Inteligencia Artificial

Estado del Arte: The Progressive Party Problem

Pablo Ibarra S.

May 13, 2017

Evaluación

Resumen (5%):	_____
Introducción (5%):	_____
Definición del Problema (10%):	_____
Estado del Arte (35%):	_____
Modelo Matemático (20%):	_____
Conclusiones (20%):	_____
Bibliografía (5%):	_____
Nota Final (100%):	_____

Abstract

En este informe se presenta un estado del arte del problema conocido como *Progressive Party Problem*. El problema consiste en programar de la mejor manera una fiesta de yates, donde distintas personas visitaran varios yates anfitriones. Además, se definen términos y conceptos importantes utilizados en la literatura e investigaciones en el área. Se explica y define el problema a tratar, así como también se realizan descripciones y clasificaciones de las distintas variantes que se han desarrollado a través de los años. Se describen las distintas estrategias y algoritmos que se han desarrollado para la resolución del problema principal junto a sus variantes y se presentan también modelos matemáticos del problema.

1 Introducción

Hoy en día la programación de horarios es un tema vital para personas y organizaciones, debido a que si se realiza bien, los recursos que se disponen y utilizan, se distribuirán de manera eficiente e inteligente. La programación lineal entera a sido una herramienta clave durante mucho tiempo para resolver este tipo de problemas, aun así, existen ciertos problemas que PLE no puede resolver debido a la explosión combinatorial de estos problemas, un ejemplo de un problema que presenta estas características es el problema llamado *Progressive Party Problem* y será nuestro foco de estudio en el presente informe.

El problema conocido como Progressive Party Problem fue introducido por Peter Hubbard miembro de la asociación de propietarios SeaWych y del departamento de matemáticas de la universidad de Southampton, cuando tuvo que organizar una fiesta de yates en la isla Wight el años 1996, que se encuentra al sur de la costa de Inglaterra. El problema nos sitúa en el contexto de una fiesta de yates durante la tarde, donde la tripulación de cada yate debe interactuar socialmente con las otras tripulaciones.

El propósito de este informe es investigar sobre dicho problema, sobre los métodos que existen para solucionarlo, presentar antecedentes de lo que se a desarrollado, presentar hacia donde apuntan las nuevas investigaciones y presentar un modelo matemático del problema.

La motivación de este problema es debido a la importancia que tiene en problemas reales. Inicialmente se planteó como un problema para la organización de fiesta en yates, pero hoy en día su estudio puede ayudar para organizar de mejor manera ferias de libros, tours y otro tipo de eventos, optimizando los recursos de estos con el fin de maximizar el beneficio, por ejemplo, ganancias.

En este informe se presenta información relevante para poder introducirse al tema de *Progressive Party Problem*. Primero se comienza definiendo el problema a estudiar y se presentan otras variantes conocidas que existen del problema. Luego le sigue una sección centrada en el Estado del Arte del problema, donde se explican los métodos mas importantes que existen de resolución para este tipo de problema. Para aportar al estudio del problema este informe presenta, en otra sección, distintos modelos matemáticos que pueden ser usados para poder solucionar el problema mediante técnicas de optimización. Finalmente, el informe concluye con unas conclusiones respecto al tema y lo que puede venir en el futuro respecto a este.

2 Definición del problema

El problema *Progresive Party Problem* descrito por Peter Hubbard habla de un total de 39 botes que participan en la fiesta, algunos de estos son seleccionados para ser yates anfitriones, la tripulación de los yates anfitriones deben mantenerse en su bote ya que tienen que organizar la fiesta en sus barcos, mientras que las otras tripulaciones, que son conocidos como tripulación huésped, se pasean por los yates anfitriones socializando con el resto de las tripulaciones. La fiesta completa dura 3 horas, por lo que las tripulaciones huéspedes deben ir rotando cada 30 minutos para que socialicen lo que mas puedan. La tripulación de cada barco se mantiene junta y cada barco tiene una capacidad máxima. Por otro lado el barco del organizador de la fiesta siempre debe ser anfitrión, independiente de la capacidad del bote, esto es por que en ese barco estarán todos los elementos para enfrentar una eventual emergencia. Debido a que el organizador y dos tripulaciones mas (Estas tripulaciones deberán ser anfitriones y corresponden ser las tripulaciones del bote 1,2 y 3) tienen niños entre sus tripulaciones se crearon tres botes virtuales con capacidad 0, uno por cada anfitrión, esto se hizo debido a que solo los adultos son anfitriones de los yates designados como anfitriones, por lo tanto el problema final se consideran 42 botes. Finalmente lo que se busca en este problema es minimizar la cantidad de barcos anfitriones (debido a que se tienen que abastecer con comida) y asignar las tripulaciones huéspedes a cada yate anfitrión durante todos los intervalos de tiempo que dure la fiesta. La Data para el problema dado esta en la siguiente tabla.

Boat	Capacity	Crew	Boat	Capacity	Crew	Boat	Capacity	Crew
1	6	2	15	8	3	29	6	2
2	8	2	16	12	6	30	6	4
3	12	2	17	8	2	31	6	2
4	12	2	18	8	2	32	6	2
5	12	4	19	8	4	33	6	2
6	12	4	20	8	2	34	6	2
7	12	4	21	8	4	35	6	2
8	10	1	22	8	5	36	6	2
9	10	2	23	7	4	37	6	4
10	10	2	24	7	4	38	6	5
11	10	2	25	7	2	39	9	7
12	10	3	26	7	2	40	0	2
13	8	4	27	7	4	41	0	3
14	8	2	28	7	5	42	0	4

Table 1: The data

Por lo tanto identificamos 42 botes (tripulaciones), definimos $i, j, k \in \{1, .., 42\}$ y un conjunto de tiempos $t \in \{1, ..., 6\}$.

1. El tamaño de las tripulaciones estará dada por s_i, y la capacidad del bote - la máxima cantidad de personas que el bote puede recibir incluyendo los dueños del bote es de C_i.
2. El número máximo de invitados que puede cada bote invitar es de $c_i = \max(0, C_i\text{-}s_i)$.
3. Se introduce la variable binaria $x_{i,j,t}$. La variable $x_{i,j,t} = 1$ si la tripulación huésped j visita a la tripulación anfitriona i en el periodo de tiempo t. En todo otro caso $x_{i,j,t} = 0$.
4. Se utiliza otra variable binaria para decidir cuales botes son anfitriones. La variable $h_i = 1$ si el bote i es anfitrión, en el caso que el bote sea huésped $h_i = 0$.
5. Solo hay fiestas en los botes anfitriones, o, si una tripulación huésped visita a un bote i en algún periodo de tiempo, entonces el bote i es anfitrión.
6. La capacidad máxima de cada bote nunca se puede exceder.
7. No hay tripulaciones hibridas, esto quiere decir que una tripulación es anfitrión o es huésped.
8. Una tripulación j visita a otra tripulación i al menos una vez.
9. Los botes 1,2 y 3 tienen que ser anfitriones. (Barco del organizador y 2 barcos que tenían tripulación con niños).
10. Los botes 40,41 y 42 deben ser botes huéspedes con capacidad 0 (Barco virtual de niños).
11. Las tripulaciones huéspedes no pueden encontrarse más de una vez durante toda la fiesta, esta restricción genera una nueva variable $m_{j,k,t}$ donde toma valor 1 si una tripulación j y una tripulación k se encuentran en una fiesta en el periodo t.
12. Finalmente lo que se busca es minimizar el número de barcos anfitriones, seleccionar que barcos serán los anfitriones y asignar las tripulaciones huéspedes a los barcos anfitriones durante todo lo que dure la fiesta.

El problema en su estructura a permanecido igual durante todos estos años, las variaciones que se proponen son simplemente relajaciones del problema original.

The Uncapacitated Problem [6]

La primera variante que se intentó resolver de *Progressive Party Problem* fue resolver el problema sin considerar la capacidad de los barcos, esto quiere decir que un barco puede almacenar a todas las tripulaciones huéspedes en un solo periodo. Ahora si pensamos en más de un periodo, es fácil ver una cota inferior de barcos anfitriones que necesitaremos para la fiesta, si g tripulaciones huéspedes visitan el barco anfitrión i en el periodo 1, necesitaremos al menos g barcos anfitriones en el periodo 2 (Esto es debido a que no podemos volver a visitar el barco i y además las tripulaciones huéspedes no pueden volver a encontrarse). Por lo tanto, necesitaremos g +1 barcos anfitriones para que las tripulaciones del periodo 1 no se encuentren en el periodo 2, entonces claramente $g(g+1)$ es la cantidad máxima de tripulación huésped que podemos acomodar en los 2 periodos utilizando solo g+1 barcos anfitriones. Usando lo anterior, para una instancia de 6 anfitriones podemos acomodar 30 huéspedes, para una instancia de 7 anfitriones podemos acomodar 42 huéspedes. Para el caso en particular de 42 botes en total, 7 serán anfitriones (35 huéspedes). Ahora como en la vida real los barcos tienen una capacidad máxima, por lo tanto, este enfoque no es el más indicado.

A Lower Bound [6]

Considerar la capacidad de los barcos es muy importante si es que se quiere encontrar una solución aceptable para este problema, es por esto que se buscó otro enfoque. La segunda variante propuesta del problema, también una relajación, es resolver el problema considerando que no existen distintas tripulaciones huéspedes, si no que existe una única y gigante tripulación huésped. Lo que se busca con esto es encontrar cual es la cantidad de botes anfitriones que se necesitan para que quepan toda la tripulación huésped en el primer periodo. Lo que no se esta considerando acá es que los miembros de cada tripulación deban permanecer juntos.

Otras Variantes [6]

Otra variante del problema es resolver el problema como un CSP, es decir, encontrar una instanciación de las variables que cumpla las restricciones sin una función objetivo. Finalmente, la última variante que nombraremos será cambiar la función objetivo de tal manera de maximizar la cantidad de periodos, con el fin de que la fiesta dure más.

Estado del Arte

Como el problema data del año 1996, muchos autores han propuesto estrategias y algoritmos para resolver el Progressive Party Problem, algunos de estos metodos utilizados los presentaremos en esta sección

- Los primeros métodos [6] se comenzaron a proponer en 1996, el mismo autor del problema, Peter Hubbard y su equipo comenzaron a resolver dos variantes del problema del *Progressive Party Problem*, estas fueron las variantes *The Uncapacitated Problem* y *Lower Bound*. El fin de resolver estos dos problemas fue para encontrar una cota inferior sobre la cantidad de botes anfitriones que se necesitarían para cierta cantidad de tripulaciones huéspedes. Luego de haber resuelto los dos problemas anteriores, Peter y su equipo hicieron dos formulaciones usando programación lineal entera con el fin de resolver el problema.

  La primera formulación se utilizaron variables binarias para resolver un CSOP con el objetivo de minimizar la cantidad de barcos anfitriones, el número de restricciones de este modelamiento fue del orden de O(B^4T^2), donde B es el número de botes y T la cantidad de periodos, el numero variables fue moderado, alrededor de O(B^2T). Se corrieron pruebas utilizando 14 botes y 4 periodos de tiempo, el modelo resulto tener 379 variables y 18212 filas, el resultado fue obtenido después de 259 segundos utilizando XPRESSMO optimiser[ref] en una IBM 486 DX PC, en tan solo 816 iteraciones.

  La segunda formulación también se utilizó variables binarias, se hicieron algunos cambios en las restricciones, el objetivo de esta formulación fue el reducir la cantidad de restricciones, lográndose bajar a O(B^3T) pero con el costo de que aumentaron el número de variables a O(B^3T). Las mismas pruebas que se realizaron en la primera formulación se utilizaron con este segundo modelo, obteniendo los mismos resultados, pero en un tiempo menor.

  Finalmente, cuando se intentó resolver el problema completo utilizando las dos formulaciones anteriores, se falló con ambas modelos. Debido a lo anterior se decidió relajar el problema, fijando que el mínimo número de barcos anfitriones deberían estar entre 13 y 15 (Esto se basó en los resultados obtenidos de las cotas inferiores obtenidas en los problemas *the Uncapacitated Proble*m y *Lower Bound*), además se ignoró la restricción de que las tripulaciones huéspedes se puedan encontrar mas de una vez, la solución fue encontrada en 598 segundos. Al ver que esta heurística funcionaba, la de fijar un rango en la cantidad de botes anfitriones, se corrió un segundo intento, que fallo al no arrojar ningún resultado luego de 189 horas.

Luego de que los modelos de programación lineal fallaran, el equipo modelo el *Progressive Party Problem* como un CSP, se logra eliminar la función objetivo ya que previamente se definen quienes serán los 13 barcos anfitriones, se eligen 13 debido a la cota inferior que se encontró previamente cuando se pensó en una gran tripulación huésped. Este CSP fue resuelto usando una librería de C++ llamada ILOG Solver, la librería posee una implementación de backtracking con full lookahead. La ventaja que tiene este modelo de estilo CSP es la cantidad de variables y restricciones que estas tienen, ambas están alrededor de O(B^2T). Finalmente se logró encontrar una resultado utilizando 13 botes anfitriones, 29 huéspedes y 6 periodos en 27 minutos, también arrojo buenos resultados con 7 y 8 periodos.

Table 4. An optimal solution

	*T*1	*T*2	*T*3	*T*4	*T*5	*T*6
Host						
1	34,35	30	29,31	14,25	32,40	20,36
2	31,32,33	17,41	14,20,40	24,29	36,37	15,34
3	14,39	20,22,31	16,23	28,38	19,24,35	26,30,42
4	22,28	25,39	24,38	15,16	23,29,42	14,21,37
5	18,38	27,29,33	22,41	39	16,20	28,31
6	19,21	16,34	25,28	17,33,42	39	22,35
7	23,24	15,38	30,37	18,32,41	14,22	16,33
8	16,17	26,28,32	35,39	19,22	21,38	24,27
9	27,30	19,37	21,42	26,31, 34,40	15,28	38,41
10	37,42	14,23,36	19,27	21,30	26,33,41	17,18, 25,40
11	20,25, 26,29	24,42	15,17,32	27,35	18,30,34	19,23
12	15,41	21,40	18,26,36	20,23	17,27	39
13	36,40	18,35	33,34	37	25,31	29,32

Figure 1: Ejemplo de solución optima

- En 1997, Joachim P. Walser [7] fue uno de los pioneros en utilizar algoritmos de búsqueda local para resolver este problema, el algoritmo que lleva por nombre WSAT, realiza una búsqueda local usando Greedy con restricciones pseudo-booleanas, además viene con un mecanismo de historial y una lista tabú. Este algoritmo trabaja dentro del mundo de lo factible, se define un movimiento, y cada vez que se realiza este se va calculando un score, si este score disminuye nos vamos quedando con la nueva instanciación, cuando el score llega a 0 quiere decir que estamos en presencia de la solución óptima. En caso de que exista un empate entre score, se utiliza la lista historia para regresar a la instanciación mas antigua, obviamente ignorando las instanciaciones que se encuentran en la lista tabú. Para resolver el *Progressive Party Problem*, Walser se baso en el trabajo hecho por Peter Hubbard y su equipo, utilizo el mismo modelo CSP, utilizo los 13 botes propuestos por Peter Hubbard y además genero 5 instanciaciones de los 13 botes iniciales considerados como anfitriones(Asegurándose que la capacidad de los 13 botes sean suficiente para las tripulación huésped), además se modificaron ciertas restricciones del CSP original a restricciones del tipo Pseudo-Boleanas y finalmente, se fijó el tamaño de la lista tabú en 1. Se obtuvieron los siguientes resultados.

host boats	h	g	%cap	WSAT ($\mathcal{PB}$)
1–12,16	100	92	.92	2.9s
1–13 (orig)	98	94	.96	5.5s
1,3-13,19	96	92	.96	6.4s
3–13,25,26	98	94	.96	8.8s
1–11,19,21	95	93	.98	31.6s
1–9,16–19	93	91	.98	42.5s

La tabla anterior muestra los botes anfitriones seleccionados, la capacidad máxima de los botes, la cantidad total de huéspedes, porcentaje total de la capacidad utilizada y tiempo promedio de correr el algoritmo 20 veces, los resultados fueron considerablemente más bajo que el modelo propuesto por el trabajo de Peter Hubbard. Adicionalmente se resolvió el CSP original utilizando Oz, un lenguaje de restricción concurrente, con el fin de comparar con el Solver ILOG que se utilizó inicialmente para resolver el problema, el tiempo que demoro Oz fue de 8 minutos, en una maquina conocida como SPARCstation 20, en cambio, el Solver ILOG demoro 27 minutos, también se aprecia una disminución notable en el tiempo de resolución del problema.

- En 1999 Philippe Galinier junto a Jin-Kao Hao propusieron una tercera forma [2] de resolver este problema, su método de resolución se basa en búsqueda local, trabajando en el mundo de lo infactible resolviendo el CSP propuesto por Peter Hubbard y su equipo. Para resolver el *Progressive Party Problem*, ellos introducen un espacio de búsqueda, una función de costo y dos diferentes vecindarios, donde se experimentan con ambos utilizando distintas heurísticas. El espacio de búsqueda que ellos utilizan es de G x T, donde G es igual a 29 y T $\in$ (6,7,8,9,10), este espacio de búsqueda se justifica dado que el CSP ya están asignados los 13 barcos anfitriones, por lo tanto, debemos asignar las 29 tripulaciones en los t periodos que definamos. La función de costo que proponen es una función que castiga cada restricción incumplida, castigando más a las restricciones duras que las blandas. Los dos vecindarios que generan dependen de los dos movimientos que ellos proponen, el primer movimiento corresponde a cambiar el barco anfitrión de una tripulación huésped con conflicto, esto quiere decir que se irán probando distintos barcos anfitriones quedándose con el barco anfitrión que presente menos conflictos. El segundo movimiento corresponde a hacer un *swap* entre los barcos anfitriones asignados a dos tripulaciones huéspedes. Ya definido lo anterior, Galinier y Hao utilizan dos algoritmos para resolver el problema, uno de ellos es *taboo search* [3] y el segundo es conocido como metrópolis (Versión simplificada de *Simulated Annealing* [5]. Finalmente, se definen cuatro criterios de parada

  1. Si la función de costo es igual a 0.
  2. Si se alcanza el tiempo máximo establecido para que el algoritmo se ejecute.
  3. Si se alcanza el máximo numero de iteraciones.
  4. Si se alcanza el máximo numero de movimientos.

  La implementación fue realizada en C++ y ejecutada en una maquina conocida como Sun ULTRA 1. Los resultados se muestra en una tabla donde se compara el tiempo que demora en encontrar una solución utilizando ILP(*Integer Linear Programing*) y la resolución del CSP propuestos por Peter Hubbard vs los algoritmos de búsqueda local propuestos.

Problem	ILP	Publisher CSP resolution	LS
p6	fail	27 min	< 1 s.
p7	fail	28 min	<1 s.
p8	fail	fail	1 s.
p9	fail	fail	4 s.
p10	fail	fail	fail

Se corrieron los experimentos 20 veces, el tiempo promedio obtenido por los algoritmos de búsqueda local son muchos más bajos que los resueltos por ILOG Solver. Si bien los algoritmos pudieron resolver para 6,7,8 y 9 periodos, observamos que para 10 periodos el algoritmo falla en resolver el problema. Para finalizar observemos la siguiente tabla del costo entregado al intentar resolver el problema con 9 periodos de tiempo utilizando los algoritmos de búsqueda local.

	[0..5[	[5..10[	[10..15[	[15..20[	[20..25[	[25..30[	[30..35[
$\psi_1(\%)$	1.7	1.4	15.6	37.3	42.3	42.8	52.0
$\psi_2(\%)$	10.3	41.1	51.2	60.6	67.1	67.9	69.5

La tabla indica en cada columna el porcentaje de soluciones que tuvieron un costo dentro de ese intervalo a medida que el algoritmo fue iterando, es decir, en la primera columna nos dice que el 1.7% de los resultados obtenidos redujeron su costo a 0,1,2,3 o 4. Esta información es importante ya que nos permite comparar y observar que el segundo movimiento tiende a mejorar mucho más la solución que el primer movimiento. Es importante también notar que estos métodos encuentran óptimos globales como óptimos locales.

- En el año 2001, Erwin Kalvelagen, propuso un modelo [4] basado en programación lineal mixta, este enfoque logro que la programación lineal pudiese resolver el problema de manera completa utilizando 7 periodos. El modelo se testeo en un software llamado GAMS(textitSoftware especializado en resolver modelos matemáticos) sobre una maquina Cray C90. El modelo generado tuvo las siguientes características.

time stage	rows	cols	nz	disc. vars	obj	gen. time	sol. time
1	38830	2620	114130	1758	13	0.73	1.62
2	73270	3445	146371	1722	13	1.17	1.72
3	107710	4306	181672	1722	13	1.58	2.35
4	142150	5167	216973	1722	13	2.10	3.00
5	176590	6028	252274	1722	13	2.52	3.84
6	211030	6889	287575	1722	13	2.96	4.39
(7)	245470	7750	322876	1722	13	3.42	5.50

En la tabla anterior las filas corresponden ser a la cantidad de restricciones que tenemos, las columnas son la cantidad de variables, nz la cantidad de elementos que no son ceros, disc. vars corresponde a la cantidad de variables discretas, obj corresponde al optimo encontrado por la función objetivo, gen. time el tiempo que demora en generar el modelo y sol. time el tiempo que demora en encontrar una solución para el periodo t. Podemos observar de la tabla que la cantidad de variables y restricciones va aumentando a medida que pasan los periodos, en consecuencia, el tiempo que demora en encontrar una solución también va en aumento.

3 Modelo Matematico

En esta sección se nombraran los dos modelos mas importantes que tuvieron un gran protagonismo en el estado del arte

Modelo CSOP [4] [6]

El modelo matemático que se muestra a continuación es el CSOP propuesto por Peter Hubbard. Se identifican 42 botes (tripulaciones), definimos $i, j, k \in \{1, .., 42\}$ y un conjunto de tiempos $t \in \{1, ..., 6\}$.

1. El tamaño de las tripulaciones estará dada por s_i, y la capacidad del bote - la máxima cantidad de personas que el bote puede recibir incluyendo los dueños del bote es de C_i.

2. El número máximo de invitados que puede cada bote invitar es de $c_i = \max(0, C_i\text{-}s_i)$.

3. Se introduce la variable binaria $x_{i,j,t}$. La variable $x_{i,j,t} = 1$ si la tripulación huésped j visita a la tripulación anfitriona i en el periodo de tiempo t. En todo otro caso $x_{i,j,t} = 0$.

4. Se utiliza otra variable binaria para decidir cuales botes son anfitriones. La variable $h_i = 1$ si el bote i es anfitrion, en el caso que el bote sea huésped $h_i = 0$.

5. Solo hay fiestas en los botes anfitriones, o, si una tripulación huésped visita a un bote i en algún periodo de tiempo, entonces el bote i es anfitrión.

$$x_{i,j,t} \to h_i \ (x_{i,j,t} \leq h_i) \text{ para todo } i, j, t \ (i \neq j)$$

6. La capacidad máxima de cada bote nunca se puede exceder.

$$\sum_{j|i \neq j} s_j x_{i,j,t} \leq c_i h_i \text{ para todo } i, t$$

7. No hay tripulaciones hibridas, esto quiere decir que una tripulación es anfitrión o es huésped.

$$h_j + \sum_i x_{i,j,t} = 1 \text{ para todo } j, t$$

8. Una tripulación j visita a otra tripulación i al menos una vez.

$$\sum_t x_{i,j,t} \leq h_i \text{ para todo } i, j \ (i \neq j)$$

9. Los botes 1,2 y 3 tienen que ser anfitriones. (Barco del organizador y 2 barcos que tenían tripulación con niños).

$$h_i = 1 \text{ para todo } i = \{1, 2, 3\}$$

10. Los botes 40,41 y 42 deben ser botes huéspedes con capacidad 0 (Barco virtual de niños).

$$h_i = 0 \text{ para todo } i = \{40, 41, 42\}$$

11. Las tripulaciones huéspedes no pueden encontrarse mas de una vez durante toda la fiesta.

$$\sum_{j|i\neq j, i\neq k} (x_{i,j,t} \wedge x_{i,j,t}) \leq 1 \text{ para todo } j,k \ (j \neq k)$$

Desafortunadamente esta restricción es difícil de resolver. Por lo tanto se introduce una nueva variable $m_{j,k,t}$ donde toma valor 1 si una tripulación j y una tripulación k se encuentran en una fiesta en el periodo t. Por lo tanto, la restricción anterior se cambia por:

$$m_{j,k,t} \geq x_{i,j,t} + x_{i,k,t} \text{ - 1 para todo } i,j,k,t, \ (j < k, i \neq j, i \neq k)$$
$$\sum_{t} m_{j,k,t} \leq 1 \text{ para todo } j,k, \ (j < k)$$

La inecuación es interpretada como: Si la tripulación j y k se conocen en el barco i en t entonces es claro que $x_{i,j,t}$ y $x_{i,k,t}$ ambos son verdaderas, entonces $m_{j,k,t} \geq 1$ (De acuerdo a la primera restricción). Como la suma en todos los periodos de $m_{j,k,t}$ no puede exceder a 1 (De acuerdo a la segunda restricción), la condicion de j y k solo se conocen una vez es satisfecha. Por otro lado, si la tripulación huésped j y la tripulación huésped k nunca se conocen, entonces $m_{j,k,t}$ será cero (De acuerdo a la primera restricción), también será cierto que la segunda restricción se cumple.

12. Finalmente, lo que se busca es minimizar el número de barcos anfitriones, seleccionar que barcos serán los anfitriones y asignar las tripulaciones huéspedes a los barcos anfitriones durante todo lo que dure la fiesta.

$$\min \sum_{i} h_i$$

Si consideramos las 3 variables el espacio de búsqueda tiene un tamaño de:

$$2^{13} \cdot 2^{42 \cdot 39 \cdot 6} \cdot 2^{42 \cdot 42 \cdot 6}$$

Modelo CSP [6]

El modelo Matemático del CSP tiene la misma estructura que el modelo matemático del CSOP, con la diferencia de que no posee función objetivo y que el punto 9 cambia a:

$$h_i = 1 \text{ para todo } i = \{1,2,3,4,5,6,7,8,9,10,11,12,13\}$$

Esta restricción nos dice que los botes anfitriones se fijan a los primeros 13 de la lista, por lo que h_i deja de ser variable. El espacio de búsqueda del CSP considera solo 2 variables, y tiene un tamaño de:

$$2^{42 \cdot 39 \cdot 6} \cdot 2^{42 \cdot 42 \cdot 6}$$

4 Conclusiones

A simple vista el problema conocido como *Progressive Party Problem* puede no ser muy practico, aun así claramente se observa en este trabajo que a lo largo de los años una gran variedad de algoritmos, solvers y nuevas heurísticas han utilizado este problema para compararse entre ellos. Si bien no todas las técnicas apuntaban a resolver el problema original(CSOP), el problema

relajado (CSP) no dejo de ser un desafío. Para resolver estos problemas los investigadores intentaron atacarlo desde distintas perspectivas, unos prefirieron la programación lineal utilizando algún *solver*, otros algún algoritmo mas directo como *taboo search*, y la diferencia entre estos dos mundos era mucha, claramente observamos que los que preferían resolver el problema utilizando un solver, obtenían el optimo global en un tiempo muy grande, en cambio los que preferían alguna técnica mas directa, independientemente si usaban soluciones factibles o infactibles, el tiempo que demoran en obtener un resultado óptimo son mucho menores, por lo que se cree que para resolver el problema original para periodos mas arriba de 10, el camino de los algoritmos de búsqueda local se ve mas prometedor. Seria interesante para trabajos futuros intentar utilizar alguna de estas técnica de búsqueda local y adaptarla para resolver el problema con 10 o mas periodos, debido a que como se vio en el estado del arte, aun es un problema abierto.

5 Bibliografía

References

[1] S. C. Brailsford, P. M. Hubbard, B. M. Smith, and H. P. Williams. Organizing a social event—a difficult problem of combinatorial optimization. *Comput. Oper. Res.*, 23(9):845–856, 1996.

[2] Philippe Galinier and Jin-Kao Hao. Solving the progressive party problem by local search. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 443–456, 1999.

[3] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[4] Erwin Kalvelagen. On solving the progressive party problem as a {MIP}. *Computers & Operations Research*, 30(11):1713 – 1726, 2003.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[6] Barbara M. Smith, Sally C. Brailsford, Peter M. Hubbard, and H. Paul Williams. The progressive party problem: Integer linear programming and constraint programming compared. *Constraints*, 1(1):119–138, 1996.

[7] Joachim P. Walser. Solving linear pseudo-boolean constraint problems with local search. *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, pages 269–274, 1997.