del conjunto de soluciones élite, las gráficas que indican la evolución de la mejor solución encontrada muestran que esta fase permite obtener rápidamente mejores soluciones, superando a GRASP3 (sin PR), que a partir de cierto momento no logra mejoras en las soluciones en la misma proporción que GRASP3+PR, y por tanto se ve superado por este.

Para las instancias de tipo II se muestra este perfil de búsqueda en la Figura 40 y la Figura 41, al comparar la ejecución de GRASP3 y GRASP3+PR el patrón es similar al observado en las instancias tipo I. Esto demuestra que el Rencadenamiento de trayectorias constituye una excelente herramienta de post-procesamiento para el método GRASP.

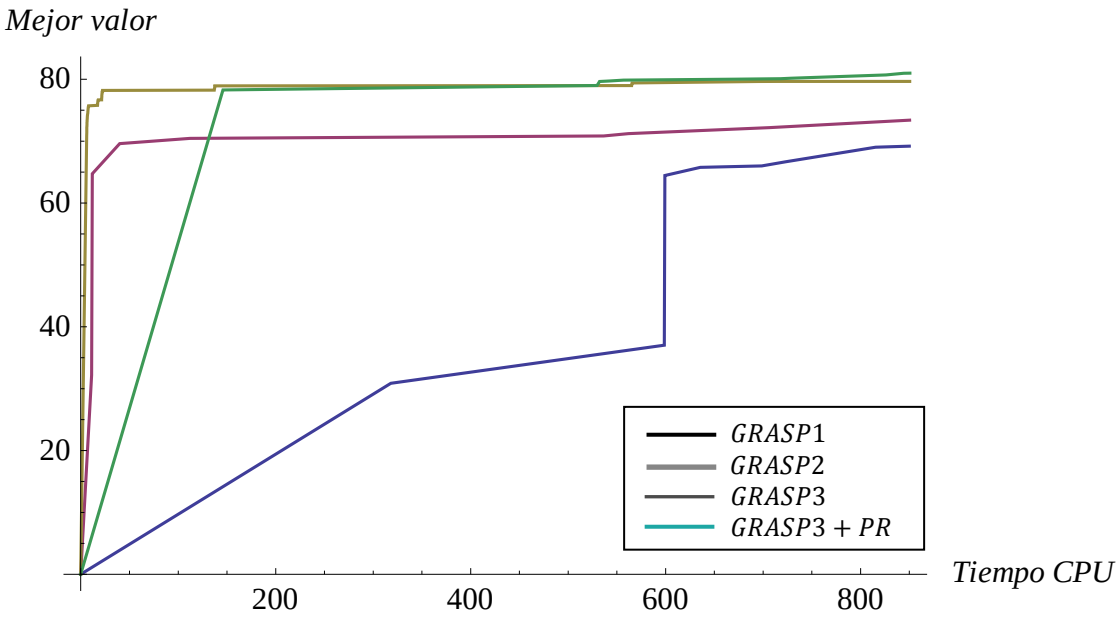


Figura 38. Perfil de búsqueda en instancias grandes de tipo I

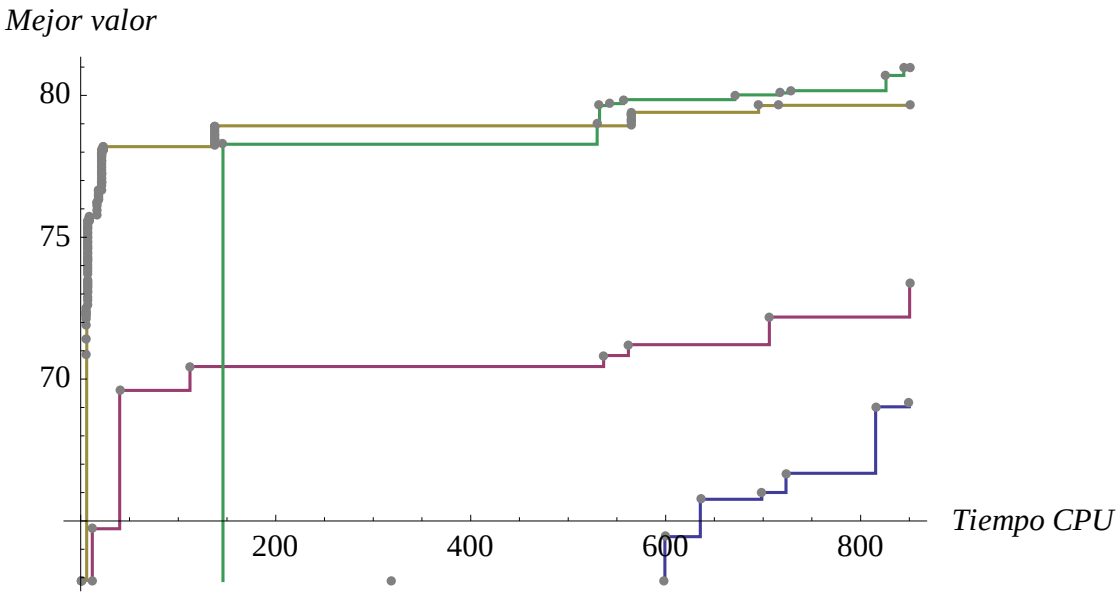


Figura 39. Detalle del perfil de búsqueda en instancias de tipo I

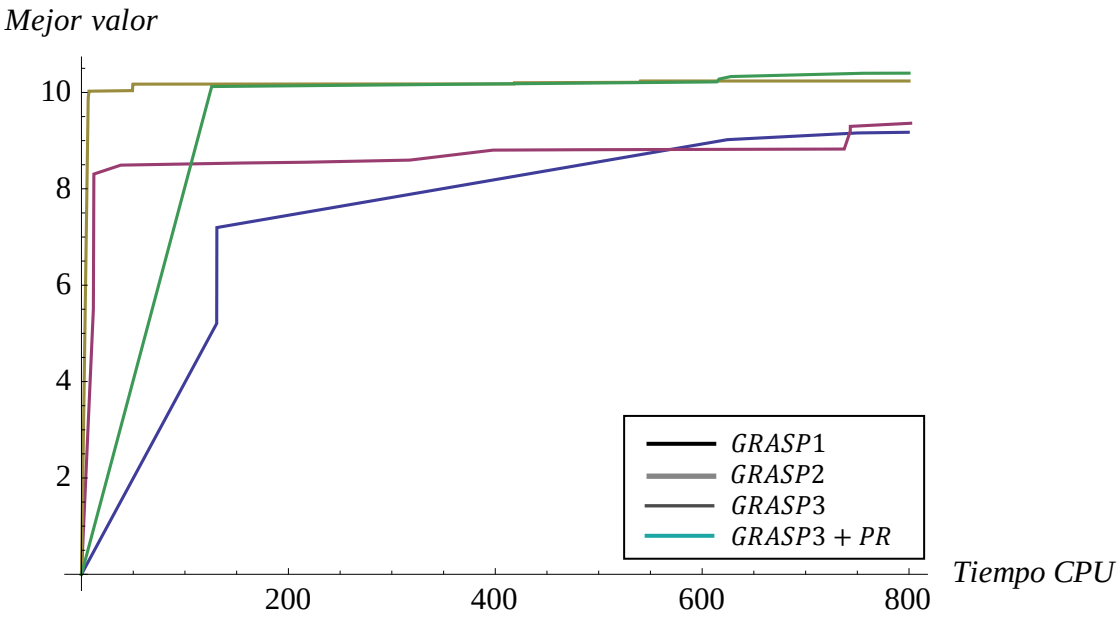


Figura 40. Perfil de búsqueda en instancias grandes de tipo II

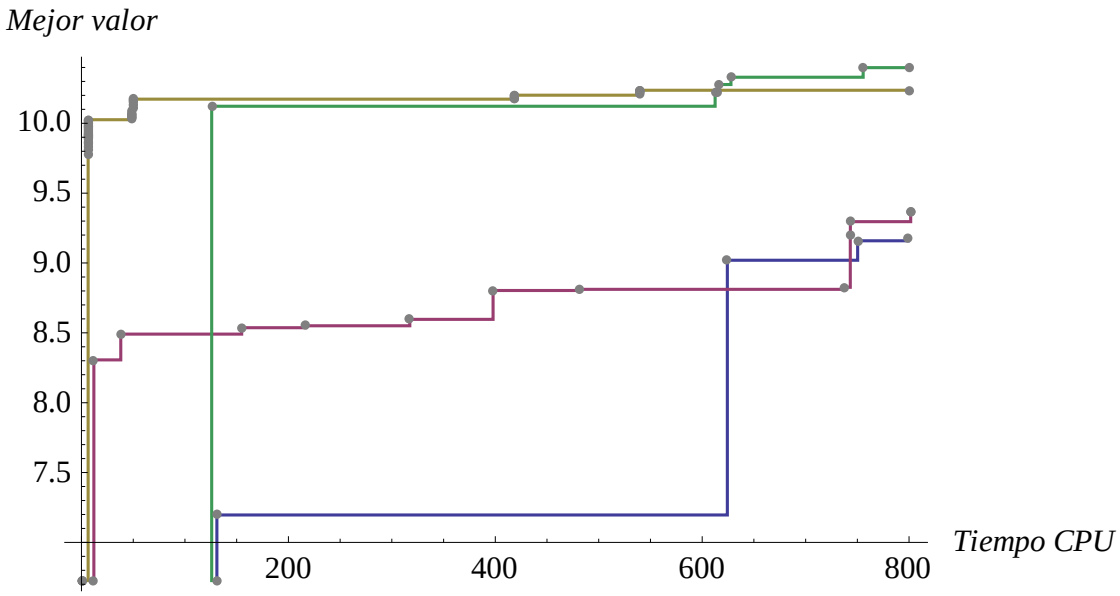


Figura 41. Detalle del perfil de búsqueda en instancias de tipo II

5.5. Comparación de la heurística GRASP3 con un Optimizador de caja negra

Cuando se resuelve un problema de optimización, una práctica común es desarrollar un método dependiente del contexto (es decir un procedimiento que explote explícitamente la estructura y propiedades del problema para conducir una búsqueda eficiente). Muchas implementaciones de heurísticas y metaheurísticas están basadas en este paradigma, es decir, desarrollan un método especializado para un problema específico. El propósito de algunas investigaciones, como proponen Gortázar y otros en (41), es lo contrario; es decir, crear procedimientos generales que permitan hallar soluciones de calidad razonable para una amplia clase de problemas de optimización basados en un paradigma independiente del contexto, y es bajo esta filosofía que se han desarrollado programas de software comercial relevantes que se han popularizado en los últimos años en esta área.

De esta manera, en los últimos años los optimizadores comerciales de propósito general se han vuelto muy populares para resolver problemas de Investigación de Operaciones, y han incursionado con relativo éxito como sistemas de ayuda para la toma de decisiones de las empresas. Muchas veces en la literatura se refiere a estos programas computacionales como "Optimizadores de caja negra" o "Procedimientos independientes del contexto" como se definen en (42). La mayoría de estos programas no explotan la estructura específica de la función objetivo del problema que se está resolviendo, sino que funcionan a manera de una caja negra en cuanto a que lo importante en ellos es poner énfasis en tener bien definidas las entradas que recibe el optimizador y saber interpretar las salidas o respuestas que produce, sin tener en cuenta en mayor medida su funcionamiento interno. Dicho de otro modo, en estos optimizadores de caja negra nos interesará describir cómo está modelado el problema y entenderemos qué es lo que hace el optimizador, pero sin dar mucha importancia a cómo lo hace. No se precisa definir ni conocer los detalles internos de su funcionamiento. Salvo ciertos aspectos que están disponibles para calibración por parte del usuario, el optimizador de caja negra se encarga de obtener una solución.

En efecto, los solvers de caja negra basados en metaheurísticas han hallado cabida en algunas implementaciones comerciales. Tres programas de software llevan la batuta, basados en metaheurísticas, y resuelven de manera general problemas de optimización:

- Solver Premium de Frontline Systems Inc. (www.frontsys.com)
- OptQuest de Opttek Systems Inc. (www.opttek.com)
- Evolver de Palisade Corporation (www.palisade.com)

Uno de los optimizadores de caja negra más populares actualmente es Evolver, de Palisade Corp., según sus fabricantes, en (43), el programa Evolver utiliza la tecnología de los algoritmos genéticos para resolver rápidamente problemas de finanzas, distribución, programación, asignación de recursos, manufactura, presupuesto, ingeniería y más, además se menciona que los algoritmos genéticos de Evolver llegan a la mejor solución "global" de un problema lineal, combinatorio o no lineal.

Entre los aspectos destacables de Evolver están: tener cierto grado de control sobre la forma en que se guía al proceso de optimización, poder definir parámetros de optimización, definición de tiempo de ejecución, elegir el tipo de operadores de cruce y mutación. Evolver incluye métodos de solución distintos para diferentes tipos de problemas, estos métodos son: "Recipe", "Order", "Budget", "Grouping", "Scheduling", o "Project", lo cual es muy ventajoso en el proceso de optimización porque estos métodos le dan al proceso de solución, información adicional que da lugar a mejores resultados, de esta manera el proceso de solución al final sí depende un poco del contexto, y no es totalmente independiente. Como en nuestro problema lo que requerimos es agrupar los n objetos iniciales en dos grupos: uno de m elementos escogidos y otro de $m-n$ elementos no escogidos, el método que se usó para resolver el problema Max-Mean es el de "Grouping". El criterio de parada para la ejecución del software fue elegido como un tiempo de ejecución pre establecido.

En esta parte comparamos nuestra heurística GRASP3 con el optimizador Evolver, en la Tabla 22 se muestran los resultados promedio alcanzados en instancias tipo I y tipo II para problemas pequeños, medianos y grandes mientras que en la Tabla A 11, Tabla A 12 y la Tabla A 13 del ANEXO 1 se pueden observar los resultados detallados para cada una de las instancias. En todos los casos se otorgó mucho más tiempo de ejecución al optimizador Evolver como una ventaja frente a nuestra heurística GRASP, así por ejemplo para los problemas pequeños de tipo I y II $(n = 30)$, nuestra heurística GRASP utilizó un CPU promedio de 0.414 y 0.363 segundos, respectivamente, entonces establecimos para la ejecución de Evolver un tiempo de 40 segundos, esto es ¡aproximadamente cien veces más! En cambio para el tratamiento de las instancias de

tamaño mediano $(n = 150)$ se utilizó en Evolver un tiempo de 200 segundos, aproximadamente diez veces más que el que utilizamos en nuestra heurística GRASP3, y para las instancias grandes $(n = 500)$ se utilizó en Evolver un tiempo de 3500 segundos, es decir aproximadamente cinco veces más que el que utilizamos en nuestra heurística GRASP3. A pesar de esta ventaja en el tiempo de cómputo otorgada al optimizador Evolver, los resultados de nuestra heurística GRASP3 no se pudieron superar en ningún caso, es más se observa en la Tabla 22 que existe un gran deterioro de la solución hallada por Evolver conforme se utilizan problemas de talla más grande. También se puede observar que para Evolver resultan más difíciles de resolver los problemas de tipo II que los de tipo I, y que para $(n = 500)$ la desviación respecto a las soluciones obtenidas con GRASP3 ya superan el 50%.

Tabla 22. Comparación de la heurística GRASP3 con el optimizador EVOLVER en el problema Max-Mean

		n = 30		n = 150		n = 500	
		EVOLVER	**GRASP**	**EVOLVER**	**GRASP**	**EVOLVER**	**GRASP**
Tipo I	Valor	1.85473	1.87495	3.15248	4.3236	5.21441	7.71370
	m	11.10	10.80	64.90	44.00	217.30	139.40
	# veces mejor	4	10	0	10	0	10
	Desviación	1.05%	0.00%	27.10%	0.00%	32.41%	0.00%
	Tiempo CPU (s)	40	0.414	200	27.061	3500	717.336
Tipo II	Valor	2.33221	2.38278	4.06619	5.684	4.90423	10.2956713
	m	11.40	10.80	65.30	46.10	243.50	143.20
	# veces mejor	4	10	0	10	0	10
	Desviación	2.10%	0.00%	28.46%	0.00%	52.38%	0.00%
	Tiempo CPU (s)	40	0.363	200	22.874	3500	662.422

En la Figura 42 se puede observar un gráfico de barras con el incremento de la desviación porcentual de las soluciones obtenidas con Evolver respecto a las soluciones reportadas con GRASP3.

En la Figura 43 en cambio, a modo de ejemplo, se observa el perfil de búsqueda del algoritmo genético utilizado por Evolver, en un problema de prueba mediano, se nota que al inicio el software encuentra muchas soluciones nuevas, cuyo valor objetivo crece rápidamente, pero a partir de cierta iteración cada vez le cuesta más al optimizador mejorar la solución, estancándose en un óptimo local e impidiendo que mejore a pesar de otorgarle mucho tiempo de cómputo.

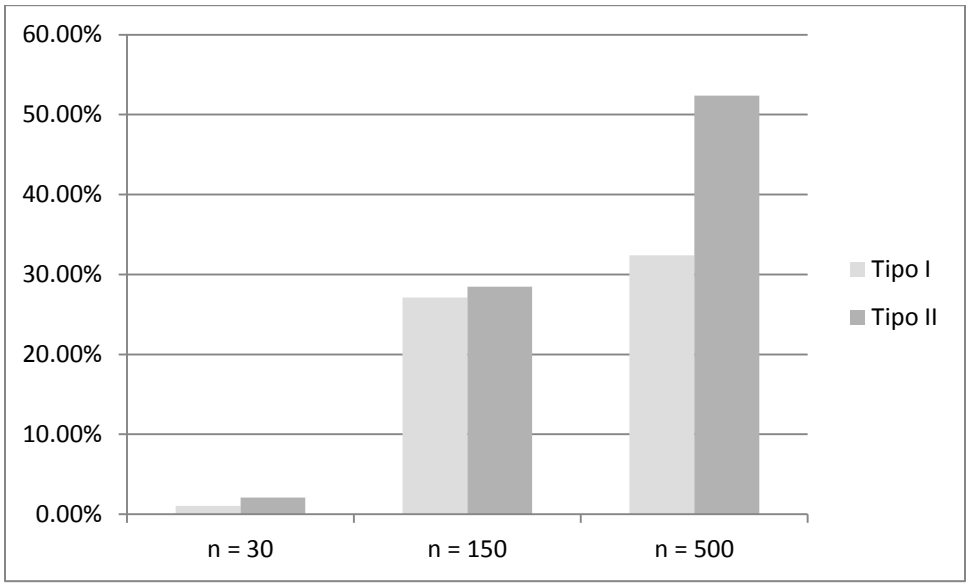


Figura 42. Desviación de los resultados encontrados con Evolver respecto a la solución encontrada con GRASP3

A partir de estos resultados descartamos el uso de este tipo de optimizadores, independientes del contexto, como mecanismos de solución eficientes para el problema Max-Mean.

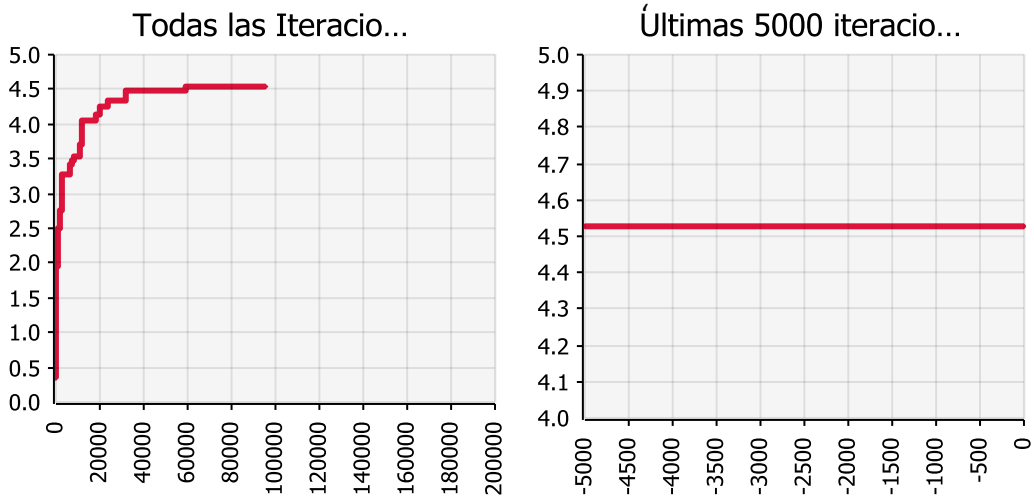


Figura 43. Perfil de búsqueda del algoritmo genético ejecutado por Evolver en problemas de tamaño mediano *n* = 150

5.6. Resumen de los métodos desarrollados

El problema de maximización Max-Mean es un problema computacionalmente difícil que se plantea en el contexto de los problemas de la dispersión equitativa y de la diversidad máxima.

Este problema nos ha servido como un buen caso de prueba para desarrollar las nuevas estrategias de búsqueda que proponemos para resolver este problema combinatorio. En particular hemos desarrollado un algoritmo GRASP constructivo con base en una combinación no estándar de codicia y aleatoriedad, una estrategia de búsqueda local basada en la metodología de vecindades descendentes variables, la cual incluye tres tipos diferentes de vecindades, y una fase de post procesamiento basada en rencadenamiento de trayectorias. Este último método está basado en una medida de control de la diversidad en el proceso de búsqueda.

Se realizaron extensos experimentos computacionales para estudiar primero el efecto de los cambios en los elementos críticos de la búsqueda y luego comparar la eficiencia de las heurísticas que proponemos con otros procedimientos adaptados de otros autores para problemas similares. La comparación con los métodos previos también basados en la metaheurística GRASP favorece nuestra propuesta.

También se modeló el problema Max-Mean en Evolver, que es software comercial que usa algoritmos genéticos y que es muy referenciado. A pesar de favorecerlo otorgándole un tiempo de ejecución excesivamente grande los resultados demostraron que el problema es muy complicado para este software, por lo que la utilización de optimizadores de caja negra no es recomendable en la solución del problema Max-Mean, dejando evidente que nuestra propuesta es muy superior para resolverlo.

Capítulo 6.

Un Caso de aplicación para el problema Max-Mean

6.1. Los grupos de personas más diversos son más eficientes

Es un hecho evidente, a partir de nuestra vivencia diaria, que los seres humanos a menudo cooperamos en todo tipo de situaciones, desde cuestiones de la familia a problemas de carácter mundial como la protección de los bosques, el agua y la lucha por preservar el medio ambiente. Sin embargo, la teoría de juegos evolutiva predice que la tentación egoísta a renunciar al bien público es mayor que el deseo de la cooperación colectiva, lo cual está avalado por múltiples experimentos económicos, entonces ¿cómo es que aparece la cooperación? Pues según estudios recientes, como en (1), (2) y (13), es la diversidad social la que proporciona un escape a esta paradoja, y si en el pasado, mucho del interés público en la diversidad social estuvo enfocado desde el punto de vista de la igualdad de oportunidades, la justicia, la equidad y la representación, recientemente se ha tratado de encontrar cómo sacar algún tipo de utilidad o ventaja de la diversidad.

Así pues, en la actividad cotidiana de las organizaciones, empresas, escuelas, equipos deportivos, etc. se han observado evidencias de que la diversidad juega un papel importante en la habilidad que tienen los grupos de personas para resolver problemas. Últimamente han aparecido en la literatura investigaciones que demuestran formalmente que este fenómeno empírico es cierto, proporcionando una justificación teórica para este hecho, más específicamente en (2). Una consecuencia de esto es que, bajo ciertas circunstancias, los grupos de personas que están conformados de manera diversa pueden superar en productividad a los grupos conformados por las personas individualmente más capacitadas para resolver estos problemas; es decir, en cierto sentido la diversidad triunfa sobre la habilidad, y el lema de Steve Jobs, ex CEO de Apple, "*piensa diferente*" puede ser una frase clave si se quiere aumentar la productividad de equipos de trabajo. O como plantea Carol K. Goman (presidenta de Kinsey Consulting Services): "La diversidad es, a menudo, más importante que la experiencia. En otras palabras cuantas más aportaciones de colaboración se consigan, sobre todo si son contradictorias, más creativas son las soluciones".

Desde un punto de vista práctico, este resultado implica que, por ejemplo, una empresa que quiere conformar un equipo de trabajo no deberá buscar seleccionar simplemente a los individuos más altamente calificados para ello, probablemente lo más eficiente será escoger un grupo diverso. En realidad lo ideal sería que los grupos de trabajo estén conformados por personas con alta habilidad y diversos; sin embargo, estos dos objetivos suelen estar contrapuestos pues la diversidad del equipo formado por las personas más hábiles tiende a hacerse pequeña, como se demuestra en (44)

Por ejemplo, consideremos que estamos tratando de seleccionar grupos de personas para formar un equipo de trabajo. Si la eficiencia con la que grupos de personas resuelven problemas es un resultado de la diversidad del grupo, plantear objetivos basados sólo en la eficiencia individual, puede resultar en una gran inequidad del sistema resultante, entonces, si estamos conscientes de que también, además de eficiencia, se debe perseguir la equidad, podemos considerar los modelos de diversidad que sí consideran de alguna manera el concepto de la equidad. Así, en el problema Max-Mean no estamos interesados en maximizar la eficiencia total del grupo de personas, sino en maximizar la eficiencia promedio de las personas que conforman el grupo.

La idea de trasfondo es que si tenemos una población de personas capacitadas para realizar alguna tarea, estas personas tienen diferentes grados de habilidad o de productividad para resolverla, y si tenemos que seleccionar un equipo de trabajo de esa población para realizar la tarea, podemos considerar dos grupos posibles: en el primero escogemos sólo los individuos más altamente calificados, y en el segundo escogemos individuos “diversos” en algún sentido. Resulta que los primeros terminarían de alguna manera llegando a la misma solución, dificultando y entorpeciendo el trabajo unos a otros, en cambio en el segundo grupo la diversidad crearía más perspectivas y por tanto más oportunidad de salir de un atascamiento en la solución de los problemas, generándose de alguna manera el ambiente propicio para aumentar la productividad individual de cada uno, y por tanto de todo el grupo. Desde un punto de vista formal lo que sucede es que en el primer grupo, bajo ciertas hipótesis, las personas más altamente calificadas tienden a convertirse en similares en cuanto a sus puntos de vista y formas de resolver un problema por lo cual el conjunto de óptimos locales a los que puede llegar el grupo se ve reducido. En cambo en el segundo grupo la diversidad de sus miembros origina un conjunto de óptimos locales más amplio, y por tanto hay más oportunidad para mejorar.

6.2. Diversidad en identidad y diversidad funcional, perspectivas y heurísticas

En términos de una población de personas, entendemos por “diversidad en identidad”, o simplemente “diversidad”, a las diferencias en sus características demográficas, culturales, étnicas, formación académica y experiencia laboral. En cambio denominamos “diversidad funcional” a las diferencias en cómo estas personas enfocan y tratan de resolver un problema. Un hecho importante es que estos dos tipos de diversidad están correlacionados, pues se ha identificado experimentalmente una fuerte correlación entre estos dos tipos de diversidad, tal como se demuestra en (44). Dada esta conexión, se puede deducir que grupos diversos en identidad son diversos funcionalmente.

En la literatura, el enfoque que emplea una persona para resolver un problema es una representación o una codificación del problema en su lenguaje interno, y se lo puede denominar “perspectiva”. Así, formalmente, una perspectiva es un mapeo P del conjunto de soluciones de un problema al lenguaje interno de la persona que va a resolver un problema.

Por otro lado la forma como las personas intentan resolver un problema, o cómo éstas buscan las soluciones se denomina “heurística”. Más formalmente una heurística es un mapeo H desde la codificación de las soluciones en el lenguaje interno de la persona que va a resolver un problema al conjunto de soluciones. Así, dada una solución particular, el subconjunto generado por el mapeo H es el conjunto de las otras soluciones que la persona considera. De esta manera, la habilidad para resolver un problema por parte de una persona está representada por su pareja perspectiva-heurística (P, H). Dos personas podrían diferir en una de estas componentes o en ambas; es decir, podrían tener perspectivas diferentes o heurísticas diferentes, o diferir en las dos. Una solución será un óptimo local para una persona si y sólo si cuando la persona codifica el problema y aplica su heurística, ninguna de las otras soluciones que la persona considera tiene un valor mayor. Por esto, cuando el grupo no es diverso, las personas llegan a ser similares en cuanto a sus habilidades, y así tenderá a tener pocos óptimos locales, ocasionando que el grupo se atasque en una de esas soluciones.

6.3. Cómo seleccionar el equipo de trabajo más productivo

Desde un punto de vista intuitivo, la conclusión de que grupos diversos en identidad pueden superar a grupos no diversos (homogéneos) debido a su gran diversidad funcional se basa en la afirmación, bien aceptada, de que si los agentes dentro de los grupos tienen igual habilidad individual para resolver un problema, un grupo funcionalmente diverso superará a un grupo homogéneo. En (2) se demuestra que grupos con diversidad funcional tienden a superar a los mejores agentes individuales con tal de que los agentes en el grupo tengan igual habilidad. Esto todavía dejaba abierta una importante pregunta: ¿Puede un grupo funcionalmente diverso, cuyos integrantes tienen menor habilidad individual, tener un rendimiento superior al grupo de personas que tienen la más alta habilidad individual?, en (21) finalmente se resolvió de manera afirmativa esta pregunta, proporcionando una demostración matemática a este hecho. Pero aún surgen de manera natural otras inquietudes al respecto: ¿Cuántos integrantes debería tener este grupo de tal manera que la diversidad promedio dentro del grupo sea máxima?, y, ¿se puede detectar cuál es este grupo funcionalmente más diverso?

Así, por ejemplo, consideremos la situación en la cual una Institución desea contratar personas para resolver un problema. Para realizar una buena selección la Institución usualmente tomaría una prueba a los aplicantes, digamos 500, para estimar sus habilidades individuales para resolver el problema. Y supongamos además que todos los aplicantes son individualmente capaces para resolverlo, pues tienen la formación y la experiencia necesarias, pero tienen diferentes niveles de habilidad. Nos preguntamos si la Institución deberá contratar:

(i). La persona con el mayor puntaje obtenido en la prueba;

(ii). Las 10 personas con los puntajes más altos;

(iii). 10 personas seleccionadas aleatoriamente desde el grupo de aplicantes;

(iv). Las 10 personas más diversas en identidad del grupo de aplicantes;

(v). El grupo de personas más diverso en promedio del grupo de aplicantes (en este caso el número a seleccionar no es asumido como conocido).

Ignorando los posibles problemas de comunicación dentro de los grupos, la literatura existente sugiere que (ii) es mejor que (i), ver (44), ya que más personas buscarán en un espacio más amplio, teniendo entonces más oportunidades para obtener mejores soluciones, en lugar de

la acción de la persona mejor puntuada que se quedará atascada en uno de sus óptimos locales. Recientemente en (14) se ha demostrado formalmente que (iii) es mejor que (ii).

De esta manera, falla la intuición basada en que el grupo de las personas con puntajes más altos, es decir los más capacitados individualmente, vayan a formar el mejor equipo de trabajo, y por tanto que la empresa debe contratar (ii), pues se demuestra que bajo ciertas hipótesis (iii) es mejor decisión, como se ve en (14). Los autores llegan a determinar que un equipo de personas escogidas aleatoriamente tiene más diversidad funcional y que bajo ciertas condiciones supera en rendimiento a (ii), ya que bajo el conjunto de condiciones que identifican los autores, la diversidad funcional del grupo de las personas que están individualmente mejor capacitadas para resolver el problema necesariamente se vuelve muy pequeña, por lo que, al final, la ventaja de tener las mejores habilidades individuales se ve más que compensada por la mayor diversidad del grupo escogido aleatoriamente. Nótese que los autores en su demostración ni siquiera usan al equipo con la máxima diversidad, sino a un grupo seleccionado aleatoriamente, y aun así logran demostrar que es mejor, gracias a la mayor diversidad inherente al grupo aleatorio frente al grupo de los más habilidosos individualmente.

Aquí se demuestra el Corolario del teorema 2, que indica que si escogiéramos al grupo más diverso en promedio, es decir contratar al grupo conformado según (iv), éste resultaría más productivo que contratar aquel formado aleatoriamente (iii), y, por transitividad, mejor que el grupo conformado por los mejores puntuados (ii) y por último mejor que simplemente escoger el más puntuado (i).

Por último, la literatura dice poco o nada sobre (v), pues clásicamente en los problemas de diversidad se ha considerado al número de elementos a escoger como un valor dado, sin embargo en las aplicaciones prácticas no está claro como escoger el número de elementos a seleccionar, y lo mejor podría ser dejar que sea el propio proceso de optimización el que nos proporcione su valor. Así, el foco de nuestro análisis se centra en la disputa entre la importancia que tienen las habilidades individuales de cada persona en el grupo, su diversidad funcional (atrapada por la diversidad en identidad), y el tamaño ideal del grupo, tal como se representa esquemáticamente en la Figura 44.

Una conclusión de todo esto, es que la diversidad en las organizaciones debe ser alentada, lo cual implica nuevas políticas, formas organizacionales y estilos de administración. En el contexto de resolver un problema, el valor de una persona depende de su habilidad para mejorar la decisión colectiva, pues la contribución de esta persona depende en gran medida de las perspectivas y heurísticas de las otras personas que conforman el equipo de trabajo. La diversidad en el enfoque de la solución a un problema respecto a las otras personas es un importante predictor de su valor, y a la larga puede ser más relevante que su habilidad individual para resolver el problema por su cuenta. Así, para estimar la contribución potencial de una persona en un equipo de trabajo, es más importante hacer énfasis en medir cómo esta persona piensa diferente, antes que en estimar la magnitud de la habilidad de la persona a través de pruebas de aptitud o test de inteligencia.

Sin embargo, hay que estar conscientes de algunos aspectos que no han sido considerados y que podrían tener influencia en el rendimiento de un equipo de personas. Un aspecto importante que hay que mencionar es que los grupos diversos en identidad a menudo podrían tener más conflictos, más problemas de comunicación, menos respeto mutuo y menos confianza entre sus miembros que los grupos homogéneos, lo que podría ocasionar en los grupos diversos una disminución en su rendimiento.

En (14) se menciona que las personas con perspectivas similares pero con heurísticas diversas pueden comunicarse unos con otros sin mayor problema, pero personas con perspectivas diversas pueden tener problemas para comprender soluciones identificadas por los otros integrantes del grupo, en ese sentido lo mejor para las organizaciones será contratar personas con perspectivas similares pero garantizando una diversidad de heurísticas, de esta manera, las organizaciones pueden explotar de mejor manera los beneficios de la diversidad mientras minimiza los costos de la falta de comunicación.

6.4. Hipótesis básicas y relación entre habilidad y diversidad

En esta sección se enuncia el teorema 1, demostrado en (14), que explica la lógica detrás del hecho de que un equipo de personas escogidas al azar, de un banco de postulantes que están capacitados para resolver un problema, es mejor que el equipo formado por las personas más capaces individualmente, de ahí establecemos el resultado, que es inmediato, que el equipo de

personas con más diversidad supera al equipo formado por las personas más habilidosas para resolver el problema.

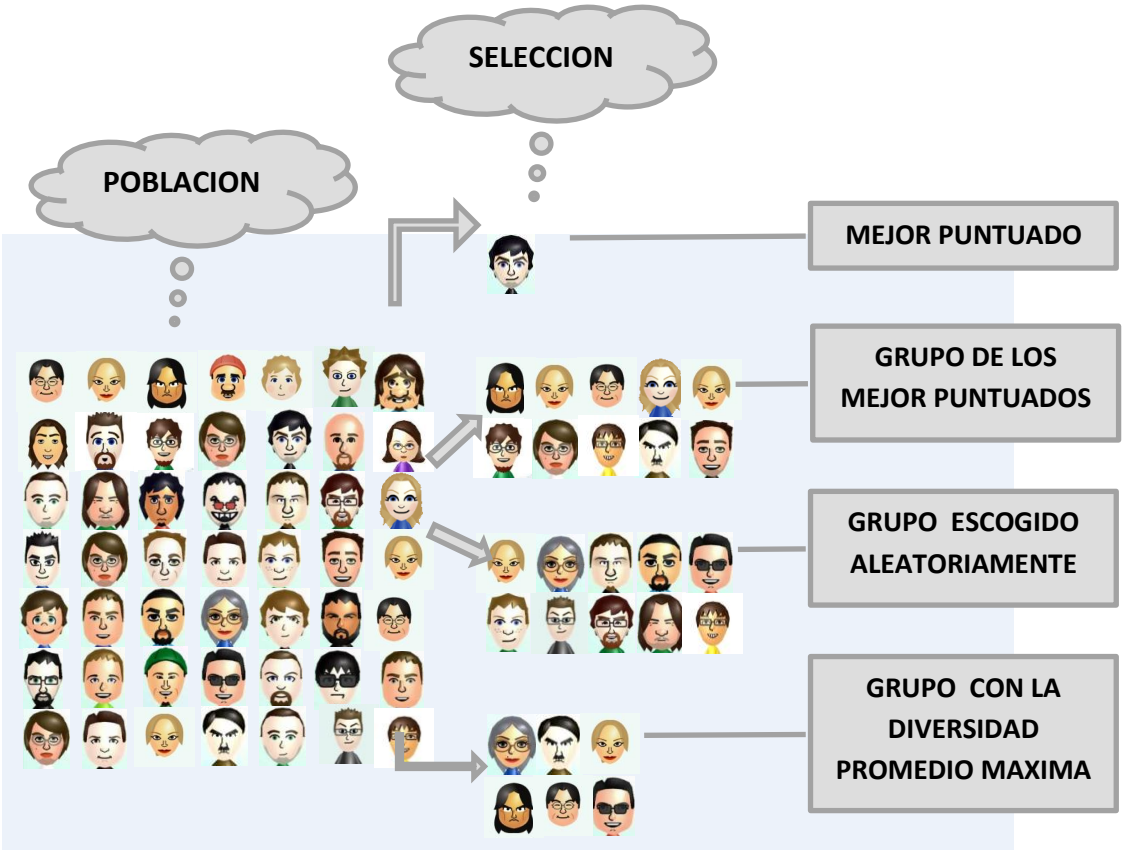


Figura 44. ¿Cuál es la mejor selección de un equipo de trabajo a partir de una población de personas capacitadas para resolver un problema pero con distintos grados de habilidad?

Para establecer el resultado teórico, consideramos que la población de donde se escogerá el equipo, es decir los postulantes, representada con Φ satisface las siguientes suposiciones:

- Los postulantes están entrenados para resolver el problema. Dada una solución inicial, los postulantes pueden hallar una solución mejor, aunque sea sólo un poco mejor;
- El problema es difícil, ningún postulante puede hallar siempre la solución óptima;
- Los postulantes son diversos, y por tanto para cualquier solución potencial que no es la óptima, existe al menos un postulante que puede hallar una mejora;
- El mejor postulante es único.

Si consideramos un equipo de postulantes escogidos aleatoriamente desde Φ de acuerdo a alguna distribución, el teorema establece que, con probabilidad 1, existen tamaños muestrales N_1 y N, $N_1 < N$, tal que el rendimiento colectivo del equipo de los N_1 postulantes escogidos aleatoriamente supera al rendimiento colectivo de los N_1 mejores postulantes en el grupo de N postulantes.

Para formular el teorema 1 más precisamente, representamos con X al conjunto de soluciones posibles del problema, una función que da el valor de cada solución $V: X \to [0,1]$, suponemos además que V tiene un único máximo x^*, y que $V(x^*) = 1$. Cada postulante ϕ parte de una solución inicial x y usa una regla de búsqueda para encontrar el máximo, pero no siempre lo encuentra, sino que generalmente se atasca en un óptimo local, sea $\phi(x)$el óptimo local que logra el postulante ϕ si inicia su búsqueda en x. De esta manera $\phi(X)$ representará el conjunto de óptimos locales para el postulante ϕ, luego suponemos que toda la habilidad del postulante para resolver un problema está capturada por ϕ. Se supone además que los postulantes seleccionan aleatoriamente de acuerdo a una distribución de probabilidades ν el punto inicial desde donde empieza la búsqueda.

Por su capacidad para resolver un problema, cada postulante está caracterizado por la pareja (ϕ, ν), y estimamos su rendimiento a través del valor esperado de su búsqueda al tratar de resolver el problema, representado por $E(V; \phi, \nu)$; es decir,

$$E(V; \phi, \nu) = \sum_{x \in X} V\big(\phi(x)\big)\, \nu(x)$$

Las hipótesis que deben satisfacer los postulantes ϕ, bajo las cuales se demuestra el teorema son las siguientes:

HIPÓTESIS 1 (Consistencia):

i. $\forall x \in X: V\big(\phi(x)\big) \geq V(x)$

ii. $\forall x \in X: \phi\big(\phi(x)\big) = \phi(x)$

HIPÓTESIS 2 (Dificultad):

$$\forall \phi \in \Phi, \exists x \in X: \phi(x) \neq x$$

HIPÓTESIS 3 (Diversidad):

$$\forall x \in X \setminus \{x^*\}, \exists \phi \in \Phi: \phi(x) \neq x$$

HIPÓTESIS 4 (Unicidad):

$$\arg\max\{E(V; \phi, v): \phi \in \Phi\} \text{ es único}$$

La hipótesis 1 indica que dada una solución inicial las personas siempre tratarán de encontrar soluciones mejores, pero nunca escogerán una solución peor, y que se atascarán en un óptimo local. La hipótesis 2 implica que ninguna persona, individualmente, puede alcanzar el óptimo siempre partiendo de cualquier punto. En la hipótesis 3 se establece de manera simple la esencia de la diversidad, cuando una persona se atasca en un óptimo local siempre existe otra persona que puede hallar una mejora debido a un enfoque diferente. Por último, la hipótesis 4 establece que dentro del conjunto de postulantes considerado existe un mejor desempeño único. Nótese que no se hacen suposiciones específicas acerca de cómo el equipo de personas trabajan juntas, excepto que la búsqueda del grupo sólo puede quedar atascada en un punto que es óptimo local para todos las personas del grupo. Con estas hipótesis, en (21) se demuestra el teorema 1 que se enuncia a continuación.

TEOREMA 1: Sea Φ un conjunto de personas que satisfacen las hipótesis 1–4. Y sea μ una distribución de probabilidades sobre Φ. Entonces, con probabilidad 1, existirán enteros positivos N y N_1, $N > N_1$, tal que el rendimiento conjunto de N_1 personas escogidas aleatoriamente supera al rendimiento conjunto de las N_1 personas individualmente más capaces, tomadas del grupo de N personas independientemente escogidas de acuerdo a μ.

La demostración del teorema 1, que se presenta en (14) se basa en dos ideas principales: Primero se demuestra que el grupo aleatorio de personas encontrará, con probabilidad 1, la solución óptima cuando el tamaño del grupo se hace suficientemente grande, gracias a que la hipótesis 3 garantizará suficiente diversidad en el grupo para salvar los atascamientos en óptimos locales de los integrantes individuales del grupo. La segunda idea se basa en la hipótesis 4 de unicidad, que indica que con probabilidad 1 cuando el número de personas es grande, las personas más capacitadas llegan a convertirse en similares y no son mejores que el más capacitado de ellos, quien por la hipótesis 2 no siempre puede hallar la solución óptima.

Si bien los autores de (14) llegan a demostrar que un grupo seleccionado aleatoriamente funciona mejor que el grupo formado por los mejores, es inmediato extender el resultado como se presenta en el siguiente corolario, que se demuestra aquí, en el cuál se establece más directamente la relación entre diversidad y habilidad. En el corolario $div(M)$ representa la diversidad del conjunto M evaluada con cualquiera de las medidas consideradas en el Capítulo 1.

COROLARIO: Si Φ es un conjunto de personas que satisfacen las hipótesis del teorema 1, entonces, con probabilidad 1, existirán enteros positivos N y N_1, $N > N_1$, tal que el rendimiento conjunto del grupo de N_1 personas que maximiza $\{div(M), M \subset \Phi, |M| = N_1\}$ supera al rendimiento conjunto de las N_1 personas individualmente más capaces, tomadas del grupo de N personas independientemente escogidas de acuerdo a μ.

Demostración:

La demostración del corolario es inmediata, pues la prueba del teorema 1 se basa en que la diversidad del conjunto de personas seleccionado aleatoriamente es más diverso que el conjunto de personas con la más alta habilidad. Así pues, si seleccionamos el grupo de personas más diverso, este superará el rendimiento del grupo de personas seleccionadas aleatoriamente, a causa de la mayor diversidad del primero, y por el teorema 1, este último grupo supera en rendimiento al grupo formado por las personas más capacitadas individualmente. Se sigue por transitividad el resultado mostrado en este corolario.

■

La idea del corolario es bastante intuitiva, ya que si escogemos las personas con más diversidad, es muy improbable que tengan óptimos locales comunes, y por tanto hay menos oportunidad para que el grupo se atasque en uno de ellos rápidamente. La propiedad asintótica del teorema, y luego del corolario, indica que cuando el número de personas crece, la probabilidad de tener óptimos locales comunes converge a cero. De esta manera se puede reivindicar que las empresas deben poner más atención en establecer la forma de medir cómo la persona piensa de manera diferente, antes que tomar exhaustivas pruebas de aptitud o de inteligencia a las personas que conocemos están individualmente capacitadas para formar parte del equipo de trabajo, en ese sentido el lema de Steve Jobs, ex CEO de Apple “think different” (piensa diferente), en este contexto, adquiere más vigencia que nunca.

6.5. Resolución de un caso

Para aplicar los algoritmos generados se escogió la población de profesores de la Escuela Superior Politécnica del Litoral, de Guayaquil – Ecuador, población consistente en 586 individuos con el fin de seleccionar un grupo representativo. Cabe señalar que actualmente en el Ecuador se expidió una nueva Ley de Educación Superior que regula las actividades de las Universidades en el país, con grandes cambios respecto a la ley anterior. Una de las primeras consecuencias de esto es que las Universidades deberán adaptar sus Estatutos para que estén en concordancia con la nueva ley hasta julio de 2012. Bajo esta situación es necesario seleccionar un grupo representativo de profesores que participe en la elaboración del nuevo estatuto, en esta investigación planteamos que este problema de selección es un problema del tipo Max-Mean, y por tanto pueden ser aplicados los algoritmos diseñados para resolverlo.

Las variables consideradas fueron:

- PROFESOR: Identificador, valores de 1, 2, …, 586
- CONDICION LABORAL: -1 CONTRATADO, 1 TITULAR
- SEXO: -1 MUJER, 1 HOMBRE
- MAXIMO NIVEL ACADEMICO: -1 Tecnologo, -0.5 Grado, 0.5 Maestría, 1 Doctorado
- ESTUDIO EN LA ESPOL: -1 SI, 1 NO
- TIPO DE PROFESOR: -1 ACADEMICO, 0 ACADEMICO – INVESTIGADOR, 1 INVESTIGADOR
- UNIDAD A LA QUE PERTENECE: -0.5 INSTITUTO, 0.5 FACULTAD
- NIVEL SALARIAL: -1, -0.9, -0.8, …, 0.9, 1 (desde el más bajo al más alto)

En (14) los autores plantean el problema de cómo la diversidad en un grupo puede aumentar la eficiencia para resolver problemas, en particular usan experimentos en los cuales plantean a diferentes personas la forma como estas podrán resolver un mismo problema. La medida de diversidad (que vendrá a ser nuestro d_{ij}) es para nuestro caso:

$$d_{ij} = \frac{\sum_{l=1}^{k} \delta\left(\phi_l^i, \phi_l^j\right)}{k}$$

Donde $\delta(\phi_l^i, \phi_l^j) = \begin{cases} -1 & si\ \phi_l^i = \phi_l^j \\ |\phi_l^i - \phi_l^j| & si\ \phi_l^i \neq \phi_l^j \end{cases}$

Es obvio que esta medida toma valores negativos (en el caso de similitud) y positivos (en el caso de disimilitud)

Como ejemplo observemos los siguientes casos:

PROFESOR (*i*)	CONDICION LABORAL	SEXO	MAXIMO NIVEL ACADEMICO	ESTUDIO EN LA ESPOL	TIPO DE PROFESOR	UNIDAD A LA QUE PERTENECE	NIVEL SALARIAL
1	1	-1	0.5	-1	-1	0.5	0.3
2	1	-1	1	-1	1	0.5	-0.6

$$d_{12} = \frac{-1 - 1 + .5 - 1 + 1 - 1 + 0.9}{7} = -0.09$$

Se aplicaron los algoritmos GRASP1, GRASP2, GRASP3 y GRASP3+PR a los datos de la aplicación seleccionada. Se hicieron 10 corridas para cada algoritmo, la Tabla 23 muestra los resultados promedio, mientras que en la Tabla 24 se muestran los resultados detallados de cada una de las 10 corridas.

Tabla 23. Resultados promedio sobre las 10 corridas de cada algoritmo

	GRASP 1	GRASP2	GRASP3	GRASP3+PR
# seleccionados	24.0	110.8	93.8	97.7
Valor	0.68331107	0.94149604	1.07652046	1.11280577
Tiempo de ejecución	309.2483	236.4243	116.06222	125.9248

Una conclusión de esto es que las instancias de prueba son mucho más complicadas que la instancia real, en la cual hay más homogeneidad entre los individuos que la heterogeneidad que está implícita en la aleatoriedad de las instancias tipo I y tipo II.

Tabla 24. RESULTADOS INDIVIDUALES PARA CADA UNA DE LAS 10 CORRIDAS DE LOS ALGORITMOS

CORRIDA	GRASP3+PR			GRASP3			GRASP1			GRASP2		
	tiempo	valor	m	tiempo	Valor	m	tiempo	valor	m	tiempo	valor	M
1	127.094	1.11542	101	116.631	1.09397	90	307.909	0.65286	20	209.509	0.94060	114
2	125.081	1.11393	100	113.7384	1.09487	96	317.706	0.66818	22	206.374	0.94139	78
3	120.028	1.10484	96	111.7161	1.07699	86	328.236	0.60268	16	212.645	0.90701	107
4	115.622	1.10311	92	113.4575	1.09058	88	330.872	0.70000	26	213.596	0.95940	120
5	114.616	1.10251	94	119.4957	1.09844	101	290.343	0.67857	24	220.633	0.96340	105
6	139.309	1.10811	96	126.6417	1.08316	95	289.407	0.61161	16	178.886	0.95260	129
7	123.162	1.12293	100	128.5092	1.03239	86	316.376	0.68214	24	291.597	0.86371	111
8	134.082	1.12600	100	119.378	1.05797	92	308.56	0.70220	26	272.362	0.98452	108
9	125.688	1.12033	101	109.6741	1.07982	97	329.812	0.75714	32	285.81	0.89794	125
10	134.566	1.11090	97	101.3805	1.05701	107	273.262	0.77773	34	272.831	1.00438	111
PROMEDIO	**125.9248**	**1.11281**	**97.7**	**116.06222**	**1.07652**	**93.8**	**309.2483**	**0.68331**	**24**	**236.4243**	**0.94150**	**110.8**

6.6. Extensión al problema de seleccionar equipos de trabajo

En muchas situaciones prácticas la solución de problemas requiere del trabajo mancomunado de un grupo de personas, como se analizó antes, a más de las habilidades individuales de sus integrantes es importante también la diversidad del grupo.

Por otro lado está la cuestión de las interacciones dentro del grupo, esto tiene que ver con las afinidades o no afinidades entre las personas y también con la potencial formación de subgrupos o alianzas intergrupales. Así, el modelo anterior ignora algunas características importantes, que incluyen la comunicación y el aprendizaje. El siguiente modelo puede ser usado para incluir este tipo de interacciones.

6.6.1. El problema de la máxima eficiencia

Sea $\Phi = \{1, 2, \dots, n\}$ una población de personas desde la cual se quiere seleccionar un subconjunto de ellas, $M \subset \Phi$, de tal manera que M sea el grupo más diverso, y, como resultado del corolario, el más eficiente para resolver un problema dado.

Con el fin de seleccionar el grupo más eficiente se debe describir como esta eficiencia para resolver un problema depende de la selección de M. Sea $eff(M)$ la eficiencia promedio asociada al conjunto M. Si es que el problema asociado es muy simple, o si para la solución de éste las

personas pueden realizar partes del trabajo independientemente, podríamos considerar que la eficiencia promedio del grupo es simplemente igual a la suma de los coeficientes de productividad de cada uno de los elementos del grupo que se representan con $c_i;\ i = 1, 2, \dots, n$, dividido para el número de personas que conforman el grupo, de esta manera:

$$eff(M) = \frac{\sum_{i \in M} c_i}{|M|}$$

Entonces el correspondiente problema de decisión, es aquel en el cuál buscamos escoger el subconjunto M de eficiencia promedio máxima:

$$\max_{M \subset V} eff(M)$$

La complejidad de este problema depende de algunas características del mismo, en el teorema 2 determina un caso en el cual la solución del problema es trivial.

TEOREMA 2: Si el problema que se va a resolver por el grupo de personas puede ser particionado en n partes, y cada parte puede ser realizada por cualquiera de las personas de una población Φ, y $c_i;\ i = 1, 2, \dots, n$, es la eficiencia individual de cada elemento de la población, el grupo que maximiza la eficiencia promedio está conformado por una única persona i con $\max_{i \in V}\{c_i > 0\}$.

Demostración:

Para la solución del problema es obvio que sólo habría que considerar como posibles candidatos a ser escogidos a aquellos elementos i con $c_i > 0$.
En otras palabras, se consideraría a las personas que puedan aportar algo en el proyecto, aumentando la productividad promedio de sus miembros.
La solución sería entonces trivial, seleccionaríamos la persona i con $\max_{i \in V}\{c_i > 0\}$;
Es decir, si el problema es tan sencillo, en lugar de seleccionar varios expertos altamente calificados que pueden resolver cada parte del problema, seleccionaríamos a la persona más capacitada para resolver todo el problema, pues cualquier otra persona que agreguemos al grupo va a tener el efecto de disminuir la productividad promedio de los miembros del grupo.

■

Pero la mayoría de los problemas que tienen que enfrentar los equipos de trabajo en las empresas, no son tan simples como para que cumplan las hipótesis del teorema 2, los trabajos en general no pueden ser divididos en pequeñas tareas para que cada una de ellas pueda ser efectuada por una persona individualmente, y la interacción entre sus miembros es crucial para aumentar o disminuir su productividad.

Es decir, además de los términos que indican la productividad individual de cada persona: c_i, intervienen términos del tipo d_{ij}, para describir las interacciones inter-pareja, y también términos del tipo e_{ijk} para reflejar las interacciones inter-ternas, y así sucesivamente, con lo cual la productividad promedio de un grupo M de personas se verá descrita por la expresión:

$$eff(M) = \frac{\sum_{i\in M} c_i + \sum_{\substack{i,j\in M \\ i<j}} d_{ij} + \sum_{\substack{i,j,k\in M \\ i<j<k}} e_{ijk} + \cdots}{|M|}$$

Se puede esperar que los efectos en la productividad de las interacciones de órdenes más altos tiendan a disminuir en intensidad, y por tanto tienen menos efecto en la valoración de la eficiencia total $eff(M)$, por tanto podemos considerar la expresión hasta las interacciones de segundo orden d_{ij}. Es decir, consideramos que la productividad promedio del grupo M es:

$$eff(M) = \frac{\sum_{i\in M} c_i + \sum_{\substack{i,j\in M \\ i<j}} d_{ij}}{|M|}$$

Así el problema de decisión será:

$$\max_{M\subset V} eff(M) = \frac{\sum_{i\in M} c_i + \sum_{\substack{i,j\in M \\ i<j}} d_{ij}}{|M|} = \frac{\sum_{i\in M} c_i}{|M|} + \frac{\sum_{\substack{i,j\in M \\ i<j}} d_{ij}}{|M|} \qquad (6.1)$$

Nótese que el problema $\mathcal{P}_1$ cuya función objetivo está definida por la ecuación (6.1) es una generalización del problema Max-Mean $\mathcal{P}_0$, que consiste en el segundo término de la expresión anterior.

Ahora en este problema ya tiene sentido agregar más miembros al grupo, pues ellos pueden aprender unos de otros, favoreciéndose de sus interacciones, aumentado su productividad promedio.

TEOREMA 3: Este problema de determinar el grupo de la máxima eficiencia $\mathcal{P}_1$ es NP-duro

Demostración:

El hecho de que $\mathcal{P}_0$ es NP-duro, , lo cual está demostrado en 2.4, significa que cada instancia $\mathscr{p}$ de cada problema $\mathcal{P}$ de la clase NP puede ser reducido a alguna instancia $\mathscr{p}_0$ del problema $\mathcal{P}_0$. Pero, ya que el problema $\mathcal{P}_1$ es más general que el problema $\mathcal{P}_0$, cada instancia $\mathscr{p}_0$ del problema $\mathcal{P}_0$ es también una instancia del problema más general $\mathcal{P}_1$.
Así, cada instancia $\mathscr{p}$ de cada problema $\mathcal{P}$ puede ser reducida a alguna instancia $\mathscr{p}_0$ del problema $\mathcal{P}_1$. Es decir, el problema más general $\mathcal{P}_1$ es también NP-duro.

■

6.6.2. Propiedades del problema y signos de los coeficientes

Consideremos la productividad total del grupo, $eff_T = \sum_{i \in M} c_i + \sum_{\substack{i,j \in M \\ i<j}} d_{ij}$, si sólo se seleccionan dos individuos: $M = \{i, j\}$, esta eficiencia total es:

$$eff_T(M) = c_i + c_j + d_{ij}$$

Nótese que si el problema es simple y puede subdividirse en tareas que pueden ser realizadas independientemente por distintas personas, $d_{ij} = 0$, y esta eficiencia total sería $c_i + c_j$. Pero si esto no es así, y el problema no puede subdividirse fácilmente, al tener dos personas igualmente capacitadas, estas podrían terminar interfiriendo o entorpeciendo su trabajo, de esta manera si la eficiencia total es parecida a la eficiencia individual de cada una de las personas, $eff_T(M) \approx c_i \approx c_j$, entonces:

$$eff_T(M) = c_i + c_j + d_{ij} < c_i + c_j \rightarrow d_{ij} < 0$$

Pero por otro lado, si el grupo es diverso, sus miembros podrían aprender unos de otros favoreciendo la productividad total del grupo, y así $eff_T(M) < c_i + c_j$, con lo cual:

$$eff_T(M) = c_i + c_j + d_{ij} > c_i + c_j \rightarrow d_{ij} > 0$$

Es decir los signos de d_{ij} indican si la interacción es beneficiosa o perjudicial en el grupo, y es la razón por la cual los grupos diversos podrían ser más eficientes en la resolución de problemas que grupos formados por gente altamente capacitada.

6.6.3. Modelo de Programación Matemática

Si usamos las variables de decisión binarias:

$$x_i = \begin{cases} 1 \text{ si la persona } i \text{ es seleccionada} \\ 0 \text{ si no} \end{cases}$$

Entonces el problema de decisión se puede expresar mediante la siguiente formulación de programación matemática:

$$\max_{\substack{x_i \in \{0,1\} \\ i \in \{1,2,\ldots,n\}}} \frac{\sum_i c_i x_i + \sum_{i<j} d_{ij}\, x_i x_j}{\sum_i x_i} \equiv \max_{\substack{x_i \in \{0,1\} \\ i \in \{1,2,\ldots,n\}}} \left\{ \frac{\sum_i c_i x_i}{\sum_i x_i} + \frac{\sum_{i<j} d_{ij}\, x_i x_j}{\sum_i x_i} \right\}$$

Que es una generalización del problema Max-Mean desarrollado en 2.1.5.

Lo interesante de este problema es que, como se verá luego en la sección 7.2, este modelo es equivalente al del problema multiobjetivo en el cuál se quiere determinar no sólo el subconjunto más diverso, sino además el mejor.

Capítulo 7.

La selección del conjunto más diverso y mejor como un problema Multiobjetivo

7.1. Problemas de optimización con objetivos múltiples

7.1.1. Los Problemas Multiobjetivo

Es un hecho evidente, a partir de nuestra vivencia diaria, que los problemas de toma de decisión que encontramos en la vida real a menudo incluyen más de un objetivo, así por ejemplo las amas de casa que van al mercado buscan seleccionar productos de la mejor calidad, más nutritivos y saludables, más sabrosos, y de menor costo. El problema es que, también a menudo, estos objetivos están en conflicto y son contradictorios entre sí, y la existencia de estos conflictos entre los objetivos involucrados hará que la mejora de uno de ellos dé lugar a un empeoramiento de algún otro, pues mientras aumentamos la calidad de un producto su precio probablemente aumentará, y mientras más nutritivo y sano es posiblemente será menos sabroso (pensemos por ejemplo en el pan integral, que es más saludable, pero menos sabroso que el pan blanco). Una implicación importante de esto es que normalmente no existirá una solución "ideal" que optimice de forma simultánea todas las funciones objetivo del problema.

Entonces estamos frente a lo que se denomina un Problema de Optimización Multiobjetivo, o Problema de Optimización Multicriterio, en el cuál la ama de casa busca encontrar una solución eficiente que sea "buena" para todos los objetivos que se consideren, intentará llegar por tanto a una situación de equilibrio en la que todos los objetivos sean satisfechos en un grado aceptable, desde el punto de vista de las preferencias de la ama de casa. Así, a diferencia de los problemas de optimización con un único objetivo, el concepto de óptimo se vuelve subjetivo y será necesario decidir de alguna forma cuál es la mejor solución al problema. Lo recomendable sería que la mencionada ama de casa tenga un conjunto de mejores decisiones "suboptimales" y escoja cualquiera de ellas como la más adecuada.

Así, el Problema de Optimización Multiobjetivo (POM) puede describirse brevemente como el problema de encontrar un vector de valores de las variables de decisión que satisfagan un conjunto dado de restricciones y optimice un vector de funciones objetivo. El término “optimizar” en este caso tiene un significado diferente al del caso de problemas con un único objetivo. En los POM las funciones objetivo conforman la formulación matemática de los criterios de desempeño, que generalmente están mutuamente en conflicto y que pueden estar evaluadas en unidades de medida diferentes.

7.1.2. Definiciones

Estas son las definiciones de algunos elementos que intervienen en los POM:

ATRIBUTOS: Son las características, cualidades o parámetros de desempeño de una decisión. Por ejemplo en el problema de decisión de la ama de casa los atributos serán consto del producto, aspecto, presentación, etc.

OBJETIVOS: Indican como son percibidas las nociones de mejorar o ganar por parte del tomador de decisiones. En el ejemplo anterior un objetivo puede ser minimizar el costo. Este objetivo indica que el tomador de decisiones prefiere un costo bajo, en ese sentido menor es mejor.

METAS: Son niveles, categorías o umbrales deseados para los atributos u objetivos que se espera alcanzar en las soluciones.

CRITERIO: Son las reglas de aceptabilidad o estándares de juicio para las decisiones, los criterios consideran atributos, metas y objetivos.

Hay que anotar sin embargo, que en la literatura muchos autores usan los términos objetivo, criterio y atributo de manera intercambiable para representar las metas u objetivos a realizarse en un problema multiobjetivo.

MEJOR SOLUCION: En los problemas de decisión con un único objetivo, la mejor solución es la solución óptima, para la cual el valor de la función objetivo es maximizado (o minimizado) y no puede ser superado por ninguna otra alternativa en el conjunto de todas las soluciones

factibles. En cambio, en los POM generalmente el óptimo de cada criterio no apunta a la misma alternativa, así pues, al tener varias funciones objetivo, la noción de optimo cambia, y lo que se trata de obtener es buenos compromisos (llamados "trade-offs" en inglés) en lugar de una solución única como en los problemas con un único objetivo. En ese contexto la noción de "optimo" que suele adoptarse más comúnmente es aquella que fue generalizada por Vilfredo Pareto, conocido como el óptimo de Pareto.

Para visualizar el concepto del óptimo de Pareto volvamos al ejemplo anterior, en el cuál la ama de casa está interesada en adquirir un producto con los criterios de maximizar su calidad y minimizar su costo de adquisición. Las elecciones posibles pueden ser dibujadas en un plano cartesiano, donde cada eje representa el objetivo calidad y costo, respectivamente. La solución ideal (utópica) es aquella con el menor costo (cero) y la mayor calidad, representada en la Figura 45 como el punto coloreado en rojo, y que además es típicamente infactible. En este gráfico pueden observarse dos tipos de soluciones que son posibles:

- Las soluciones dominadas, que son aquellas soluciones de menor calidad que por lo menos otra solución en todos los objetivos considerados. Por ejemplo como se observa en la Figura 45, la solución P_1 está dominada por la solución P_2, ya que P_2 es de mejor calidad y menor costo que P_1, y por tanto nunca tomaríamos la decisión de escoger la alternativa P_1. Pero es fácil ver que P_1 también está dominada por otras soluciones, por tanto, tanto P_1 como P_2, pertenecen al conjunto de decisiones dominadas, representados en azul en la Figura 45.

- Las soluciones no dominadas son aquellas para las que no existe otra solución que es mejor en todos los objetivos considerados. El conjunto de decisiones no dominadas configura la denominada "frontera de Pareto". Por tanto, desde un punto de vista práctico, las soluciones no dominadas son las únicas que son interesantes de identificar en un problema multiobjetivo.

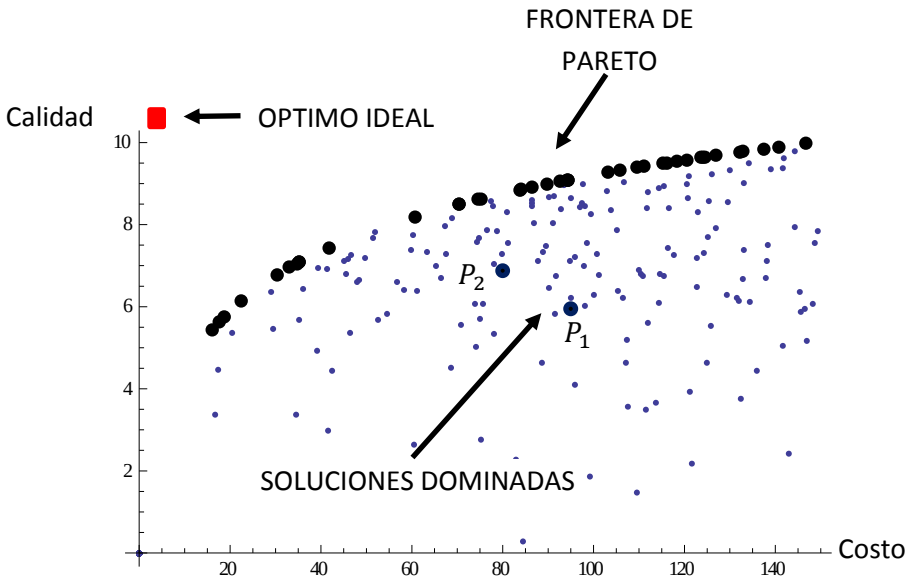


Figura 45. Soluciones dominadas, óptimos de Pareto y óptimo ideal

7.1.3. Modelo general del problema de Optimización Multiobjetivo

Más formalmente, el POM puede formularse matemáticamente y sin pérdida de generalidad de la siguiente manera:

$$Max\ F(x) = \{f_1(x), f_2(x), \dots, f_k(x)\}$$

$$s.t.\ x \in \Omega$$

Donde $x = \{x_1, x_2, \dots, x_n\}$ es una decisión específica

Ω es el conjunto de decisiones factibles

$f_i: \Omega \to \mathbb{R}; i = 1, \dots, k$ son las funciones que describen los objetivos individuales

Si el problema es de minimización, o una combinación de minimización de algunos objetivos y maximización de otros, puede ser reformulado como un problema de maximización pura eligiendo adecuadamente los signos de las funciones individuales f_i.

7.1.4. Soluciones dominadas y óptimo de Pareto

A continuación se enuncian algunas definiciones formales sobre la optimalidad de Pareto:

- Se dice que una solución $x \in \Omega$ domina a una solución $y \in \Omega$, si y solamente si:

$$\forall\, i = 1, 2, \dots, k\colon f_i(x) \geq f_i(y)\ y\ \exists\, j \in \{1, 2, \dots, k\}\ tal\ que\ f_j(x) > f_j(y)$$

Es decir, x es estrictamente mejor que y en al menos uno de los objetivos, y nunca es peor en ninguno de ellos. Cuando x domina a y se representa con: $x \succ y$.

- Una solución $x \in \Omega$ se denomina no dominada o Pareto optimal si $\nexists\, y \in \Omega\ tal\ que\ y \succ x$
- Se denomina Frontera de Pareto al conjunto: $\wp = \{\, x \in \Omega / x\ es\ no\ dominada\}$
- Se dice que una solución $x \in \Omega$ ε-domina a una solución $y \in \Omega$ para algún $\varepsilon > 0$ si y solo si:

$$\forall\, i = 1, 2, \dots, k\colon (1 + \varepsilon) f_i(x) \geq f_i(y)$$

Con el concepto débil de la ε-dominancia se amplia el concepto de la optimalidad de Pareto, originando las siguientes implicaciones:

1) La primera implicación es que la región de soluciones ε-dominadas contiene a la región de dominancia pura de Pareto, de esta manera muchas soluciones Pareto-optimales pasan a ser ε-dominadas, y por tanto ya no son consideradas parte de la frontera de Pareto, así la región de soluciones ε-dominadas incluye soluciones Pareto optimales.
2) La segunda implicación de la ε-dominancia proviene de la primera: el conjunto de soluciones ε-no dominadas depende de la elección de ε y del orden en el que son añadidas (o removidas) las soluciones al conjunto. Para comprender esto obsérvese la Figura 46 y las soluciones x e y, donde x es una solución Pareto dominada y y es una solución de la frontera pura de Pareto. Si consideramos primero la solución x, y pasa a ser ε-dominada, pero si consideramos primero y, entonces x será ε-dominada. Por tanto el conjunto de las soluciones ε-dominadas dependerá del orden de selección de las soluciones.

De esta manera la relación de ε-dominancia origina dos conjuntos distintos de soluciones que son de interés:

- El conjunto ε-aproximado de Pareto, conformado por las soluciones que son ε-dominantes pero no necesariamente Pareto optimales.
- El subconjunto de todas las soluciones no dominadas, denominado conjunto ε-Pareto cuyos elementos son también soluciones Pareto optimales.

3) Otra implicación de la ε-dominancia es de interés práctico, ya que el conjunto ε-Pareto es obviamente más pequeño que el conjunto de Pareto. Esto es muy interesante pues el número de soluciones no dominadas puede ser muy grande. Por un lado los algoritmos de optimización tratan de encontrar la frontera de Pareto completa y no solo partes de ella. Usando ε-dominancia se puede evitar que el algoritmo se concentre en regiones muy densamente pobladas de la frontera, pues se tendrían muchos puntos cercanos que no aportarían gran diferencia al momento de tomar una decisión por uno u otro, evitando aquello se facilitaría al usuario la toma de una decisión.

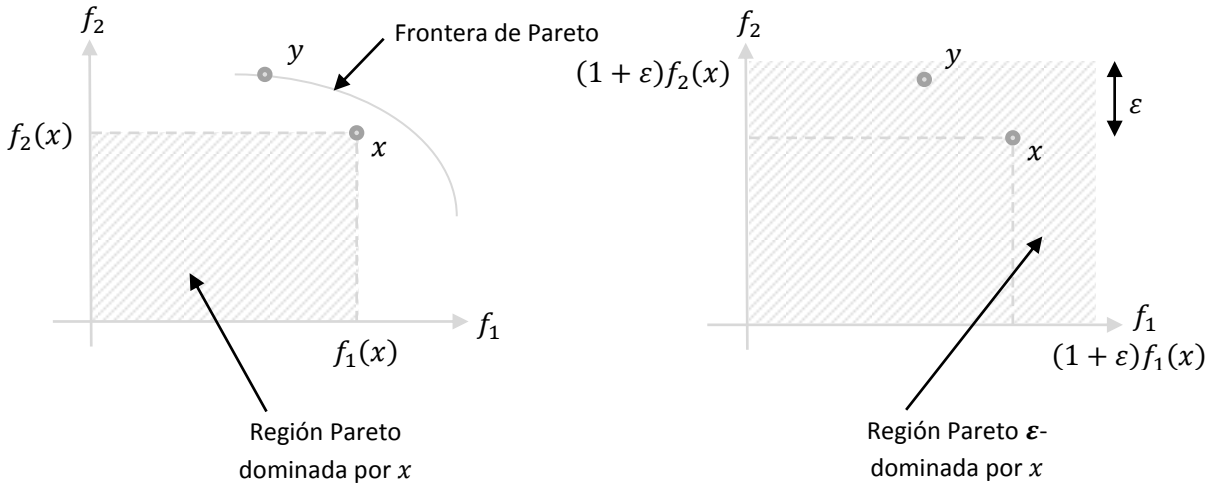


Figura 46. La región Pareto dominada y la región Pareto ε-dominada

4) Por último, hay que tener claro que, en problemas reales, la verdadera frontera de Pareto generalmente no se puede conocer, pues si encontrar el óptimo exacto con un

solo objetivo, por ejemplo en el problema Max-Mean, es una tarea imposible, incrementando otro objetivo tampoco se podrá obtener. Desde este punto de vista, la tarea de los algoritmos de optimización es hallar una buena aproximación de la frontera de Pareto.

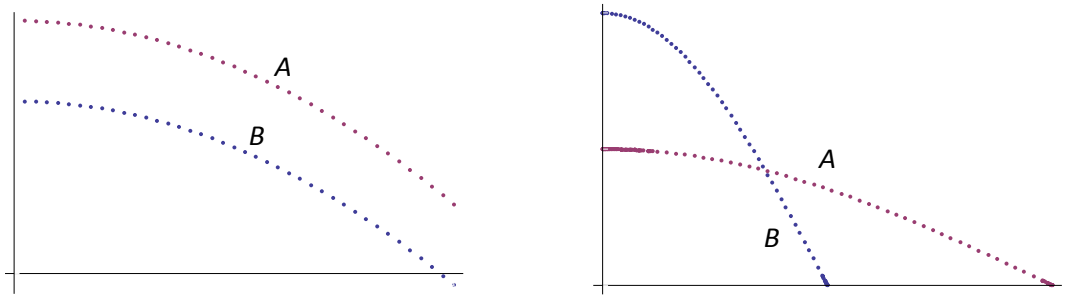
7.1.5. Dominancia de Pareto Extendida

Cuando se tienen dos aproximaciones a la frontera de Pareto, por ejemplo obtenidas con dos algoritmos diferentes, el concepto de dominancia extendida de Pareto permite responder a la pregunta ¿cuál de las dos aproximaciones es mejor?. Para introducir este concepto se consideramos dos conjuntos de soluciones con las cuales se quiere aproximar la frontera de parto, estos conjuntos están mostrados en la Figura 47. Es claro que en el ejemplo a., el conjunto A es una mejor aproximación de la frontera de Pareto que el conjunto B, pero en el ejemplo b esto no es claro. Nótese que aquí necesitamos una definición de dominancia ya no entre parejas de elementos, sino entre parejas de conjuntos.

La extensión del concepto de dominancia se realiza de la siguiente manera:

Definición: Sea ψ = Aproximaciones de la frontera de Pareto. Se dice que $A \in \psi$ domina a $B \in \psi$, y se representa con $A \succ B$, si todas las soluciones de A dominan a todas las soluciones de B; es decir:

$$A \succ B \; si \; y \; solo \; si \; \forall x \in A, \forall y \in B: x \; domina \; a \; y$$



Ejemplo a. Ejemplo b.

Figura 47. Aproximaciones de la Frontera de Pareto

Entonces es claro que en el Ejemplo a. de la Figura 47 si se puede afirmar que $A \succ B$ y que por tanto A es una mejor aproximación a la frontera de Pareto, pero en el Ejemplo b. no se puede decir lo mismo. Por lo tanto son necesarias medidas más sofisticadas para comparar dos aproximaciones de la frontera de Pareto. Las más populares son las basadas en las denominadas funciones de indicadores de calidad, que no comparan la pareja de conjuntos, sino que calculan un número para cada aproximación de la frontera de Pareto, a través de la función $I: \psi \rightarrow \mathbb{R}$ que satisface las propiedades de monotonía estricta, invarianza de escala y cálculo eficiente:

MONOTONIA ESTRICTA: Significa que esta función de indicadores sea compatible con la relación de dominancia de Pareto; es decir,

$$\forall A, B \in \psi: A \succ B \ \rightarrow I(A) > I(B)$$

INVARIANZA DE ESCALA: Significa que si los valores de la función objetivo están escalados por una transformación monótona, por ejemplo si los normalizamos en el intervalo [0,1], entonces el valor de I no debe cambiar.

CÁLCULO EFICIENTE: Significa que el cálculo de $I(A)$ deberá ser computacionalmente de bajo costo $\forall A \in \psi$.

Existen diferentes funciones de indicadores de calidad que se han descrito en la literatura, pero la más usada es la función indicadora de hipervolumen, representada con I_H, que consiste en el cálculo del hipervolumen encerrado por el conjunto de aproximación de la frontera de Pareto, esto requiere también de la selección de un punto a partir del cual se genera el hipervolumen, idealmente este punto de referencia debe tener valores peores que cualquier solución del conjunto, tal como se observa en la Figura 48. El uso de I_H tiene un gran efecto práctico, pues se puede hacer una comparación de dos o más aproximaciones de la frontera de Pareto para decidir cuál es mejor

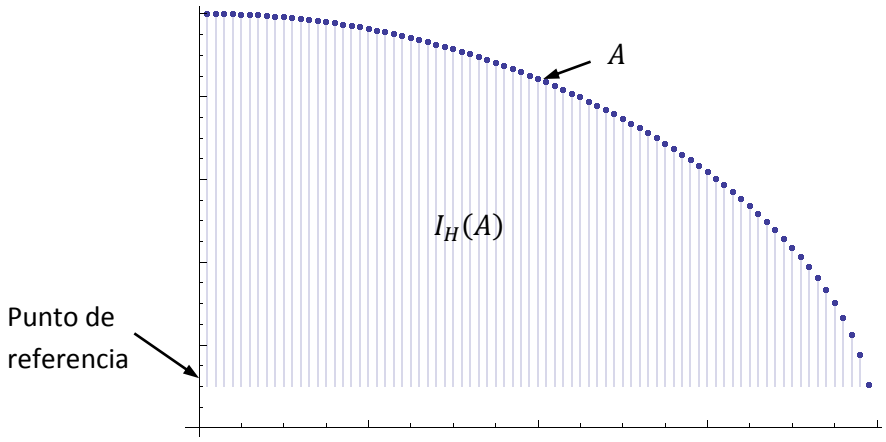


Figura 48. Hipervolumen generado por un conjunto de aproximación de la frontera de Pareto con dos funciones objetivo

Tal como se muestra en la Figura 48, el hipervolumen es calculado tomando la unión de todos los hipercubos, los cuales están construidos desde el punto de referencia, y una de las soluciones como el vértice opuesto. Para dos objetivos el cálculo es sencillo, ya que el hipervolumen puede ser dividido en muchos rectángulos que no se solapan y por lo tanto será igual a la suma de las áreas de dichos rectángulos. En el caso general, con más objetivos, el cálculo del hipervolumen podría ser muy complicado, requiriendo un tiempo de cálculo exponencial en el número de objetivos, como se describe en (3).

7.1.6. Métodos de solución de un POM

En la literatura se han descrito una gran variedad de métodos para resolver un POM, los más usados, para fines prácticos, son precisamente aquellos que permiten obtener la frontera de Pareto o una aproximación de ella, y no únicamente una solución Pareto optimal. Pues al tener un conjunto de soluciones, en lugar de una sola, el tomador de decisiones finalmente podrá decidirse por alguna de ellas en base a sus preferencias o posiblemente basado en alguna otra consideración no incluida en las restricciones o en los objetivos del POM, por ejemplo, en el problema que estamos resolviendo, definido por las ecuaciones (1.1) y (1.2), el tomador de decisiones finalmente podría elegir una de las soluciones generadas en función al número de elementos seleccionados, característica que no está tomada en cuenta en las restricciones del problema.

Entre los métodos más usados para resolver un POM están: el método del orden lexicográfico, programación por metas, método de las ponderaciones, método de la ε-restricción, Escalarización de logros, etc., cada uno de estos métodos buscan determinar una aproximación de la frontera de Pareto. Ahora bien, ¿qué método es el que más se adecua para resolver nuestro problema original, descrito en las ecuaciones (1.1) y (1.2)?. Para responder a esta pregunta tenemos que observar que si sólo se considera el primer objetivo (1.1), el problema ya es de por sí difícil, pues pertenece a la clase de problemas NP-duros. Por otro lado, se puede utilizar el hecho de que en el Capítulo 4 y Capítulo 5 se ha diseñado una heurística muy eficiente para resolver este problema con un único objetivo. Luego, si utilizamos un método que transforme al problema multiobjetivo en un problema con objetivo único que sea del tipo que se ha resuelto por el algoritmo GRASP3+PR, tendríamos un método para resolver el problema. La respuesta a la pregunta está por tanto en el método de las ponderaciones, pues según se establece en 7.2, el POM puede ser transformado en un problema del tipo Max-Mean, que es resuelto eficientemente por el algoritmo GRASP3+PR. El método de las ponderaciones funciona de la siguiente manera:

MÉTODO DE LAS PONDERACIONES: Este método permite escalarizar la función objetivo del POM, combinando linealmente las funciones uniobjetivo que son parte del POM en una única función objetivo, donde los coeficientes de la combinación lineal son los pesos (importancia) que se asigna a cada uno de los objetivos del problema. Es decir, dado el problema $(\boldsymbol{P})$ de POM:

$$(\boldsymbol{P}) \quad Max\ F(x) = \{f_1(x), f_2(x), \dots, f_k(x)\}$$

$$s.t. x \in \Omega$$

El método de las ponderaciones considera ahora el problema $(\boldsymbol{P_\alpha})$ escalarizado:

$$(\boldsymbol{P_\alpha}) \quad Max \sum_{i=1}^{k} \alpha_i f_i(x)$$

$$s.t. x \in \Omega$$

Si representamos con $S(P_\alpha)$, $\alpha \geq 0$, al conjunto de soluciones de dicho problema, es obvio que $\forall \lambda > 0$: $S(P_{\boldsymbol{\alpha\lambda}}) = S(P_\alpha)$. Por tanto, sin pérdida de generalidad, se puede tomar el vector de pesos normalizado; es decir, si $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_k)$; entonces, $\alpha_i \geq 0$ para todo $i = 1, \dots, k$, y

$\sum_{i=1}^{k} \alpha_i = 1$. El siguiente resultado se encuentra demostrado en (45), y determina la utilidad del método de las ponderaciones.

TEOREMA: Si x^* es solución del problema escalarizado $(\boldsymbol{P_\alpha})$, con todos los pesos estrictamente positivos; es decir, $\alpha_i \geq 0$ para todo $i = 1, \dots, k$, entonces x^* es un punto óptimo de Pareto del problema original.

Por tanto, al resolver un POM por el método de las ponderaciones, obtenemos una solución eficiente, de la frontera de Pareto, si restringimos los pesos a ponderaciones estrictamente positivas.

La debilidad del procedimiento está en que no se tiene el recíproco de lo que establece el teorema; es decir, no todas los puntos Pareto optimales se pueden obtener con alguna combinación de valores para los coeficientes $\alpha_i \geq 0$, salvo cuando todas las funciones objetivo unidimensionales son convexas.

Entonces, este enfoque es efectivo y puede ser automatizado escogiendo α_i en el intervalo [0,1]. Variando gradualmente los valores de α_i se va generando paulatinamente el conjunto de soluciones.

7.2. Transformación del Problema Multiobjetivo en uno del tipo Max-Mean

Como se planteó en la introducción de esta investigación, nuestro objetivo no sólo es determinar el subconjunto más diverso en promedio, sino además el mejor, esto último en el sentido que los nodos seleccionados sean los mejores respecto a sus pesos. Como se planteó en las ecuaciones (1.1) y (1.2), el problema es seleccionar el subconjunto M que optimice el POM siguiente:

$$Max\ f_1(M) = div(M)$$
$$Max\ f_2(M) = \sum_{i \in M} w(i)$$

Donde $div(M)$ es la medida de diversidad calculada sobre el conjunto M, y $w(i)$ son los pesos o score de cada nodo i, que refleje la importancia de considerar este nodo en la solución final.

Adaptado con nuestra medida de la diversidad del promedio, el problema toma la forma del siguiente POM:

$$Max\ f_1(M) = \frac{\sum_{i<j,i,j\in M} d_{ij}}{|M|}$$

$$Max\ f_2(M) = \sum_{i\in M} w(i)$$

En el Capítulo 4 y en el Capítulo 5 se diseñó una heurística muy eficiente para resolver el problema de optimización que involucra únicamente el primer objetivo. Sin embargo, este método no considera la importancia de los pesos de los nodos $w(i)$ en la solución. Una forma de plantear esto es tratar de transformar el problema, escalarizando la función objetivo vectorial compuesta por los dos objetivos individuales, y esto se puede lograr integrando los pesos de los objetos dentro de las distancias d_{ij}, como se ve a continuación.

La meta es representar al problema, y resolverlo, de tal manera que se pueda presentar al usuario un conjunto de soluciones Pareto optimales o ε-Pareto optimales. Así es como entra en juego el método de las ponderaciones, con el que la función objetivo puede ser escalarizada de la siguiente manera:

$$f(M) = \alpha\frac{\sum_{i<j,i,j\in M} d_{ij}}{|M|} + (1-\alpha)\sum_{i\in M} w(i)$$

Si representamos con $m = |M|$, consideramos dos veces cada distancia d_{ij} y retiquetamos a los elementos del conjunto M como:

$$M = \{u_1, u_2, \dots, u_m\}$$

Entonces tendremos una modificación de la función objetivo original, esta nueva función objetivo puede expresarse como:

$$f(M) = \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha \frac{d(u_i, u_j)}{2m} + (1-\alpha)\sum_{i=1}^{m} w(u_i)$$

Antes de seguir con las transformaciones, hay que observar que al considerar dos veces cada distancia, la diversidad promedio es el doble de la usual, por tanto el denominador del término que determina la diversidad es $2m$, lo cual no distorsiona al problema. Nuestro objetivo es lograr expresar este problema como un problema de la diversidad máxima del tipo Max-Mean. Continuando con las transformaciones tenemos:

$$f(M) = \sum_{i=1}^{m}\left(\sum_{j=1}^{m} \alpha \frac{d(u_i, u_j)}{2m} + (1-\alpha) w(u_i)\right)$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m}\left(\alpha \frac{d(u_i, u_j)}{2m} + \frac{(1-\alpha) w(u_i)}{m}\right)$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m}\left(\alpha \frac{d(u_i, u_j)}{2m} + (1-\alpha)\frac{w(u_i) + w(u_j)}{2m}\right)$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\left(\alpha \frac{d(u_i, u_j)}{2} + (1-\alpha)\frac{w(u_i) + w(u_j)}{2}\right)}{m}$$

$$f(M) = \sum_{i=1}^{m}\sum_{j=1}^{m}\frac{d'(u_i, u_j)}{m}$$

Dónde:

$$d'(u_i, u_j) = \frac{\alpha d(u_i, u_j) + (1-\alpha)\left(w(u_i) + w(u_j)\right)}{2}$$

Es decir, el nuevo problema escalarizado el del mismo tipo que el Max-Mean, considerando una transformación a las distancias inter elemento d_{ij}, a las cuales se añade de manera ponderada los pesos de cada uno de los nodos i, j.

Mediante esta transformación estamos deformando el espacio original, estirando las distancias entre nodos con gran peso, y encogiendo las distancias entre nodos de poca importancia. De esta manera los métodos diseñados para resolver el problema de optimización con el único objetivo de maximizar la diversidad pueden ser usados para resolver este problema modificado.

Por supuesto, el hecho de usar las etiquetas de las aristas d' en lugar de las originales altera el problema, y las soluciones a este no son optimales. Sin embargo hay que tener en cuenta que los algoritmos usados son heurísticas, y no hay garantía de hallar la solución óptima. Es decir la verdadera frontera de Pareto con respecto a los problemas del mundo real generalmente nunca es conocida, entonces la única tarea de los métodos que se diseñan en la optimización multiobjetivo es hallar una buena aproximación de la frontera de Pareto.

7.3. Resultados numéricos con los problemas de prueba

Una vez que el problema multiobjetivo, de seleccionar lo mejor y lo más diverso, ha sido transformado en un problema equivalente al problema Max-Mean, y dado que en el Capítulo 4 se diseñó una heurística que se comprobó es muy eficiente para resolverlo, la heurística GRASP3+PR, se puede utilizarla para resolver el problema de seleccionar lo mejor y lo más diverso. El resultado será un conjunto de decisiones Pareto-optimales, del cual el tomador de decisiones podrá seleccionar la que considere más adecuada.

Para mostrar la estructura de las soluciones consideramos problemas de prueba, de tamaño mediano ($n = 150$ nodos), de tipo I y de tipo II. En estos ejemplos los pesos de los nodos, w_i, han sido generados aleatoriamente, con una distribución uniforme en el intervalo [0,1].

En la Figura 49 y la Figura 52 se muestra la Frontera de Pareto generada para una instancia de tipo I y una instancia de tipo II, respectivamente. Los valores de α en la ecuación 0 varían de 0 a 1. Mientras que en la Figura 50 y Figura 53 se muestra el detalle de 1000 soluciones generadas

para valores de α entre 0.8 y 1, para una instancia de tipo I y una de tipo II, respectivamente, en estas figuras se puede apreciar las soluciones dominadas y las soluciones Pareto optimales obtenidas. Es interesante observar estos puntos en la Figura 51 y Figura 54 en relación con el número de nodos seleccionados en la solución respectiva, nótese que cuando los pesos son más altos, el número de elementos seleccionados aumenta, esto es porque estamos considerando más importancia a la calidad de los nodos (medida por sus pesos w_i), y por tanto se está forzando a que más nodos entren en la solución, en detrimento de la diversidad del subconjunto seleccionado.

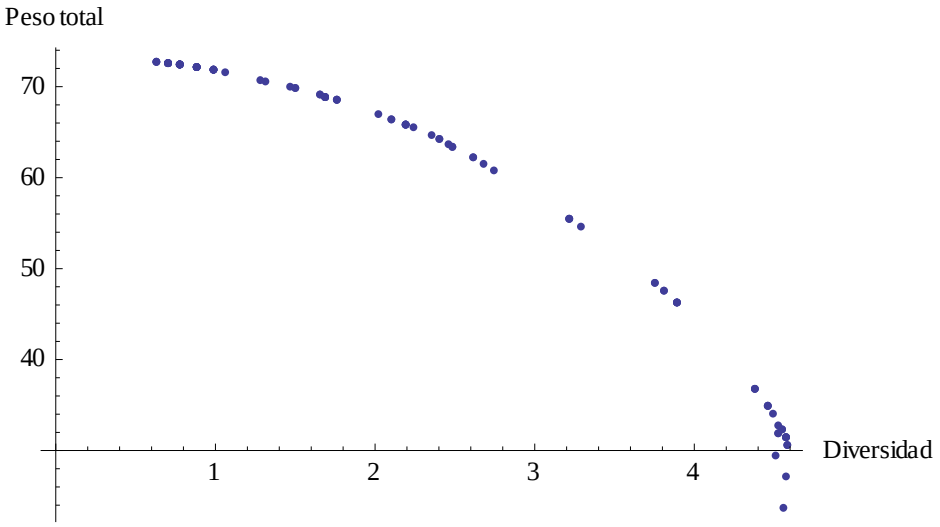


Figura 49. Aproximación de la Frontera de Pareto para una instancia de tipo I de tamaño mediano $n = 150$

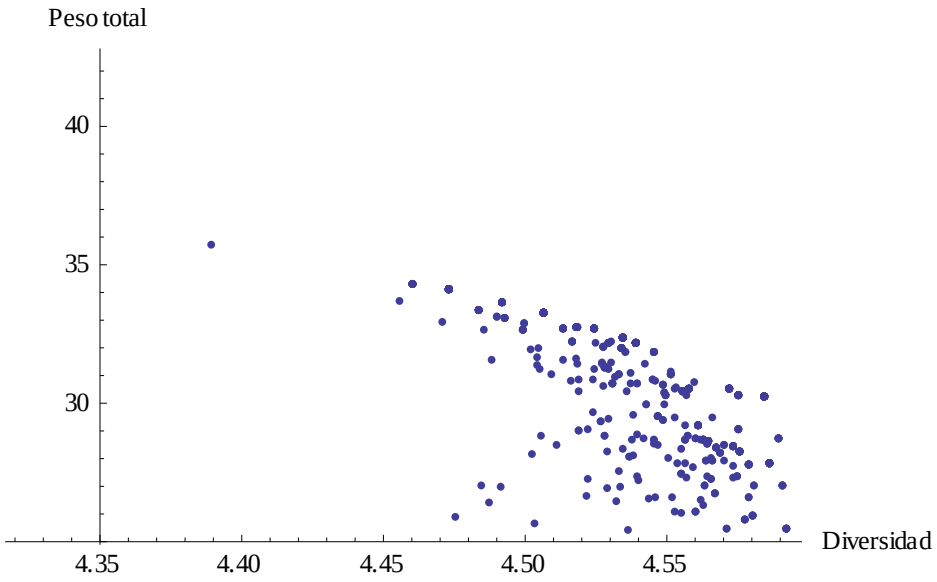


Figura 50. Detalle de la Frontera de Pareto para una instancia de tipo I de tamaño mediano $n = 150$

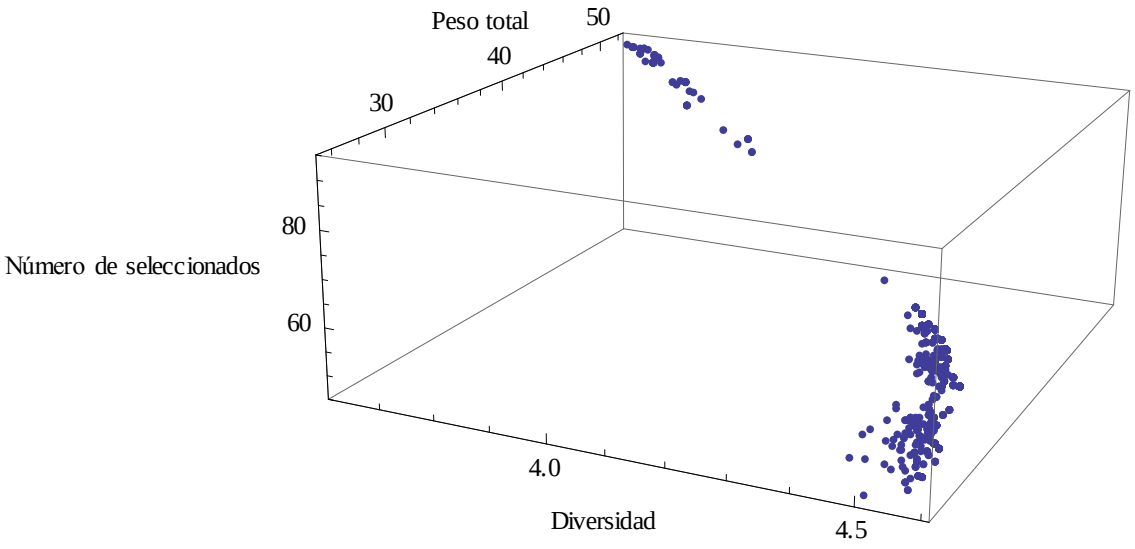


Figura 51 Las soluciones del problema multiobjetivo y el número de elementos seleccionados para una instancia de tipo I de tamaño mediano $n = 150$

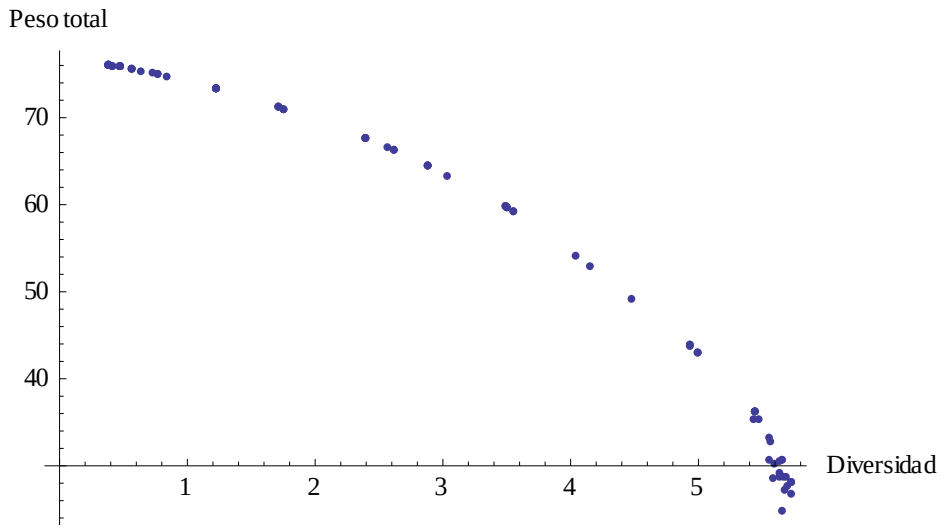


Figura 52. Aproximación de la frontera de Pareto para una instancia de tipo II de tamaño mediano $n = 150$

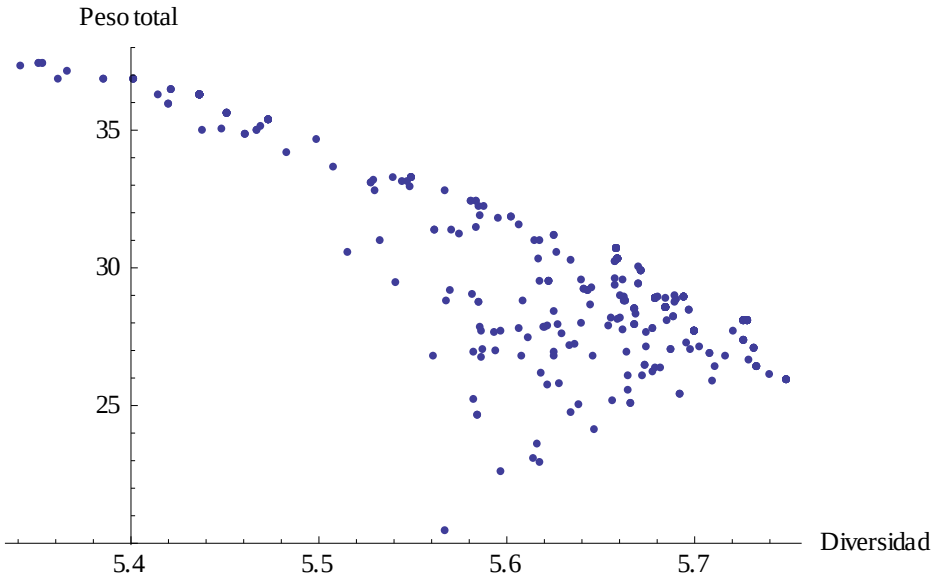


Figura 53. Detalle de la Frontera de Pareto para una instancia de tipo II de tamaño mediano $n = 150$

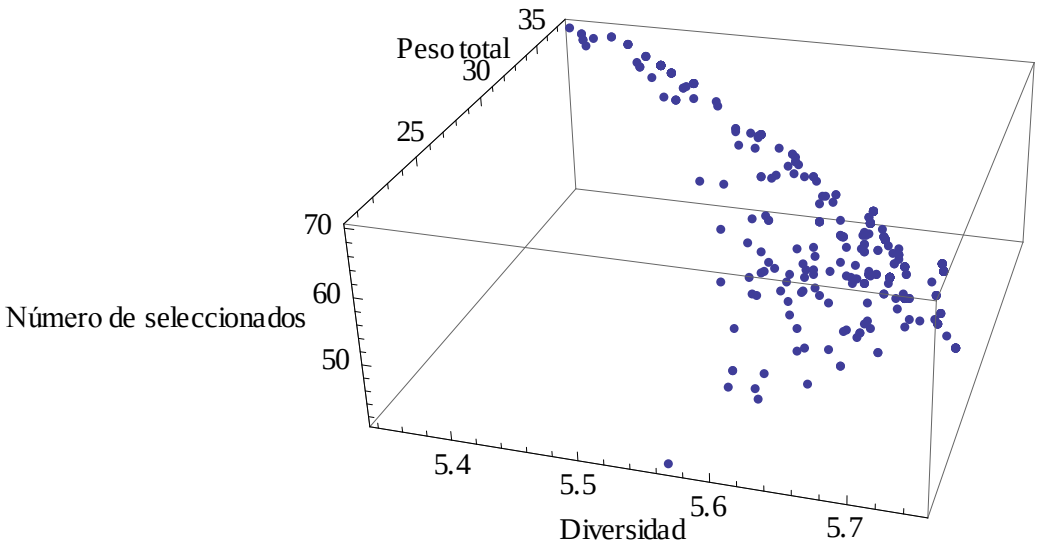


Figura 54. Soluciones del problema multiobjetivo y el número de elementos seleccionados para una instancia de tipo II de tamaño mediano $n = 150$

Algunas observaciones pueden ser realizadas a partir de estos experimentos numéricos, y sobre todo de las figuras:

- La distribución de las soluciones conforma una región convexa, que según el teorema de la página 162 determina una buena aproximación de la frontera de Pareto.
- La ventaja de usar la representación 3D de la frontera de Pareto, originada por la incorporación de la variable: número de elementos seleccionados, proporciona una forma interesante de resolver el problema de decisión resultante; esto es, decidir por la solución no dominada que corresponda a un número de elementos que el tomador de decisiones considere conveniente.

CONCLUSIONES

Esta investigación permitió hacer un aporte al estado actual del conocimiento sobre el diseño de algoritmos basados en métodos GRASP y path relinking para resolver problemas de optimización multiobjetivo en los cuales uno de sus objetivos configura un problema de tipo NP-duro. En este trabajo se ha considerado un problema bi-objetivo difícil de resolver para así poder evaluar la eficacia y eficiencia de la metodología basada en GRASP y path reliking. En la investigación se ha usado el problema de la diversidad máxima como el problema a tratar, al cual se le ha incorporado otro objetivo: el de seleccionar lo mejor, con lo cual el problema a resolver fue el de seleccionar lo mejor y lo más diverso de una población.

Se ha aportado un nuevo modelo al problema de optimización combinatoria clásico de la diversidad máxima, al modificar la forma de medir la diversidad. Este nuevo enfoque es particularmente útil para resolver problemas en los que se tienen relaciones de afinidad entre los objetos, esto se logra al incorporar un signo, positivo o negativo, a las relaciones de afinidad entre los nodos.

Se han revisado algoritmos previos, los más eficientes reportados en la literatura, para problemas similares al problema presentado en esta investigación, para así poder determinar la eficiencia de la metodología propuesta. En concreto, se proponen varias adaptaciones de algoritmos eficientes que se proponen en la literatura para resolver los problemas MaxSum y MaxMinSum, cuyas soluciones tienen alta correlación con el modelo MaxMean que se estudia aquí. Así, el método propuesto se compara con los mejores métodos propuestos en la literatura relacionada. La conclusión que se puede extraer de esta comparativa es que la métodología propuesta supera claramente a los métodos previos.

Por otro lado se determina que el tratamiento del problema en software comercial de uso profesional, de tipo independiente de contexto, no da resultados satisfactorios y las soluciones que se obtienen son de muy mala calidad a pesar de proporcionar un gran tiempo de ejecución a estos programas.

La generalización de los conceptos de la distancia inter elemento que se usó en esta investigación permitió superar las limitaciones de las métricas usadas tradicionalmente en los

problemas de diversidad máxima, pues permite introducir conceptos como la afinidad, que generaliza las aplicaciones del problema clásico.

La determinación de algunas propiedades del problema, a partir de las características de la relación de afinidad entre los nodos, y de la forma del espacio de soluciones, permitió diseñar una metodología que se probó ser muy eficiente, y por supuesto, resulta útil para la creación de nuevas metaheurísticas que puedan explotar esta estructura del espacio de soluciones.

Aunque aún sea muy difícil encontrar una solución óptima para el problema uni objetivo MaxMean, o, el frente Pareto del problema bi-objetivo analizado, el conocer la forma convexa de este frente, permitirá acotar la región de búsqueda a un conjunto más pequeño de soluciones, y a partir de ahí se podrá desarrollar otros algoritmos adecuados, que logren mejores aproximaciones a las soluciones Pareto.

Se puede avanzar en el desarrollo de nuevas metodologías para resolver el problema, como la creación de metaheurísticas basadas totalmente en explotar la característica de convexidad del espacio de soluciones, y en la identificación de problemas de optimización combinatoria similares.

Aunque el algoritmo GRASP desarrollado no garantiza la optimalidad de las soluciones, se puede asegurar que el algoritmo si encontró soluciones muy buenas, ya que los puntos generados se Evaluación de cercanía al óptimo.

Hay una gran dificultad para tratar al problema con métodos exactos, el tratamiento con métodos estándar como ramificación y acotamiento resultó inaplicable para problemas de tamaño mediano o grande.

Los programas informáticos de tipo caja negra no permitieron obtener soluciones buenas a pesar de proporcionar un gran tiempo de ejecución en problemas de tamaño mediano.

REFERENCIAS

1. *Intelligence Techniques Are Needed to Further Enhance the Advantage with Diversity in Problem Solving.* **Castillo, O., y otros, y otros.** 2009. IEEE Workshop Hybrid Intelligent Models and Applications. HIMA'09. págs. 48-55.

2. **Page, S.** *The Difference: How the Power of Diversity Creates better Groups, Firms, Schools, and Societies.* New Jersey : Princenton University Press, 2007.

3. **Meinl, Thorsten.** *Maximum-Score Diversity Selection.* primera. s.l. : Südwestdeutscher Verlag, 2010.

4. *Analytic Models for locating undesirable facilities.* **Ercut, E. y Neuman, S.** 1989, European Journal of Operational Research, Vol. 40, págs. 275-291.

5. *The dense k-subgraph problem.* **Feige, U., Kortsarz, G. y Peleg, D.** 3, 2001, Algorithmica, Vol. 29, págs. 410-421.

6. *Heuristic Algorithms for the Maximum Diversity Problem.* **Glover, F., Kuo, C. C. y Dhir, K.** 1, 1998, Journal of Information and Optimization Sciences, Vol. 19, págs. 109 - 132.

7. *A discrete optimization model for preserving biological diversity.* **Glover, F., Kuo, C. y Dhir, K. S.** 1995, Appl. Math. Modeling, Vol. 19, págs. 696 - 701.

8. *An application of tabu search heuristic for the maximum edge-weighted subgraph problem.* **Macambira, E.** 2002, Annals of Operational Research, Vol. 117, págs. 175-190.

9. *Iterated greedy for the maximum diversity problem.* **Lozano, M., Molina, D. y García-Martínez, C.** 2011, European Journal of Operational Research.

10. *Analyzing and modeling the maximum diversity problem by zero-one programming.* **Kuo, M., Glover, F. y Dhir, K.** 24, 1993, Decision Sciences, págs. 1171 - 1185.

11. *Tabu Search and GRASP for the Maximum Diversity Problem.* **Duarte, A. y Martí, R.** 2007, European Journal of Operational Research, Vol. 178, págs. 71 - 84.

12. *Similarity vs. Diversity.* **Smyth, B. y McClave, P.** s.l. : Springer, 2001, Lecture Notes in Computer Science, Vol. 2080, págs. 347-361.

13. *Social diversity promotes the emergence of cooperation in public good games.* **Santos, F., Santos, M. y Pacheco, J.** 2008, Nature, Vol. 454, págs. 213-216.

14. *Groups of diverse problem solvers can outperform groups of high-ability problem solvers.* **Page, Scott y Hong, Lu.** 46, 2004, PNAS, Vol. 101, págs. 16385 - 16389.

15. **Resende, M. y Ribeiro, C.** Greedy Randomized Adaptative Search Procedures. [ed.] F. Glover and G. Kochenberg. *State of-the-art Handbook in Metaheuristics.* Boston : Kluwer Academic Publishers, 2001, págs. 219-250.

16. **Hansen, P. y Mladenovic, N.** Variable neighborhood search. [ed.] F. Glover y G. Kochenberger. *Handbook in Metaheuristics.* 2003, págs. 145-184.

17. **Glover, F. y Laguna, M.** *Tabu Search.* Boston : Kluwer Academic Publishers, 1997.

18. *The equitable dispersion problem.* **Prokopyev, O., Kong, N. y Martínez-Torres, D.** 197, 2009, European Journal of Operational Research, págs. 59 - 67.

19. *Similarity Measures.* **Santini, Simone y Jain, Ramesh.** 1999, IEEE Transactions on Pattern Analysis and Machine Intelligence.

20. *A similarity Measure for Collaborative Filtering with Implicit Feedback.* **Lee, T. Q. y Park, Y. T.** [ed.] D. S. Huang, L. Heutte y M. Loog. 2007, ICIC 2007, págs. 385-397.

21. *Groups of diverse problem solvers can outperform groups of high-ability problem solvers.* **Lu Hong, Scott E. Page.** 46, 2004, PNAS, Vol. 101, págs. 16385-16389.

22. *The importance of strategic change in achieving equity in diversity.* **French, E.** s.l. : Wiley InterScience, 2005, Strategic Change, Vol. 14, págs. 35–44.

23. *Computational aspects of the maximum diversity problem.* **Ghosh, J.** 19, 1996, Operations Research Letters, págs. 175 - 181.

24. *Competitive Hopfield Network Combined With Estimation of Distribution for maximum DIversity Problems.* **Wang, J., y otros, y otros.** 4, 2009, IEEE Transactions on systems, man, and cybernetics, Vol. 39, págs. 1048-1065.

25. *Solving the maximum edge weight clique problem via unconstrained quadratic programming.* **Alidaee, B., y otros, y otros.** 181, 2007, European Journal of Operational Research , págs. 592 – 597.

26. *The discrete p-dispersion problem.* **Erkut, E.** 1990, European Journal of Operational Research, Vol. 46, págs. 48-60.

27. **Garey, M. R. y Johnson, D. S.** *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York : W.H. Freeman, 1979.

28. *GRASP with path relinking for the max-min diversity problem.* **Resende, M., y otros, y otros.** 37, 2010, Computers and Operations Research, págs. 498 - 508.

29. **Díaz, A., y otros, y otros.** *Optimización heurística y redes neuronales.* Madrid : Paraninfo, 1996.

30. *Comparing local search metaheuristics for the maximum diversity problem.* **Aringhieri, R. y Cordone, R.** 2011, Journal of the Operational Research Society, Vol. 62, págs. 266-280.

31. **Martí, R.** Multistart methods. [ed.] Fred Glover y Gary Kochenberger. *Handbook of Metaheuristics.* s.l. : Kluwer Academic, 2003, págs. 355 - 368.

32. *A probabilistic heuristic for a computationally difficult set covering problem.* **Feo, T. y Resende, M.** 1989, Operations Research Letters, Vol. 8, págs. 67 - 71.

33. *GRASP and VNS for Max-cut.* **Festa, P., y otros, y otros.** 2001, Proceedings of MIC'2001, págs. 371 - 376.

34. **Dréo, J., y otros, y otros.** *Metaheuristics for Hard Optimization.* Berlin : Springer, 2006. págs. 169 - 170.

35. *A hybrid heuristic for the p-median problem.* **Resende, M. y Werneck, R.** 1, 2004, Journal of heuristics, Vol. 10, págs. 59-88.

36. **Doerner, K., y otros, y otros.** *Metaheuristics, Progress in Complex Systems Optimization.* New York : Springer, 2007. págs. 138 - 140.

37. **Martí, R.** Instances MMDPLIB. [En línea] 2011. [Citado el: 24 de 05 de 2011.] http://heur.uv.es/optsicom/mmdp/.

38. **Project, Opticom.** *Opticom Project.* [En línea] Mayo de 2012. http://www.optsicom.es/.

39. **Palisade.** Evolver. *sitio Web de Palisade Corporation.* [En línea] 2011. [Citado el: 05 de Marzo de 2011.] http://www.palisade.com/evolver/.

40. **Hazrat, Roozbeh.** *Mathematica: A Problem-Centered Approach.* [ed.] Springer Undergraduate Mathematics Series. New York : Springer, 2010.

41. *Black box scatter search for general classes of binary optimization problems.* **Gortázar, F., y otros, y otros.** 11, 2010, Computers & Operations Research, Vol. 37, págs. 1977 - 1986.

42. *Context-independient scatter and tabu search for permutation problems.* **Campos, V., Laguna, M. y Martí, R.** 1, 2005, INFORMS Journal on Computing, Vol. 17, págs. 111 - 122.

43. **Palisade.** Evolver -Para optimización por medio de algoritmos genéticos- Palisade. [En línea] 16 de 05 de 2011. [Citado el: 16 de 05 de 2011.] http://www.palisade-lta.com/evolver/.

44. *Capitalizing on Diversity: Interpersonal Congruence in Small Work Groups.* **Polzer, J., Milton, L. y Swann, W.** 2, 2002, Administrative Science Quarterly, Vol. 47, págs. 296 - 324.

45. **Miettinen, K.** *Nonlinear multiobjective optimization.* Massachusetts : Kluwer Academic Publishers, 1999.

46. *An Evolutionary Approach for the Maximum Diversity Problem.* **Katayama, K. y Narihisa, H.** 2005, Studies in Fuzziness and Soft Computing, Vol. 166, págs. 31-47.

47. *Genetic local search with adaptative crossover probability and its application to maximum diversity problem.* **Asada, H., y otros, y otros.** 2010, IEEJ Transactions on Electronics, Information and Systems, Vol. 130, págs. 529-520.

48. **COR.** Computers & Operations Research Special Issue. [En línea] 2011. [Citado el: 12 de 01 de 2011.] http://www.elsevierscitech.com/cfp/Call_for_Papers_COR_GRASP_WITH_PATH_RELINKING.pdf.

49. **Altbach, P. y Salmi, J.** *To roal to academic excellence: the making of world-class research universities.* Washington DC : The World Bank, 2011.

50. *GRASP and Path Relinking for the Equitable Dispersion Problem.* **Martí, R. y Sandoya, F.** [ed.] Elsevier. April de 2012, Computers & Operations Research, págs. 1-18. http://0-dx.doi.org.library.hct.ac.ae/10.1016/j.cor.2012.04.005.

ANEXO 1. Tablas con los resultados detallados de los experimentos numéricos

Tabla A 1. Tiempos de procesamiento con Cplex de cada uno de los problemas de tipo I y tipo II por el procedimiento de resolver (n-1) veces el problema Max-Sum

n	Instancia	cpu tipo I	cpu tipo II
20	1	12.957	19.972
	2	9.879	20.999
	3	9.715	20.059
	4	19.948	17.317
	5	11.533	21.441
	6	22.484	23.779
	7	11.753	21.135
	8	18.28	17.361
	9	17.092	19.61
	10	12.975	12.245
25	1	36.6	67.202
	2	46.21	54.397
	3	41.028	60.42
	4	41.296	67.337
	5	49.471	64.21
	6	36.62	61.26
	7	32.116	72.864
	8	47.929	50.265
	9	39.99	41.961
	10	47.002	55.897
30	1	103.216	152.844
	2	77.609	128.517
	3	109.135	237.688
	4	77.246	339.208
	5	137.198	205.441
	6	102.044	235.111
	7	128.467	120.903
	8	133.806	120.441
	9	81.101	142.015
	10	73.204	139.593

Tabla A 2. Tiempos de procesamiento con Cplex de cada uno de los problemas de tipo I y tipo II por el procedimiento de resolver directamente la formulación del problema Max-Mean

	Instancia	Max-Mean	
n		cpu tipoI	cpu tipo II
20	1	43.081	67.317
	2	15.906	69.947
	3	28.022	76.314
	4	63.816	70.38
	5	39.633	94.651
	6	81.675	86.54
	7	43.361	55.708
	8	75.316	46.19
	9	71.504	61.349
	10	31.028	38.747
25	1	774.595	1941.214
	2	651.935	1591.13
	3	800.364	1713.52
	4	698.283	2411.884
	5	719.992	2152.417
	6	734.373	2228.587
	7	602.28	2023.871
	8	701.394	1899.254
	9	591.293	2002.876
	10	671.552	1986.247

Tabla A 3. Resultados de la fase constructiva de los algoritmos para cada uno de los problemas de prueba

		TIPO I				TIPO II			
		GRASP1	GRASP2	GRASP3	**Cplex**	GRASP1	GRASP2	GRASP3	**Cplex**
1	m	7	16	8	**8**	10	15	8	**14**
	valor	1.65071	1.4933	1.9861	**1.9861**	1.4916	2.061533	2.19625	**2.2272**
	t	2.9704	1.4232	0.3120	**103.2160**	4.7928	1.1232	0.312	**152.8440**
2	m	10	12	9	**9**	20	21	14	**13**
	valor	1.5822	1.5644	1.8813	**1.8813**	2.03795	2.085619	2.673357	**2.6914**
	t	2.8456	1.0232	0.2864	**77.6090**	4.4552	1.024	0.3744	**128.5170**
3	m	10	5	9	**9**	11	16	11	**11**
	valor	1.0566	1.0902	1.5249	**1.5249**	1.8552727	1.60181	2.1556363	**2.1897**
	t	2.6832	1.2104	0.2744	**109.1350**	4.892	1.2736	0.3248	**237.6880**
4	m	16	19	15	**15**	13	10	8	**8**
	valor	1.85293	1.9488	2.2717	**2.2717**	1.6341538	1.6264	2.05375	**2.0538**
	t	3.0448	1.2104	0.4992	**77.2460**	4.9168	1.248	0.3368	**339.2080**
5	m	15	10	10	**9**	18	16	11	**11**
	valor	1.3601	1.4309	1.7094	**1.7237**	1.6590555	1.73143	2.279	**2.2790**
	t	2.8960	1.3848	0.2996	**137.1980**	5.504	1.0736	0.3368	**205.4410**
6	m	12	14	11	**11**	11	14	10	**10**
	valor	1.46533	1.597	1.8375	**1.8376**	1.7215454	1.522	2.0351	**2.0351**
	t	2.9824	1.0856	0.3368	**102.0440**	5.0672	1.1856	0.324	**235.1110**
7	m	8	10	8	**8**	10	10	10	**10**
	valor	0.91425	1	1.5293	**1.5293**	2.1554	2.0047	2.7655	**2.7655**
	t	2.3584	1.2480	0.2752	**128.4670**	5.1048	0.9856	0.3496	**120.9030**
8	m	14	14	11	**11**	9	17	12	**12**
	valor	1.5339	1.6707	1.9247	**1.9247**	2.430777	2.157	2.688416	**2.6884**
	t	3.0072	1.2608	0.3992	**133.8060**	4.9048	1.1728	0.3616	**120.4410**
9	m	13	14	14	**14**	14	11	9	**9**
	valor	1.62361	1.9855	2.2004	**2.2004**	1.852714	1.8098181	2.409111	**2.4177**
	t	2.9200	1.1736	0.4624	**81.1010**	4.8928	1.1232	0.3496	**142.0150**
10	m	9	17	13	**13**	15	11	10	**10**
	valor	1.6258	1.4918	1.8698	**1.8699**	1.79966666	1.9317272	2.48	**2.4800**
	T	3.3816	1.1984	0.3992	**73.2040**	4.268	1.2104	0.3744	**139.5930**

Tabla A 4. Resultados para cada uno de los ejemplos de prueba de tamaño $n = 30$ de los algoritmos incluyendo la fase de búsqueda local

		TIPO I				TIPO II			
		GRASP1	GRASP2	GRASP3	**Cplex**	GRASP1	GRASP2	GRASP3	**Cplex**
1	m	9	8	8	**8**	14	14	14	**14**
	valor	1.9243	1.9861	1.9861	**1.9861**	2.2272	2.2272	2.2272	**2.2272**
	t	7.3790	3.4480	0.3430	**103.2160**	8.3930	3.1350	0.3430	**152.8440**
2	m	9	10	9	**9**	13	14	13	**13**
	valor	1.8813	1.8238	1.8813	**1.8813**	2.6914	2.6419	2.6914	**2.6914**
	t	7.2700	2.9640	0.2970	**77.6090**	7.4570	3.2760	0.4050	**128.5170**
3	m	9	10	9	**9**	11	10	11	**11**
	valor	1.5249	1.4645	1.5249	**1.5249**	2.1165	2.1388	2.1897	**2.1897**
	t	7.9560	3.2610	0.2810	**109.1350**	8.0030	3.3700	0.3430	**237.6880**
4	m	15	15	15	**15**	12	13	8	**8**
	valor	2.2717	2.2717	2.2717	**2.2717**	1.9947	1.9839	2.0538	**2.0538**
	t	7.4720	3.5880	0.5310	**77.2460**	8.5340	3.0260	0.3270	**339.2080**
5	m	11	9	9	**9**	11	11	11	**11**
	valor	1.6894	1.7237	1.7237	**1.7237**	2.1649	2.2790	2.2790	**2.2790**
	t	7.9240	3.0890	0.3120	**137.1980**	8.8610	3.0890	0.3440	**205.4410**
6	m	10	13	12	**11**	11	11	10	**10**
	valor	1.8183	1.7920	1.8103	**1.8376**	1.9675	1.9382	2.0351	**2.0351**
	t	8.0650	3.3070	0.3740	**102.0440**	7.9870	2.9640	0.3120	**235.1110**
7	m	9	9	8	**8**	10	10	10	**10**
	valor	1.4282	1.4221	1.5293	**1.5293**	2.7655	2.7655	2.7655	**2.7655**
	t	8.9860	3.5100	0.2650	**128.4670**	7.8780	3.3070	0.3430	**120.9030**
8	m	9	11	11	**11**	11	12	12	**12**
	valor	1.8744	1.9247	1.9247	**1.9247**	2.6668	2.6884	2.6884	**2.6884**
	t	7.9250	3.2440	0.4210	**133.8060**	7.6130	3.3380	0.4370	**120.4410**
9	m	15	15	14	**14**	11	9	9	**9**
	valor	2.1398	2.1398	2.2004	**2.2004**	2.3823	2.4091	2.4177	**2.4177**
	t	8.4860	3.2760	0.8740	**81.1010**	8.5340	3.1830	0.3900	**142.0150**
10	m	13	13	13	**13**	10	10	10	**10**
	valor	1.8698	1.8698	1.8698	**1.8699**	2.4800	2.4800	2.4800	**2.4800**
	t	7.6280	3.4010	0.4370	**73.2040**	8.0970	3.0890	0.3900	**139.5930**

Tabla A 5. Resultados de la aplicación de la fase constructiva para problemas de prueba tipo I de tamaño $n = 150$.

	GRASP1a				GRASP1b				GRASP1c			
Inst.	TIEMPO	VALOR	n selecc	DESVIACIÓN	TIEMPO	VALOR	n selecc	DESVIACIÓN	TIEMPO	VALOR	n selecc	DESVIACIÓN
1	317.468	2.79586	53	34.93%	989.5424	3.23980	55	24.60%	108.6888	2.23584	14	47.96%
2	334.307	2.47113	70	41.42%	987.549	3.15843	66	25.13%	106.367	2.31407	16	45.14%
3	371.161	2.13576	45	44.40%	994.038	2.65330	46	30.93%	103.734	2.18614	15	43.09%
4	302.770	2.62960	89	36.96%	988.809	2.87371	72	31.11%	106.980	2.14645	12	48.54%
5	319.662	2.51959	34	36.59%	992.079	2.60394	53	34.46%	107.753	2.07540	16	47.77%
6	323.494	2.62185	47	37.15%	990.831	2.96849	52	28.84%	109.388	2.44144	17	41.47%
7	318.856	2.61228	72	41.44%	988.622	2.87100	56	35.64%	108.502	2.39933	19	46.21%
8	316.510	2.42956	43	42.01%	995.613	2.62547	50	37.34%	108.714	2.39350	11	42.87%
9	361.626	2.35541	44	42.93%	990.066	2.67726	38	35.14%	108.240	2.16836	12	47.47%
10	333.594	2.37633	51	40.67%	988.562	2.63194	81	34.29%	108.577	2.22357	15	44.48%
PROM	**329.945**	**2.49474**	**54.8**	**39.85%**	**990.571**	**2.83033**	**56.9**	**31.75%**	**107.694**	**2.25841**	**14.7**	**45.50%**

	GRASP2a				GRASP2b				GRASP2c			
Inst.	TIEMPO	VALOR	n selecc	DESVIACIÓN	TIEMPO	VALOR	n selecc	DESVIACIÓN	TIEMPO	VALOR	n selecc	DESVIACIÓN
1	26.8944	3.24724	77	24.42%	47.2744	3.45839	63	19.51%	12.6672	1.3793	5	67.90%
2	28.255	3.13883	60	25.59%	48.061	3.17874	38	24.65%	12.742	1.33600	6	68.33%
3	25.984	2.50463	62	34.80%	48.123	2.98315	60	22.34%	12.505	1.52333	12	60.34%
4	25.547	3.02109	89	27.58%	47.686	3.48798	61	16.39%	12.730	1.25140	5	70.00%
5	26.682	2.69958	43	32.06%	47.811	3.22903	67	18.73%	12.330	1.27843	7	67.82%
6	30.289	3.39994	53	18.50%	48.074	3.19311	71	23.45%	12.854	1.43400	7	65.62%
7	28.617	2.87975	71	35.44%	48.547	3.20851	53	28.07%	12.967	1.64650	10	63.09%
8	23.886	2.63935	82	37.01%	48.985	3.27310	71	21.88%	12.543	1.41622	9	66.20%
9	27.881	2.77048	61	32.88%	47.936	3.08351	70	25.29%	12.168	1.48240	5	64.09%
10	25.871	2.43746	84	39.14%	49.646	2.96264	66	26.03%	12.506	1.36000	5	66.04%
PROM	**26.991**	**2.87384**	**68.2**	**30.74%**	**48.214**	**3.20582**	**62**	**22.63%**	**12.601**	**1.41073**	**7.1**	**65.94%**

Instancia	GRASP3			
	TIEMPO	VALOR	n selecc	DESVIACIÓN
1	5.1544	42.9661905	42	0.00%
2	5.054	42.1850	40	0.00%
3	3.819	38.4147	43	0.00%
4	5.142	41.7148	56	0.00%
5	4.356	39.7330	30	0.00%
6	4.992	41.7155	42	0.00%
7	5.292	44.6083	48	0.00%
8	4.830	41.8984	37	0.00%
9	4.718	41.2755	44	0.00%
10	4.194	40.0517	36	0.00%
PROMEDIO	**4.755**	**41.4563**	**41.8**	**0.00%**

Tabla A 6. Resultados de la aplicación de la fase constructiva para los problemas de prueba tipo II de tamaño $n = 150$

	GRASP1a				GRASP1b				GRASP1c			
Instancia	TIEMPO	VALOR	n selecc	DESVIACIÓN	TIEMPO	VALOR	n selecc	DESVIACIÓN	TIEMPO	VALOR	n selecc	DESVIACIÓN
1	223.707	2.94842	55	45.62%	597.748	3.49423	46	35.56%	71.461	2.78958	13	48.55%
2	226.992	3.32532	56	40.36%	594.526	3.95843	53	29.01%	71.112	3.67230	21	34.14%
3	263.392	3.32714	73	41.71%	593.645	3.75018	60	34.30%	71.033	3.16392	14	44.57%
4	263.322	3.91521	67	28.68%	592.664	3.77275	62	31.27%	72.308	2.94318	12	46.38%
5	214.946	2.94556	72	43.11%	595.689	3.51387	29	32.14%	71.426	2.93993	16	43.22%
6	229.332	3.27565	62	37.16%	599.388	3.81120	73	26.89%	70.805	3.24570	11	37.74%
7	221.525	3.31257	68	40.87%	594.824	3.93831	71	29.70%	72.736	3.34250	17	40.34%
8	217.856	4.33641	46	20.85%	596.134	3.98443	76	27.27%	70.823	2.87373	12	47.55%
9	216.798	3.07063	60	46.04%	596.863	3.76512	68	33.84%	70.072	2.98350	13	47.58%
10	222.238	3.14412	57	44.20%	596.801	3.49838	60	37.92%	72.457	2.91200	17	48.32%
PROMEDIO	**230.011**	**3.36010**	**61.6**	**38.86%**	**595.828**	**3.74869**	**59.8**	**31.79%**	**71.423**	**3.08663**	**14.6**	**43.84%**

	GRASP2a				GRASP2b				GRASP2c			
Instancia	**TIEMPO**	**VALOR**	**n selecc**	**DESVIACIÓN**	**TIEMPO**	**VALOR**	**n selecc**	**DESVIACIÓN**	**TIEMPO**	**VALOR**	**n selecc**	**DESVIACIÓN**
1	25.996	3.66919	94	32.33%	48.6728	4.32046	63	20.32%	11.943	1.93087	8	64.39%
2	25.010	4.89143	72	12.28%	48.498	4.30183	65	22.85%	12.230	1.65683	6	70.29%
3	25.447	3.51423	74	38.43%	49.196	4.36236	59	23.57%	12.156	1.82483	6	68.03%
4	25.446	3.91235	91	28.73%	48.286	4.43712	80	19.17%	12.130	1.64138	8	70.10%
5	26.133	2.98451	76	42.36%	48.598	3.91691	68	24.36%	12.293	2.04522	9	60.50%
6	23.587	3.71144	85	28.80%	48.759	4.42913	64	15.04%	12.979	2.05800	8	60.52%
7	26.595	4.39348	73	21.58%	48.660	4.49265	88	19.81%	12.268	1.69700	6	69.71%
8	25.622	3.64989	95	33.38%	49.022	3.94986	81	27.91%	12.056	1.76071	7	67.86%
9	28.006	4.08430	61	28.23%	48.360	4.19789	76	26.24%	11.806	1.67000	7	70.66%
10	25.297	3.45503	86	38.69%	48.161	4.11439	46	26.98%	11.906	1.87771	7	66.68%
PROMEDIO	**25.714**	**3.82659**	**80.7**	**30.48%**	**48.621**	**4.25226**	**69**	**22.62%**	**12.177**	**1.81626**	**7.2**	**66.87%**

	GRASP3			
Instancia	**TIEMPO**	**VALOR**	**n selecc**	**DESVIACIÓN**
1	5.154	54.2211	45	0.00%
2	5.454	55.7605	40	0.00%
3	5.378	57.0763	43	0.00%
4	4.967	54.8938	34	0.00%
5	4.418	51.7803	31	0.00%
6	4.742	52.1293	42	0.00%
7	5.304	56.0234	53	0.00%
8	5.454	54.7873	37	0.00%
9	4.967	56.9103	36	0.00%
10	4.331	56.3490	40	0.00%
PROMEDIO	**5.017**	**54.9931**	**40.1**	**0.00%**

Tabla A 7. Resultados de la aplicación de los métodos GRASP en los ejemplos de prueba de tipo I con $n = 150$

	TIPO I											
	GRASP 1				GRASP 2				GRASP 3			
Instan	TIEMPO	VALOR	DESV	m	TIEMPO	VALOR	DESV	m	TIEMPO	VALOR	DESV	m
1	254.344	4.161	9.15%	51	63.336	4.121	10.02%	65	28.432	4.580	0.00%	51
2	217.116	3.966	8.36%	62	66.906	4.014	7.24%	58	29.400	4.328	0.00%	45
3	217.878	3.635	8.42%	34	66.594	3.636	8.41%	41	20.077	3.970	0.00%	41
4	218.289	4.068	6.39%	35	53.914	4.137	4.81%	49	29.273	4.346	0.00%	49
5	226.052	3.864	10.10%	50	62.626	4.037	6.09%	46	26.718	4.298	0.00%	45
6	192.854	3.919	10.27%	50	58.682	3.966	9.19%	36	26.704	4.367	0.00%	40
7	286.842	4.270	9.14%	52	59.892	4.265	9.26%	55	34.230	4.700	0.00%	49
8	276.521	3.814	10.08%	51	54.626	3.851	9.20%	37	24.556	4.241	0.00%	38
9	291.997	3.773	10.73%	60	66.194	3.699	12.48%	33	22.717	4.226	0.00%	41
10	234.314	3.608	13.69%	49	62.226	3.871	7.38%	37	21.523	4.180	0.00%	41
Prom	**241.620**	**3.907**	**9.63%**	**49.4**	**61.499**	**3.959**	**8.41%**	**45.7**	**26.363**	**4.323**	**0.00%**	**44**

Tabla A 8. Resultados individuales de la aplicación de los métodos GRASP en los ejemplos de prueba de tipo II con $n = 150$

	TIPO II											
	GRASP 1				GRASP 2				GRASP 3			
Instan	TIEMPO	VALOR	DESV	m	TIEMPO	VALOR	DESV	m	TIEMPO	VALOR	DESV	m
1	241.567	5.123	10.88%	53	82.992	5.261	8.48%	48	21.996	5.748	0.00%	50
2	220.848	4.899	14.84%	56	77.751	5.348	7.04%	55	24.804	5.753	0.00%	46
3	249.389	5.296	9.14%	59	86.924	5.572	4.39%	50	22.464	5.828	0.00%	45
4	250.499	4.939	13.80%	65	87.173	5.158	9.97%	63	23.385	5.729	0.00%	48
5	251.955	4.720	10.94%	39	83.055	4.848	8.53%	34	19.188	5.300	0.00%	35
6	245.839	4.322	21.77%	59	72.150	4.998	9.53%	60	22.698	5.525	0.00%	51
7	258.222	5.022	13.69%	58	75.318	5.414	6.95%	59	25.678	5.819	0.00%	50
8	250.669	5.100	10.69%	62	76.924	5.279	7.55%	62	24.383	5.710	0.00%	59
9	260.835	5.112	11.23%	50	81.105	5.376	6.65%	48	23.182	5.759	0.00%	37
10	270.495	4.899	13.58%	45	77.267	4.974	12.26%	43	20.966	5.669	0.00%	40
Prom	**250.031**	**4.943**	**13.06%**	**54.6**	**80.065**	**5.222**	**8.14%**	**52.2**	**22.874**	**5.684**	**0.00%**	**46.1**

Tabla A 9. Resultados individuales de la aplicación de todos los métodos en los ejemplos de prueba de tipo I para problemas grandes $n = 500$

Instancia	Característica	GRASP1	GRASP2	GRASP3	GRASP3+PR
1	TIEMPO	1387.59042	878.327205	685.576905	715.873
	VALOR	6.71160033	7.08387892	7.81308904	7.8605
	DESVIACION	14.62%	9.88%	0.60%	0.00%
	m	200	223	146	152
2	TIEMPO	1476.71348	940.697331	631.929	682.04091
	VALOR	6.86465478	6.99086096	7.5714	7.68734667
	DESVIACION	10.70%	9.06%	1.51%	0.00%
	m	154	187	145	150
3	TIEMPO	1251.00725	877.738302	570.558	667.66513
	VALOR	6.4260908	6.65051553	7.37127778	7.56914063
	DESVIACION	15.10%	12.14%	2.61%	0.00%
	m	179	161	108	128
4	TIEMPO	1217.5328	964.173672	834.621	647.187025
	VALOR	6.83388322	7.28929293	7.94525767	8.18058434
	DESVIACION	16.46%	10.90%	2.88%	0.00%
	m	202	198	163	166
5	TIEMPO	1236.99533	912.157764	787.586	683.154785
	VALOR	6.43838881	6.99495862	7.83191176	7.85695714
	DESVIACION	18.05%	10.97%	0.32%	0.00%
	m	139	145	136	140
6	TIEMPO	1304.13099	1047.15967	795.464	732.03865
	VALOR	6.8811459	7.17309554	7.93990909	7.96426282
	DESVIACION	13.60%	9.93%	0.31%	0.00%
	m	151	157	165	156
7	TIEMPO	1382.80777	987.905184	678.479	607.297
	VALOR	6.19245665	6.55618812	7.49202239	7.54989726
	DESVIACION	17.98%	13.16%	0.77%	0.00%
	m	148	101	134	146
8	TIEMPO	1418.04722	958.121764	803.249	666.426805
	VALOR	7.00625431	7.43283778	7.64362143	7.69836424
	DESVIACION	8.99%	3.45%	0.71%	0.00%
	m	127	139	140	151
9	TIEMPO	1213.89109	1048.21813	697.901	634.79019
	VALOR	6.77533381	7.00838833	7.53062931	7.57209449
	DESVIACION	10.52%	7.44%	0.55%	0.00%
	m	112	120	116	128
10	TIEMPO	1356.97182	893.716722	687.994085	849.254
	VALOR	6.6663552	6.98274483	7.99792199	8.03789051
	DESVIACION	17.06%	13.13%	0.50%	0.00%
	m	132	145	141	137
PROMEDIO	TIEMPO	1324.56882	950.821574	717.335799	688.572749
	VALOR	6.67961638	7.01627616	7.71370405	7.79770381
	DESVIACION	14.31%	10.01%	1.07%	0.00%
	m	154.4	157.6	139.4	145.4

Tabla A 10. Resultados individuales de la aplicación de todos los métodos en los ejemplos de prueba de tipo II para problemas grandes $n = 500$

Problema	Característica	GRASP1	GRASP2	GRASP3	GRASP3+PR
1	TIEMPO	739.293	638.84	733.4063	765.887
	VALOR	8.99381	9.43822111	10.6889583	10.8152545
	DESVIACION	16.84%	12.73%	1.17%	0.00%
	m	219	199	144	165
2	TIEMPO	801.3883	742.393	621.2305	655.828
	VALOR	8.56288	8.98597521	10.2057586	10.3287851
	DESVIACION	17.10%	13.00%	1.19%	0.00%
	m	129	121	145	121
3	TIEMPO	699.9362	715.81	698.0908	709.709
	VALOR	9.03848	9.45038503	10.4677239	10.6301714
	DESVIACION	14.97%	11.10%	1.53%	0.00%
	m	147	187	134	140
4	TIEMPO	753.3732	796.681	640.9986	725.483
	VALOR	8.99494	9.04863473	10.2932199	10.4618442
	DESVIACION	14.02%	13.51%	1.61%	0.00%
	m	298	167	141	154
5	TIEMPO	736.4732	702.457	691.4831	706.918
	VALOR	9.87388	9.47994253	10.3508681	10.3608188
	DESVIACION	4.70%	8.50%	0.10%	0.00%
	m	169	174	144	149
6	TIEMPO	751.2033	683.846	674.9754	713.19
	VALOR	9.42223	9.34205851	10.2211987	10.4813987
	DESVIACION	10.11%	10.87%	2.48%	0.00%
	m	207	188	156	158
7	TIEMPO	693.6111	719.414	644.8772	625.607
	VALOR	8.09283	9.39207921	10.2676565	10.4503378
	DESVIACION	22.56%	10.13%	1.75%	0.00%
	m	212	202	131	148
8	TIEMPO	703.4829	650.477	643.083	609.3897
	VALOR	8.46598	9.20002206	9.97980576	10.0021407
	DESVIACION	15.36%	8.02%	0.22%	0.00%
	m	149	136	139	135
9	TIEMPO	679.3293	641.741	644.6539	635.748
	VALOR	8.99383	9.40310526	10.3385597	10.4927769
	DESVIACION	14.29%	10.38%	1.47%	0.00%
	m	132	114	159	130
10	TIEMPO	809.874	790.8	631.4231	648.652
	VALOR	8.54321	8.94602791	10.142964	10.3497014
	DESVIACION	17.45%	13.56%	2.00%	0.00%
	m	199	215	139	144
PROMEDIO	TIEMPO	736.79645	708.2459	662.42219	679.64117
	VALOR	8.898207	9.26864515	10.2956713	10.437323
	DESVIACION	14.74%	11.18%	1.35%	0.00%
	m	186.1	170.3	143.2	144.4

Tabla A 11. Resultados del optimizador Evolver y GRASP3 para problemas pequeños *n* = 30

		n = 30					
		Tipo I			Tipo II		
		EVOLVER	desv	GRASP3	EVOLVER	desv	GRASP3
1	m	8		8	8		14
	valor	1.9861	0.00%	1.9861	2.1644	2.82%	2.2272
	t	40		0.343	40		0.343
2	m	10		9	13		13
	valor	1.8644	0.90%	1.8813	2.6914	0.00%	2.6914
	t	40		0.297	40		0.405
3	m	9		9	11		11
	valor	1.5249	0.00%	1.5249	2.1897	0.00%	2.1897
	t	40		0.281	40		0.343
4	m	14		15	13		8
	valor	2.2624	0.41%	2.2717	2.0494	0.21%	2.0538
	t	40		0.531	40		0.327
5	m	13		9	12		11
	valor	1.6530	4.10%	1.7237	2.1943	3.72%	2.279
	t	40		0.312	40		0.344
6	m	11		12	11		10
	valor	1.8375	0.00%	1.8376	1.9675	3.32%	2.0351
	t	40		0.374	40		0.312
7	m	8		8	10		10
	valor	1.5293	0.00%	1.5293	2.7655	0.00%	2.7655
	t	40		0.265	40		0.343
8	m	12		11	12		12
	valor	1.8810	2.27%	1.9247	2.4576	8.59%	2.6884
	t	40		0.421	40		0.437
9	m	15		14	14		9
	valor	2.1398	2.75%	2.2004	2.3624	2.29%	2.4177
	t	40		0.874	40		0.390
10	m	11		13	10		10
	valor	1.8688	0.05%	1.8698	2.48	0.00%	2.48
	t	40		0.437	40		0.390

Tabla A 12. Resultados del optimizador Evolver y GRASP3 para problemas medianos *n* = 150

		n = 150					
		Tipo I			Tipo II		
		EVOLVER	desv	GRASP3	EVOLVER	desv	GRASP3
1	m	75		51	63		50
	valor	3.3304	27.28%	4.58	3.9796	30.77%	5.748
	t	200		28.432	200		21.996
2	m	62		45	75		46
	valor	3.1258	27.78%	4.328	4.2339	26.40%	5.753
	t	200		29.400	200		24.804
3	m	58		41	64		45
	valor	2.9881	24.73%	3.97	4.1046	29.57%	5.828
	t	200		20.077	200		22.464
4	m	66		49	64		48
	valor	3.1308	27.96%	4.346	3.9700	30.70%	5.729
	t	200		29.273	200		23.385
5	m	69		45	60		35
	valor	3.0398	29.27%	4.298	3.8342	27.66%	5.3
	t	200		26.718	200		19.188
6	m	60		40	59		51
	valor	3.3567	23.13%	4.367	3.9317	28.84%	5.525
	t	200		26.704	200		22.698
7	m	64		49	69		50
	valor	3.5316	24.86%	4.7	4.3298	25.59%	5.819
	t	200		34.230	200		25.678
8	m	61		38	71		59
	valor	2.9713	29.94%	4.241	4.0808	28.53%	5.71
	t	200		24.556	200		24.383
9	m	67		41	69		37
	valor	3.1577	25.28%	4.226	4.0595	29.51%	5.759
	t	200		22.717	200		23.182
10	m	67		41	59		40
	valor	2.8926	30.80%	4.18	4.1377	27.01%	5.669
	t	200		21.523	200		20.966

Tabla A 13. Resultados del optimizador Evolver y GRASP3 para problemas grandes *n* = 500

		n = 500					
		Tipo I			Tipo II		
		EVOLVER	desv	GRASP3	EVOLVER	desv	GRASP3
1	m	212		146	225		144
	valor	5.2409	32.92%	7.8131	5.1484	51.83%	10.6890
	t	3500		685.577	3500		733.406
2	m	232		145	247		145
	valor	5.1352	32.18%	7.5714	4.5288	55.62%	10.2058
	t	3500		631.929	3500		621.231
3	m	196		108	241		134
	valor	4.9681	32.60%	7.37127778	5.0656	51.61%	10.4677
	t	3500		570.558	3500		698.091
4	m	230		163	267		141
	valor	5.4030	32.00%	7.9453	4.9679	51.74%	10.2932
	t	3500		834.621	3500		640.999
5	m	223		136	239		144
	valor	5.2951	32.39%	7.8319	5.0057	51.64%	10.3509
	t	3500		787.586	3500		691.483
6	m	221		165	255		156
	valor	5.3679	32.39%	7.9399	4.8469	52.58%	10.2212
	t	3500		795.464	3500		674.975
7	m	219		134	228		131
	valor	4.9697	33.67%	7.4920	4.8596	52.67%	10.2677
	t	3500		678.479	3500		644.877
8	m	212		140	256		139
	valor	5.2821	30.90%	7.6436	4.6043	53.86%	9.9798
	t	3500		803.249	3500		643.083
9	m	215		116	236		159
	valor	4.9806	33.86%	7.5306	5.0703	50.96%	10.3386
	t	3500		697.901	3500		644.654
10	m	213		141	241		139
	valor	5.5015	31.21%	7.9979	4.9448	51.25%	10.1430
	t	3500		687.994	3500		631.

ANEXO 2. Conceptos y Demostraciones

Breve revisión sobre complejidad computacional

La intención de esta sección es presentar una breve revisión de las nociones y conceptos relacionados con la NP-dureza, con el fin de comprender la naturaleza de los problemas abordados en esta investigación, y también dar una demostración formal del carácter $NP-duro$ de estos problemas.

De manera informal, un problema $\mathcal{P}_0$ se denomina $NP-duro$ si es al menos tan duro como otros problemas de cierta clase razonable de problemas. En lo que sigue se trata de formalizar esta definición, que se basa en las nociones de algoritmos factibles y problemas tratables.

A. Algoritmos factibles

La noción de $NP-duro$ está relacionada al hecho empíricamente conocido de que algunos algoritmos son factibles y otros no lo son. La factibilidad de un algoritmo está relacionada con el tiempo computacional requerido para su ejecución. Por ejemplo, si para alguna entrada x de longitud $len(x)=n$, un algoritmo requiere 2^n pasos, entonces para una entrada de tamaño mediano, digamos $n=200$, se necesitarán 2^{200} pasos, que no podría realizarse en un tiempo humano razonable.

Se denomina algoritmo de tiempo exponencial a cualquier algoritmo arbitrario cuyo tiempo de ejecución $t(n)$ con entradas de longitud n crece al menos como una función exponencial, es decir, para el que $\exists c>0, N\in\mathbb{N}, \forall n\geq N, t(n)\geq e^{c\,n}$. Como resultado de esto, son usualmente considerados no factibles. Aunque el hecho de que un algoritmo sea no factible no significa que nunca puede ser aplicado, simplemente indica que hay casos para los que el tiempo de ejecución será tan grande que no es práctico utilizarlos, aunque para ciertas entradas el algoritmo si podría ser útil.

Por otro lado, el algoritmo se denomina de tempo polinomial si el tiempo de ejecución crece sólo como un polinomio de n; es decir, si y solo si existe un polinomio $P(n)$ tal que para toda entrada x de longitud $len(x)$, el tiempo computacional $t_{\mathcal{U}}(x)$ del algoritmo $\mathcal{U}$

con la entrada x está acotado por $P\big(len(x)\big)$: $t_{\mathcal{U}}(x) \leq P\big(len(x)\big)$. Estos algoritmos suelen ser bastante factibles.

Como resultado de estas dos posibilidades, en la literatura sobre complejidad computacional se ha consensuado sobre la siguiente idea: Un algoritmo $\mathcal{U}$ se denomina factible si y solo si es de tiempo polinomial. En muchos casos prácticos, esta definición describe adecuadamente la idea intuitiva de factibilidad: los algoritmos de tiempo polinomial son usualmente factibles, y los algoritmos de tiempo no polinomial son usualmente no factibles. Sin embargo habrán casos en que esta intuición no funcione, por ejemplo si un algoritmo tiene un tiempo de ejecución $2^{500}n$, es claramente de tiempo polinomial, pero infactible. Pero a pesar de estos aparentes vacíos en esta formalización, por ahora "tiempo polinomial" es la mejor descripción conocida de factibilidad, por tanto establecemos la siguiente definición:

Definición 1. Un algoritmo $\mathcal{U}$ se denomina factible si existe un polinomio $P(n)$ tal que para toda entrada x de longitud $len(x)$, el tiempo computacional de ejecución del $t_{\mathcal{U}}(x)$ del algoritmo $\mathcal{U}$ está acotado por $P\big(len(x)\big)$, donde $len(x)$ representa la longitud de la entrada x, es decir el número de bits que forma esta entrada.

B. Problemas tratables

Una vez que se ha formalizado la definición de algoritmo factible, se puede describir qué es un problema manejable (o tratable) y que es un problema inmanejable (o intratable): Si existe un algoritmo de tiempo polinomial que resuelve todas las instancias de un problema, este problema se dice que es tratable, caso contrario se dice que es intratable.

Es decir, para decidir sobre el carácter del problema, o bien tenemos explícitamente un algoritmo de tiempo polinomial para resolver todas las instancias del problema, o tenemos una demostración de que no existe un algoritmo de ese tipo. Desafortunadamente, en muchos casos, no se conoce ni lo uno ni lo otro, o es sumamente complicado llegar a establecer una de las dos. Esto no significa que no haya solución al problema, pues en lugar de la falta de información ideal, tenemos otra información que es casi tan buena. Es decir, en algunos casos, no sabemos si el problema se puede resolver en tiempo polinomial o no, pero por ejemplo, sí podríamos saber que este problema es tan duro como pueden ser algunos problemas prácticos: entonces si pudiéramos solucionar este problema fácilmente, habría un algoritmo que resuelve todos los problemas con

facilidad, y la existencia de tal algoritmo es muy improbable, pues nuestra intuición nos indica que en la realidad hay problemas muy difíciles. A estos problemas "duros" se les denomina intratables.

Para formular esta definición en términos más formales, debemos primero describir lo que entendemos por un problema, y que significa reducir un problema a otro.

Para describir que es un problema práctico, supongamos que:

- Tenemos alguna información, representada por x, y
- Conocemos la relación $R(x, y)$ entre la información conocida x y el objeto deseado y

Suponemos que tanto la información x como el objeto deseado y se han representado como sucesiones binarias, esto siempre se puede hacer ya que en la computadora cualquier cosa puede ser representada como una sucesión binaria.

Por ejemplo si se tiene una sentencia matemática x, y el objeto deseado y es o bien una prueba de x, o una refutación de x. Aquí, $R(x, y)$ significa que y es una demostración cualquiera de x, o de "no x".

Para un problema que es práctico, debemos tener una manera de comprobar si la solución propuesta es correcta. En otras palabras, debemos suponer que existe un algoritmo factible que chequee $R(x, y)$ dados x e y. Si no existe tal algoritmo factible, entonces no existe un criterio para decidir si logramos una solución o no.

Otro requerimiento para un problema de la vida real es que en tales problemas, usualmente conocemos una cota superior para la longitud $len(y)$ de la descripción de y. En el ejemplo anterior: una demostración no debería ser demasiado grande, sino sería imposible comprobar si se trata de una prueba o no.

En todos los casos, es necesario que un usuario sea capaz de leer la solución deseada símbolo tras símbolo, y el tiempo requerido para la lectura debe ser factible. Anteriormente se formalizó "tiempo factible" como un tiempo que está acotado por algún polinomio de $len\,(x)$. El tiempo de lectura es proporcional a la longitud $len\,(y)$ de la respuesta y. Por lo tanto, el hecho que el tiempo de lectura es acotado por un polinomio de $len\,(x)$ significa que la longitud de la salida y está también acotada por algún

polinomio de $len\,(x)$. Es decir: $len\,(y) \leq P_L(len\,(x))$ para algún polinomio P_L. Así, se llega a la siguiente formulación de un problema práctico:

Definición: Se denomina problema práctico (o simplemente problema) a la pareja $\langle R, P_L \rangle$, donde $R(x, y)$ es un algoritmo factible que transforma dos sucesiones binarias en un valor Booleano, verdadero o falso, y P_L es un polinomio.

Definición: Se denomina instancia de un problema $\langle R, P_L \rangle$ el siguiente problema: Dada una sucesión binaria x, generar o bien y tal que $R(x, y)$ es verdadero y $len(y) \leq P_L(len\,(x))$, o, si tal y no existe, un mensaje diciendo que no hay solución. Por ejemplo, para el problema matemático descrito antes, una instancia puede ser: hallar una demostración o una refutación.

Lo que llamamos "problemas prácticos generales" es usualmente descrito como "problemas de la clase NP", para separarlos de aquellos problemas más complicados en los cuales la solución no es fácilmente verificable. Los problemas para los cuales existe un algoritmo factible que resuelve todas sus instancias se denomina tratable, fácil de resolver, o "problemas de la clase P". La opinión generalizada es que no todos los problemas (prácticos generales) deben tener una solución fácil (es decir, que $NP \neq P$), pero nunca ha sido probada.

Una manera de resolver un problema NP es chequear $R(x, y)$ para todas las sucesiones binarias y con $len(y) \leq P_L\big(len(x)\big)$. Este algoritmo, denominado algoritmo del Museo Británico, requiere $2^{P_L(len(x))}$ chequeos, es decir es de tiempo exponencial, y por tanto no factible.

C. Reducción de un problema a otro

Como ejemplo, supongamos que tenemos un algoritmo que chequea cuando un sistema dado de desigualdades lineales es consistente. Consideremos ahora otro problema, el de chequear cuando un sistema dado de desigualdades e igualdades es consistente. Es fácil determinar que éste último puede ser reducido al problema de la consistencia de un sistema de desigualdades, reemplazando cada igualdad por dos desigualdades.

En general, podemos decir que un problema $\mathcal{P} = \langle R, P_L \rangle$ puede reducirse al problema $\mathcal{P}' = \langle R', P_L' \rangle$ si existen tres algoritmos factibles U_1, U_2 y U_3 con las siguientes propiedades:

- El algoritmo factible U_1 transforma cada entrada x del primer problema en una entrada del segundo problema.
- El algoritmo factible U_2 transforma cada solución y del primer problema en la solución del caso correspondiente del segundo problema; es decir, si $R(x, y)$ es verdadero, entonces $R'\big(U_1(x), U_2(y)\big)$ es también verdadero.
- El algoritmo factible U_3 transforma cada solución y' de la correspondiente instancia del segundo problema en la solución del primer problema; es decir, si $R'(U_1(x), y')$ es verdadera, entonces $R\big(x, U_3(y')\big)$ también es verdadera.

Si existe una reducción, entonces una instancia x del primer problema es resoluble si y solo si la correspondiente instancia $U_1(x)$ del segundo problema es resoluble también. Por otra parte, si podemos resolver la segunda instancia (y encontrar una solución y'), entonces seremos capaces de encontrar una solución de la instancia original x del primer problema (como $U_3\ (y')$). Por lo tanto, si tenemos un algoritmo factible para resolver el segundo problema, sería posible diseñar un algoritmo para resolver el primer problema también.

Se puede ver fácilmente que la reducción es una relación transitiva: si un problema $\mathcal{P}$ puede reducirse a un problema $\mathcal{P}_0$, y el problema $\mathcal{P}_0$ se puede reducir a un problema $\mathcal{P}_1$, entonces, combinando estas dos reducciones, podemos demostrar que $\mathcal{P}$ se puede reducir a $\mathcal{P}_1$.

Definición: Un problema (no necesariamente de la clase NP) se denomina $NP - duro$ si cada problema de la clase NP puede ser reducida a él.

Definición: Si un problema de la clase NP es $NP - duro$, es llamado $NP - completo$.

Si un problema $\mathcal{P}$ es $NP - duro$, entonces cada algoritmo factible para resolver este problema $\mathcal{P}$ daría lugar a algoritmos factibles para resolver todos los problemas de la clase NP, lo cual se conjetura que es imposible. En vista de esta convicción, los problemas $NP - duros$ son denominados también intratables. Cabe señalar que aunque la mayoría de los científicos creen que los problemas difíciles no son factibles, todavía no se puede probar (o refutar) este hecho. Si un problema $NP - duro$ puede ser resuelto por un

algoritmo factible, entonces (por definición de $NP - dureza$) todos los problemas de la NP clase se podrían resolver por medio de algoritmos factibles y por lo tanto, $P = NP$. Viceversa si $P = NP$, entones todos los problemas de la clase NP, incluidos los problemas $NP - completos$, podrían ser resueltos por algoritmos de tiempo polinomial (factibles).

Pero si aceptamos la creencia común, que $P \neq NP$, entonces el hecho de que el problema es $NP - duro$ significa que no importa el algoritmo que usamos, siempre habrá casos en los que el tiempo de ejecución crece más rápido que cualquier polinomio. Por lo tanto, para estos casos, el problema es verdaderamente intratable.

D. Demostración del carácter $NP - duro$ de un problema

La demostración original del carácter $NP - duro$ de cierto problema $\mathcal{P}_0$ es bastante compleja, ya que se basa en demostrar explícitamente que todos los problemas de la clase NP se pueden reducir al problema $\mathcal{P}_0$. Sin embargo, una vez que se ha demostrado la $NP - dureza$ de un problema $\mathcal{P}_0$, la demostración de que otro problema $\mathcal{P}_1$es $NP - duro$ es mucho más fácil a partir del concepto de reducción, ya que para demostrar que $\mathcal{P}_1$es $NP - duro$, es suficiente demostrar que uno de los conocidos problemas $NP - duros$ puede ser reducido a este problema $\mathcal{P}_1$.

Propiedad: Si un problema $\mathcal{P}_0$ es $NP - duro$, cualquier problema más general $\mathcal{P}_1$también es $NP - duro$.

Demostración: El hecho de que $\mathcal{P}_0$ es $NP - duro$ implica que cada instancia $\mathcal{p}$ de cualquier problema $\mathcal{P}$ puede ser reducida a alguna instancia $\mathcal{p}_0$ del problema $\mathcal{P}_0$. Ya que el problema $\mathcal{P}_1$es más general que el problema $\mathcal{P}_0$, cada instancia $\mathcal{p}_0$ del problema $\mathcal{P}_0$ es también una instancia del problema más general $\mathcal{P}_1$. Así, cada instancia $\mathcal{p}$ de cualquier problema $\mathcal{P}$ puede ser reducida a alguna instancia $\mathcal{p}_0$ del problema $\mathcal{P}_1$, es decir el problema general $\mathcal{P}_1$ es $NP - duro$.