PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL TRAVELING TOURNAMENT PROBLEM MEDIANTE TABU SEARCH

SEBASTIÁN IGNACIO LESEIGNEUR OSSA

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE

INGENIERO DE EJECUCIÓN EN INFORMÁTICA

SEPTIEMBRE DE 2011

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

RESOLUCIÓN DEL TRAVELING TOURNAMENT PROBLEM MEDIANTE TABU SEARCH

SEBASTIÁN IGNACIO LESEIGNEUR OSSA

Profesor Guía: **Rodrigo Alfaro Arancibia**

Carrera: **Ingeniería de Ejecución en Informática**

Septiembre de 2011

Dedicado a mi querida abuelita Marta Lisboa Armijo,
quien se encuentra en la gloria de nuestro Señor.

Resumen

El *Traveling Tournament Problem* es un problema de obtención de calendarios deportivos que debe cumplir con dos condiciones fundamentales: un patrón para partidos de ida y regreso y el tiempo que deben emplear los equipos para trasladarse durante la temporada, los que de forma conjunta deben ayudar a generar calendarios deportivos factibles. Existen distintas instancias de este problema con distinto número de equipos involucrados, los que al aumentar logran que el problema se convierta en un interesante desafío. Se introduce el problema, se describe la forma en que se modeló y para su resolución se utilizó el algoritmo meta heurístico *Tabu Search*.

Palabras-claves: Traveling Tournament Problem, búsqueda tabú, meta heurísticas, TTP, TS, vecindario, fixture, Cuadrados Latinos, Double Round Robin, Anagnostopoulos.

Abstract

The Traveling Tournament Problem is a timetableing problem that must meet two fundamental conditions: the home/away pattern feasibility, and the team travel during season, both should help to get feasible schedules. Several instances of this problem exists including different number of teams, which makes of this problem an interesting challenge when the number of teams is raised. It's introduced the problem, describe its modeling, and for its resolution was used a metaheuristhic algorithm known as Tabu Search.

Keywords: Traveling Tournament Problem, tabu search, meta heuristics, TTP, TS, neighborhood, fixture,Latin Squares, Double Round Robin, Anagnostopoulos.

Índice

1 Introducción 1
2 Objetivos del Proyecto 2
2.1 Objetivo General 2
2.2 Objetivos Específicos 2
3 Traveling Tournament Problem 3
3.1 Definición 4
3.2 Formulación 4
3.3 Estado del Arte para Traveling Tournament Problem 5
3.3.1 Técnicas Matemáticas 5
3.3.1.1 Branch and Bound y Relajación Lagrangiana 6
3.3.2 Tabu Search 6
3.3.3 Simulated Annealing 7
3.3.4 Algoritmos Genéticos 7
3.3.4.1 Inicios 7
3.3.4.2 Algoritmo Genético canónico 7
3.3.4.3 Codificación 9
3.3.4.4 Selección 9
3.3.4.5 Operadores Genéticos 10
3.3.4.6 Sustitución 12
3.3.4.7 Criterio de término 12
4 Tabu Search 14
4.1 Inicios 14
4.2 Definición 14
4.3 Formulación 15
4.3.1 Espacio de Búsqueda, Función de Costo y Estado Inicial 16
4.3.2 Vecindarios (Neighborhood) 16
5 Desarrollo 20
5.1 Decisiones de Diseño 20
5.2 Algoritmo Tabu Search para Resolver el TTP 21
5.2.1 Esquema del Algorimo Implementado 21
5.2.2 Espacio de Soluciones 22
5.2.3 Solución Inicial 23
5.2.4 Función Objetivo 23
5.2.5 Vecindarios 26
5.2.5.1 Intercambio Parcial de Rondas – IPR 26
5.2.5.2 Intercambio de Rondas – IR 30
5.2.5.3 Intercambio de Equipos – IE 32
5.2.5.4 Intercambio de Localias – IL 33
5.2.6 Manejo de la Lista Tabú 36
5.2.7 Optimizaciones Locales 36
5.2.8 Criterio de Aspiración 36
5.2.9 Intensificación 37

5.2.9.1 Disminución del tamaño de la lista Tabú 37
5.2.10 Diversificación 37
5.2.11 Criterio de Término 39
6 Diseño Experimento y Resultados 40
6.1 Representaciones 40
6.1.1 Formato Anagnostopoulos 40
6.1.2 Formato Trick 40
6.1.3 Formato Urrutia 41
6.1.4 Formato Di Gaspero 41
6.1.5 Cuadrados Latinos 41
6.2 Instancias 42
6.2.1 Instancias Circulares (CIRC) 42
6.2.2 Instancias de la National League (NL) 43
6.3 Resultados 43
6.3.1 Resultados Generales 43
6.3.2 Resultados Anagnostopoulos 45
6.3.3 Resultados Cuadrados Latinos 46
6.3.4 Comparación Resultados Ambas Notaciones 48
7 Conclusiones 50
8 Referencias 52
9 ANEXO A: Bibliografía Relacionada 53
10 ANEXO B: Fixtures Obtenidos 54

Lista de Figuras

Figura 3.1 Representación del recorrido y tablas de un fixture deportivo de 4 equipos 3
Figura 3.2 Transiciones entre dos generaciones consecutivas 8
Figura 4.1 Fred Glover 14
Figura 4.2 N1, SwapHomes 17
Figura 4.3 N2, SwapTeams 17
Figura 4.4 N3, SwapRounds 18
Figura 4.5 N4, SwapMatches 19
Figura 4.6 N5, SwapMatchRound 19
Figura 5.1 Diagrama algoritmo implementado 21
Figura 5.2 Diagrama algoritmo cálculo de la función objetivo 25
Figura 5.3 Encuentros seleccionados para aplicar vecindad IPR a un fixture *x* 26
Figura 5.4 Resultado de la aplicación de la vecindad IPR a un fixture *x* 27
Figura 5.5 Ejemplo de vecindad IPR 28
Figura 5.6 Diagrama algoritmo vecindad IPR 29
Figura 5.7 Representación de la aplicación de una vecindad IR a un fixture *x* 30
Figura 5.8 Diagrama algoritmo vecindad IR 31
Figura 5.9 Representación de la aplicación de una vecindad IE a un fixture *x* 32
Figura 5.10 Diagrama algoritmo vecindad IE 33
Figura 5.11 Representación de la aplicación de una vecindad IL a un fixture *x* 34
Figura 5.12 Diagrama algoritmo vecindad IL 35
Figura 5.13 Diversificación de soluciones en el espacio de búsqueda 37
Figura 5.14 Diagrama del algoritmo de diversificación 39
Figura 6.1 Gráfico comparativo tiempo promedio de ejecución instancias 45
Figura 6.2 Gráfico comparativo de resultados del algoritmo instancias NL 48
Figura 6.3 Gráfico comparativo de tiempos de ejecución del algoritmo instancias NL 48
Figura 6.4 Gráfico comparativo de resultados del algoritmo instancias CIRC 49
Figura 6.5 Gráfico comparativo de tiempos de ejecución del algoritmo instancias CIRC .. 49

Lista de Tablas

Tabla 6.1 Instancia NL4 utilizando el Formato Anagnostopoulos 40
Tabla 6.2 Instancia NL4 utilizando el Formato Trick 41
Tabla 6.3 Instancia NL4 utilizando el Formato Di Gaspero 41
Tabla 6.4 Instancia NL4 utilizando Cuadrados Latinos 42
Tabla 6.5 Tabla consolidada de resultados algoritmo TTP 44
Tabla 6.6 Tabla parámetros de ejecución algoritmo – Notación Anagnostopoulos 45
Tabla 6.7 Tabla resultados instancia NL – Notación Anagnostopoulos 46
Tabla 6.8 Tabla resultados instancia CIRC – Notación Anagnostopoulos 46
Tabla 6.9 Tabla parámetros de ejecución algoritmo – Notación Cuadrados Latinos 46
Tabla 6.10 Tabla resultados instancia NL – Notación Cuadrados Latinos 47
Tabla 6.11 Tabla resultados instancia CIRC – Notación Cuadrados Latinos 47
Tabla 10.1 Fixture instancia NL4 – Notación Anagnostopoulos 54
Tabla 10.2 Fixture instancia NL6 – Notación Anagnostopoulos 54
Tabla 10.3 Fixture instancia NL8 – Notación Anagnostopoulos 54
Tabla 10.4 Fixture instancia NL10 – Notación Anagnostopoulos 55
Tabla 10.5 Fixture instancia NL12 – Notación Anagnostopoulos 55
Tabla 10.6 Fixture instancia CIRC4 – Notación Anagnostopoulos 56
Tabla 10.7 Fixture instancia CIRC6 – Notación Anagnostopoulos 56
Tabla 10.8 Fixture instancia CIRC8 – Notación Anagnostopoulos 56
Tabla 10.9 Fixture instancia CIRC10 – Notación Anagnostopoulos 57
Tabla 10.10 Fixture instancia CIRC12 – Notación Anagnostopoulos 57
Tabla 10.11 Fixture instancia NL4 – Notación Cuadrados Latinos 58
Tabla 10.12 Fixture instancia NL6 – Notación Cuadrados Latinos 58
Tabla 10.13 Fixture instancia NL8 – Notación Cuadrados Latinos 59
Tabla 10.14 Fixture instancia NL10 – Notación Cuadrados Latinos 59
Tabla 10.15 Fixture instancia NL12 – Notación Cuadrados Latinos 60
Tabla 10.16 Fixture instancia CIRC4 – Notación Cuadrados Latinos 60
Tabla 10.17 Fixture instancia CIRC6 – Notación Cuadrados Latinos 60
Tabla 10.18 Fixture instancia CIRC8 – Notación Cuadrados Latinos 61
Tabla 10.19 Fixture instancia CIRC10 – Notación Cuadrados Latinos 61
Tabla 10.20 Fixture instancia CIRC12 – Notación Cuadrados Latinos 62

1 Introducción

El problema de la optimización de viajes y calendarios para las temporadas deportivas es de vital importancia económica para la organización de torneos por parte de las diferentes federaciones deportivas, las que deben considerar la cantidad de viajes a realizar, el número de equipos que participarán y algunas restricciones relacionadas con la cantidad de partidos consecutivos que los equipos juegan de local o visita, entre otras.

El problema descrito es conocido en la literatura como *Traveling Tournament Problem* (TTP), que es planteado como la optimización de los calendarios de juego entre equipos T_i y T_j en que el par *(i, j)* representa una lista de equipos desordenada. Es un problema de *Round Robin* (cada equipo juega contra todos los demás), que puede ser representado como *Double Round Robin* (DRR) cuando se desea que los equipos se enfrenten dos veces, el encuentro de ida y la revancha.

Utilizando la cantidad de equipos y las distancias entre las localidades en que reside cada equipo, se utilizan métodos para lograr obtener un calendario que minimice los viajes de todos los equipos involucrados. Para poder solucionar este problema se pueden utilizar métodos directos (completos) o indirectos (incompletos), los que tienen como objetivo obtener el mejor valor de todos los viajes realizados durante la temporada. El trabajo se centró en la investigación del método indirecto *Tabu Search* para obtener soluciones del problema TTP, el que utiliza espacios de búsqueda acotados para encontrar las soluciones óptimas del problema, sin realizar cálculos directos como lo hacen los métodos completos.

Por ello en este trabajo se busca estudiar e implementar un modelo que solucione el problema del TTP mediante *Tabu Search*, para así poder demostrar la aplicación del estado del arte y tener la posibilidad de encontrar algún valor que pudiese equiparar o mejorar los existentes actualmente en la literatura.

Este trabajo comienza por el capítulo 2, donde se abordan los objetivos que serán tratados y profundizados en el marco teórico definido en los capítulos 3 y 4, los que detallan el problema a analizar y su estado del arte respecto a las distintas soluciones que se han propuesto. En el capítulo 5 se desarrolla en detalle el algoritmo a implementar para resolver el problema del *Traveling Tournament Problem* y se indican decisiones de diseño, asociadas al desarrollo de este trabajo. El capítulo 6 está enfocado a mostrar el diseño del experimento y sus resultados. Finalmente, los capítulos 7, 8 y 9 corresponden a las conclusiones, referencias bibliográficas y bibliografía relacionada, respectivamente.

2 Objetivos del Proyecto

2.1 Objetivo General

Desarrollar un modelo que sea capaz de obtener soluciones para el *Traveling Tournament Problem* (TTP) utilizando la técnica metaheurística *Tabu Search* (TS).

2.2 Objetivos Específicos

- Recolectar y revisar bibliografía del problema TTP y métodos de solución indirectos.
- Seleccionar el método a utilizar para solucionar el problema y, utilizando la literatura relacionada, realizar el análisis de las propuestas de implementación para dar solución al TTP.
- Realizar pruebas del programa de experimentación utilizando datos de entrada apropiados y formalmente descritos (obtenidos de problemas planteados en la literatura), junto con la presentación de los resultados obtenidos.
- Realizar ajustes necesarios al modelo implementado.
- Proponer e implementar mejoras al modelo desarrollado.
- Presentar un modelo final ajustado que contenga las mejoras propuestas y, que además resuelva el problema del TTP.

3 Traveling Tournament Problem

Según [3], en [4] Easton, Nemhauser y Trick propusieron un nuevo problema, denominado *Traveling Tournament Problem* (TTP) el cual abstrae algunos elementos clave de la confección de calendarios de encuentros deportivos combinado condiciones de *Home Away Pattern* (HAP) y la minimización de las distancias recorridas por los equipos durante la temporada.

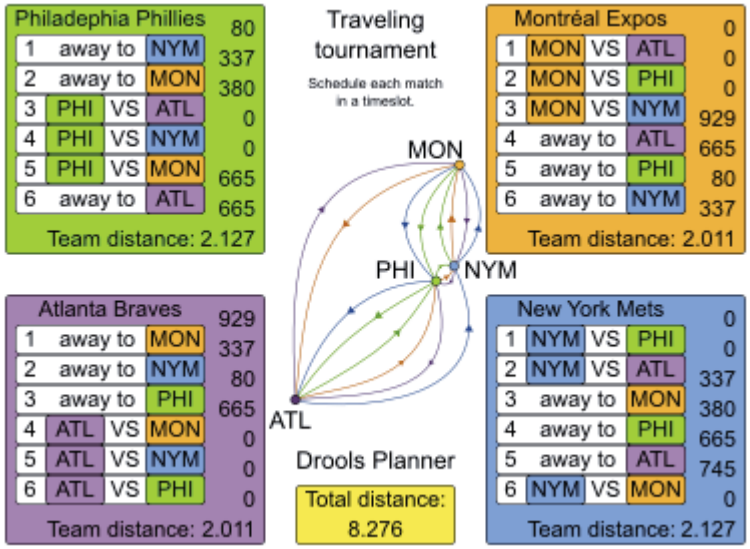

Figura 3.1 Representación del recorrido y tablas de un fixture deportivo de 4 equipos

El problema se enuncia de la siguiente manera: dada una matriz de distancias D de tamaño $n\ x\ n$, el TTP consiste en obtener un calendario de encuentros tipo *Double Round Robin* (DRR) correspondiente a $2\ (n\ -\ 1)$ rondas para n equipos con las siguientes características [5]:

- Ningún par de equipos puede enfrentarse en dos rondas consecutivas.
- Ningún equipo puede jugar más de tres partidos consecutivos de visitante ni de local.
- Se asume que los equipos comienzan en su lugar de origen y deben retornar al mismo al final del torneo.
- Cuando un equipo juega varios partidos seguidos de visitante, viaja de una localidad rival a otra directamente sin pasar por su lugar de origen entre cada viaje [3].

El objetivo del problema es minimizar la distancia total recorrida por todos los equipos durante el torneo, asumiendo la última condición en que cada equipo comienza en su localidad y retorna a ella al final de la temporada [5].

En sus comienzos el TTP fue pensado como un problema de referencia por [4] para iniciar la investigación de la estructuración de campeonatos deportivos. Siendo un problema del tipo NP-Completo ha resultado interesante para diversos campos de estudio y un buen desafío al incluir dos ingredientes fundamentales mezclados en un único problema: factibilidad y optimalidad.

Finalmente en [3] se indican diversas técnicas utilizadas para resolver este problema, entre las que se destacan la utilizada por Easton, Nemhauser y Trick quienes utilizaron un enfoque de programación entera combinado para resolver el problema, conocido como *Constraint Programming* y Programación Entera. Benoist, Laburthe y Rottembourgh que combinan *Constraint Programming* y Relajación Lagrangriana. Crauwels y Oudheusden utilizaron la meta heurística conocida como Colonia de Hormigas. Xingwen Zhang obtuvo resultados interesantes aplicando una combinación de *Constraint Programming*, *Simulated Annealing* y técnicas de *Hill-Climbing*.

3.1 Definición

Según [4] dados n equipos, n par, un torneo en el cual cada equipo juega contra otros en un torneo *Round Robin*. Ése tipo de torneo tiene $n - 1$ casillas en las que se disputan $n / 2$ juegos. Para cada juego un equipo es local y su oponente es el visitante. Como indica el nombre del equipo local, el encuentro se disputará en su localidad (lo que difiere de otras competencias en que ambos equipos viajan a un lugar determinado como visitantes). Un torneo del tipo *Double Round Robin* (DRR) tiene $2(n - 1)$ casillas y cada par de equipos se enfrenta dos veces alternando su calidad de local y visitante.

Las distancias entre las localidades de los equipos se encuentran representadas en una matriz *D* (matriz de distancias) cuyas dimensiones son $n\ x\ n$ (n corresponde a la cantidad de equipos). Cuando un equipo juega de visitante, se asume que viaja desde su localidad inicial a la localidad del otro equipo. En cambio sí se juegan partidos consecutivos de visita, el viaje se realiza desde la localidad rival en la que se disputó el último encuentro a la próxima localidad rival. Mientras que cada equipo inicia la temporada desde su localidad, a la que debe regresar al final de ésta.

Por lo tanto para los partidos jugados de visita se consideran los desplazamientos realizados por los equipos, en cambio cuando estos son jugados de local, no se considera desplazamiento para el equipo que recibe a su rival. El total del desplazamiento del torneo es la suma de los desplazamientos de todos los equipos durante la temporarda, incluyendo los retornos de los equipos visitantes a sus localias en la última ronda.

3.2 Formulación

El Traveling Tournament Problem se describe a continuación [4]:

Entrada: El número de equipos, *n*; una matriz de distancias $n\ x\ n$, *D*; los siguientes parámetros enteros, *L*, *U*.

Salida: Una matriz *Double Round Robin* (DRR) con el orden de los encuentros para toda la temporada.

Sujeto a las siguientes restricciones:

- La cantidad de viajes de visita o estadías de local es un número entre L y U inclusive, y
- La distancia total de los viajes de todos los equipos es minimizada.

Además de las restricciones básicas, hay requisitos adicionales en las soluciones, los que incluyen:

- Espejado, lo que implica que el torneo *Round Robin* ocurre sólo en las primeras $n-1$ rondas, las que luego en las siguientes $n-1$ se juegan con todas las localidades reversas, es decir que los locales juegan de visita y viceversa.
- No hay repeticiones, lo que implica la restricción que dos equipos que se enfrentan en una ronda no pueden enfrentarse en la ronda siguiente.

Los parámetros L y U hacen referencia a la cantidad de partidos consecutivos que un equipo tiene que jugar de local y como visitante. Para $L = 1$ y $U = n$ se produce el caso que el equipo se mantiene en viaje durante todo el torneo, lo que es equivalente al problema del TSP (*Traveling Salesman Problem*), en cambio si tenemos un valor muy pequeño para U, el equipo deberá retornar muy seguido a su origen, lo que origina que cada partido de visita está prácticamente intercalado con los de local.

3.3 Estado del Arte para Traveling Tournament Problem

Un gran número de aproximaciones al modelado y solución de TTP han sido señalados en la literatura, con distinto grado de éxito. Las aproximaciones giran en torno de una serie de avances tecnológicos que han ocurrido en los últimos años. Estas incluyen programación matemática, T*abu Search*, *Simulated Annealing*, algoritmos genéticos, entre otros. En esta sección se describirán de manera somera algunas técnicas destacadas en cada área separándolas según su naturaleza.

3.3.1 Técnicas Matemáticas

La programación matemática ha sido aplicada extensamente a problemas de optimización. Estos han sido formulados utilizando programación entera, programación entera mixta y programación dinámica. Hasta hace poco el uso de estas aproximaciones se encontraba limitada por el hecho que el problema pertenece a la clase NP-Completo. Para superar esta diferencia, se ha propuesto la descomposición del problema en sub-problemas, planteando un número de técnicas para resolverlos. En adición, nuevas técnicas de solución, heurísticas más poderosas y el aumento del poder computacional han posibilitado que estas aproximaciones sean utilizadas en problemas de mayores dimensiones. Aun así las dificultades

en la formulación de las restricciones de flujo de material como desigualdades matemáticas y el desarrollo de soluciones de software han limitado el uso de estas técnicas.

3.3.1.1 Branch and Bound y Relajación Lagrangiana

La técnica de *branch and bound* es una técnica enumerativa cuya idea principal es conceptualizar el problema como un árbol de decisión. Cada punto de decisión –un nodo- corresponde a una solución parcial. De cada nodo surge un número de nuevas ramas, una por cada posible decisión. El proceso continúa hasta que los nodos hoja, que no generan más ramas, son alcanzados.

A pesar que métodos eficientes de *branch and bound* han sido desarrollados para mejorar la velocidad de búsqueda, sigue siendo un procedimiento intensivo computacionalmente para problemas de gran envergadura.

La técnica llamada relajación Lagrangiana, que se ha utilizado por más de treinta años, considera que si una restricción es considerada un problema para la resolución, entonces por qué mejor no removerla. La relajación Lagrangiana resuelve problemas de programación entera por medio de la omisión de restricciones de valor entero específicas y agrega los correspondientes costos (debido a las omisiones o relajaciones) a la función objetivo.

Al igual que *branch and bound*, esta técnica es computacionalmente intensiva para problemas de gran tamaño.

3.3.2 Tabu Search

Tabu Search es una técnica de optimización iterativa, que consiste en un procedimiento que restringe la búsqueda y utiliza una función de memoria de corto plazo que convierte en prohibidos los t movimientos más recientes.

Este algoritmo aplicado a TTP genera vecinos a través de la inversión de arcos que unen tareas adyacentes en la ruta crítica. Después que un arco ha sido invertido se introduce su inverso a la lista tabú, de longitud definida que contiene los desplazamientos prohibidos. En cada iteración se evalúan todas las soluciones del vecindario que no se encuentren en la lista tabú, calculando para cada una el valor de $C_{máx}$ y seleccionando la más prometedora.

El algoritmo de tabú search más avanzado es llamado i-TSAB, que fue introducido por Nowicki y Smutnicki el 2001, que se basa en el previamente exitoso TSAB de los mismos autores. El algoritmo se caracteriza por sus componentes: un altamente restrictivo operador de movimiento y re-intensificación de la búsqueda alrededor de soluciones de alta calidad. A pesar de su efectividad, no se tiene completa certeza de cómo estos componentes y otros de carácter secundario interactúan para alcanzar el rendimiento que lo pone a la cabeza entre los algoritmos de búsqueda tabú.

3.3.3 Simulated Annealing

La denominación del algoritmo proviene de la analogía con el proceso de templado utilizado en metalurgia. Consistiendo este en el enfriamiento gradual del metal fundido, de modo que sus moléculas adoptan poco a poco una configuración de mínima energía. Al iniciarse el proceso, las moléculas vibran y se mueven caóticamente adoptando diversos tipos de configuraciones en la estructura del metal. A medida que la temperatura decrece, el movimiento se ralentiza y las moléculas tienden a adoptar de manera gradual las configuraciones de menor energía, siendo nula en el cero absoluto.

Simulated Annealing intenta realizar un proceso similar de manera numérica, donde el espacio de configuraciones viene dado por la secuencia de tareas que se desea procesar y la energía es un papel asumido por la función objetivo.

3.3.4 Algoritmos Genéticos

3.3.4.1 Inicios

Los Algoritmos Genéticos fueron creados y desarrollados por John Holland y sus estudiantes de la Universidad de Michigan en los 60s. Más que un tipo de algoritmo que permitiera resolver problemas específicos, la intención de Holland era estudiar el fenómeno de la adaptación natural y desarrollar vías por la cuales se pudieran llevar estos mecanismos a sistemas computacionales. Presentó los Algoritmos Genéticos como una abstracción de la evolución biológica y entregó un marco de trabajo teórico para la adaptación natural bajo los Algoritmos Genéticos.

El algoritmo propuesto por Holland se puede describir como un método para moverse desde una población de *cromosomas* a una nueva población, usando cierto tipo de selección natural por medio del uso de operadores inspirados en la genética, como son el cruzamiento (*crossover*), mutación (*mutation*) e inversión (*inversion*). Cada *cromosoma* está compuesto de *genes* (por ejemplo, un bit en el caso de representación binaria), los que a su vez pueden ser una instancia particular de un *allele* (por ejemplo, el valor particular: 0 o 1). Un operador de selección escoge en la población los cromosomas considerados más aptos para reproducirse, logrando que en promedio los cromosomas con un valor más alto de *fitness* (calidad) generen nuevos individuos para la población, en contraste con los menos aptos. El cruzamiento intercambia partes de un cromosoma, imitando de cierto modo la recombinación biológica entre dos organismos cromosómicos simples. La mutación cambia de forma aleatoria el *allele* de ciertas partes de los cromosomas. La inversión revierte el orden de una sección continua de un cromosoma, cambiando el orden de los genes respectivos.

3.3.4.2 Algoritmo Genético canónico

En esta sección se hará una descripción general del funcionamiento del denominado Algoritmo Genético canónico. La primera etapa del algoritmo consiste en generar una población inicial (generalmente de forma aleatoria) de individuos, que corresponden a los *cromosomas* o *genotipos*. Cada uno de ellos es evaluado y se le asigna una medida de *calidad* (*fitness*) determinada. En este contexto, la función de evaluación corresponde a una medida de

comportamiento que se obtiene en base a una serie de parámetros, mientras que la función de calidad toma dicho valor para transformarlo en una medida que representa posibilidades de reproducción. La función de evaluación es considerada independientemente para cada individuo, no así la de calidad.

Un Algoritmo Genético puede verse como un proceso con dos etapas. La primera inicia con la población actual y tras un proceso de selección se genera una población intermedia, con lo que se inicia la segunda etapa en la cual se realizan sobre la población intermedia distintas operaciones de *cruzamiento* (*o recombinación*) y *mutación*, obteniendo de ese modo la siguiente población. Una generación es el paso desde una población a otra tras el proceso descrito. Esto puede apreciarse gráficamente en la Ilustración 3.2.

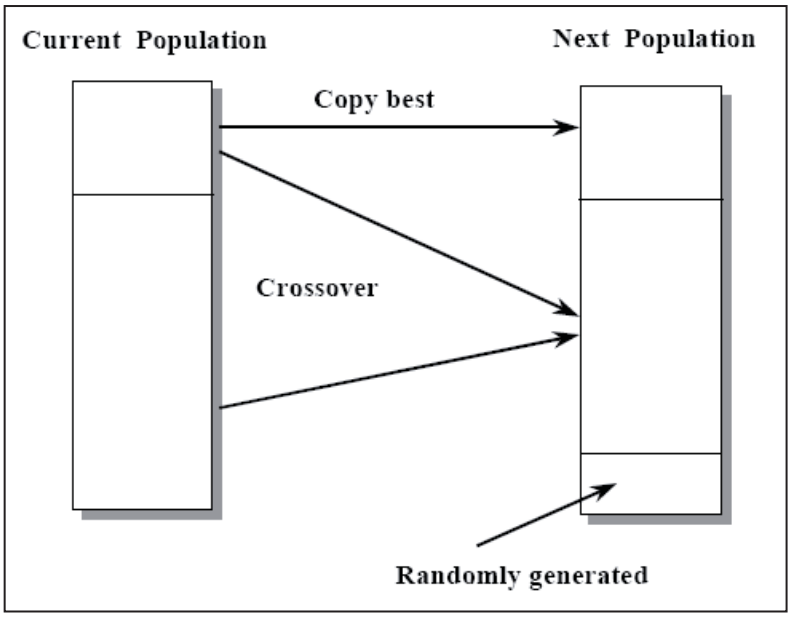

Figura 3.2 Transiciones entre dos generaciones consecutivas

En el Algoritmo Genético canónico, la calidad de un individuo se calcula con la expresión f_i/F, donde f_i corresponde al resultado de la evaluación y F corresponde al promedio de los resultados de evaluación en la población actual. Con esta calidad calculada, se realiza la selección considerando que la posibilidad de que un individuo pase a la generación intermedia sea proporcional a su calidad.

Con la población intermedia mezclada adecuadamente, se procede a realizar la recombinación. La operación de cruzamiento se realiza sobre un par de cromosomas elegidos de forma aleatoria de la población intermedia, con una probabilidad P_c. Supóngase que se tienen los cromosomas:

1101|010111

0100|011010

Utilizando cruzamiento en un punto (*One-Point Crossover*), para este ejemplo, en la posición 5, se obtienen dos nuevos cromosomas, que constituirán la nueva generación:

Hijo 1: **1101|011010**

Hijo 2: **0100|010111**

Ya realizada la recombinación, se puede realizar la mutación. Para cada bit en cada elemento de la nueva población, hay una probabilidad P_m de que ocurra mutación, lo que generalmente se traduce en cambiar el bit de 0 a 1 o viceversa.

Con el proceso completo se obtiene la nueva generación, lista para ser evaluada y volver a empezar las etapas de selección, recombinación y mutación.

3.3.4.3 Codificación

Un elemento fundamental en la construcción de Algoritmos Genéticos es la codificación que se utilice. El cromosoma, más que un arreglo de bits, funciona como la representación de una solución para el problema que está abordando el Algoritmo Genético. Así como se formula un mecanismo para llevar la información de parámetros, variables o cualquier elemento que tiene alguna relación con el problema en cuestión, debe existir un mecanismo adecuado para extraer esa información.

En la práctica, la codificación va de la mano con la función de evaluación. Cuando se consideran los parámetros del problema, su codificación dentro del cromosoma debe ser adecuada para asegurar que al ser decodificados, la evaluación pueda realizarse eficientemente. Por ejemplo, si se quiere representar una variable con un rango posible de 1200 valores, se necesitan al menos 11 bits para cubrir este rango, sin embargo, hay otros 848 valores que no se usarán para la evaluación o se usarán incorrectamente.

No sólo una codificación binaria es posible, otros tipos de representación, ya sea con caracteres, enteros, etc., se han estudiado y evaluado. Whitley y Mitchell tienen breves reseñas respecto a este punto, donde se destaca que hay opiniones tanto a favor como en contra de usar representaciones distintas a la binaria tradicional. Sin embargo, no existen reglas claras para definir cuando una representación distinta puede resultar efectiva o no, o que resulte muy dependiente del problema.

3.3.4.4 Selección

La selección es el proceso mediante el cual un subconjunto de individuos en una generación es escogido en razón de su calidad para formar parte de otra generación y para engendrar nuevos individuos. Se pueden describir tres mecanismos conocidos:

Selección proporcional a la calidad (*fitness proportionate selection*): También conocida como selección de la ruleta (*Roulette-Wheel Selection*), se basa en asignar una probabilidad de selección a cada individuo de acuerdo a su calidad. Esto permite que aunque los individuos con alta calidad tengan altas posibilidades, no necesariamente serán seleccionados. Lo mismo en el caso de los individuos con baja calidad, igualmente tienen posibilidades de ser seleccionados, esto es favorable para la variedad y evitar la convergencia prematura.

Selección por torneo (*tournament selection*): Se trata de seleccionar un conjunto de individuos en la población y escoger los mejores entre ellos para ser parte de futuras generaciones y procesos de recombinación. En este tipo de selección, el tamaño del conjunto seleccionado es un factor relevante. En el torneo, se va evaluando con una probabilidad p cada individuo en orden de calidad, si es seleccionado o no para recombinación. Una selección por torneo se define determinista si $p = 1$.

Muestreo estadístico de resto (*remainder stochastic sampling*): Considerando la expresión de calidad f_i/F, cuando este valor es mayor que 1, la porción entera de este número indica cuantas copias del cromosoma respectivo son copiadas a la población intermedia. Todos los cromosomas pondrán copias en la población intermedia con probabilidad equivalente a la parte fraccionaria de f_i/F. Por ejemplo, si este valor es 1.36 se coloca una copia y hay una probabilidad de 0.36 de que se coloque otra.

3.3.4.5 Operadores Genéticos

a. Recombinación o Cruzamiento

En la recombinación, dos individuos generan un tercero mezclando copias de sus genes en el nuevo individuo. El mecanismo más tradicional de cruzamiento es el de un punto. Este mecanismo tiene una desventaja, la que radica en que dependiendo de cómo se encuentren ordenados los bits en el cromosoma, puede romper la composición de un *schema* presente en él. Como se puede apreciar más adelante, la mezcla en recombinación de esquemas de bajo orden pueden ayudar a encontrar soluciones de forma más efectiva, por lo que romper su estructura en el cromosoma no es algo que resulte conveniente. Bajo este contexto, es que se han desarrollado otros tipos de recombinaciones.

A continuación una breve descripción de algunos tipos de recombinación:

Cruzamiento en dos puntos (*double-point crossover*): De forma análoga al cruzamiento en un punto, en este mecanismo se escogen dos puntos en los cromosomas y los segmentos entre los puntos seleccionados se combinan en los nuevos individuos. Por ejemplo, si se tienen los siguientes cromosomas:

1101|0111|00001

0101|0101|00011

Al realizar el cruzamiento en dos puntos, para este caso, en las posiciones 5 y 9, se obtienen los siguientes cromosomas:

Hijo 1: 1101|0101|00001

Hijo 2: 0101|0111|00011

Cruzamiento uniforme (*uniform crossover*): En este mecanismo los genes en cada posición de los cromosomas padres se comparan. Con una probabilidad constante, el *allele* de ambos genes se intercambia. Una variante a este método de cruzamiento es el medio

cruzamiento uniforme (*half uniform crossover*), donde exactamente la mitad de los genes distintos se intercambian.

Cruzamiento parcialmente coincidente (*partially matched crossover*): En este mecanismo se seleccionan dos puntos de cruzamiento de forma aleatoria. Los *alleles* de los cromosomas padres, entre estos dos puntos, son intercambiados con los *alleles* correspondientes a aquellos mapeados por el otro padre. Por ejemplo:

984|567|1320

871|230|9546

Al mirar en el primer padre, el primer gen entre los dos puntos, 5 corresponde al 2 en el otro padre. Por lo que son intercambiados en el primer padre, similarmente se intercambian 6 y 3, 0 y 7; de forma de crear el primer hijo y se aplica al segundo padre. Esto da como resultado:

Hijo 1: 984|230|1657

Hijo 2: 801|567|9243

b. Mutación

Este operador corresponde a la modificación de genes particulares de un cromosoma con una probabilidad muy baja de ocurrencia. La función principal de la mutación es mantener la diversidad de la población y evitar que exista convergencia prematura en la ejecución del algoritmo, además de permitir explorar otros sectores en el espacio de búsqueda.

Pese al mecanismo sencillo con que trabaja la mutación, resulta ser un elemento muy importante en el funcionamiento del Algoritmo Genético. Diversos estudios se han enfocado en comprobar la influencia de este operador en la resolución de problemas complejos, en que se habla de casos comparativos respecto a las capacidades de la recombinación. Sin embargo, se destaca que el balance entre mutación, selección y cruzamiento es importante para evitar los problemas fruto de la falta de balance.

c. Inversión

Este operador revierte el orden de una sección continua en el individuo modificando el orden de los genes en la sección respectiva. A diferencia de la mutación, que se puede aplicar de manera individual a los genes del individuo, la inversión se aplica al individuo como un todo. Por ejemplo, si se tiene el siguiente individuo:

984|567|1320

Al aplicar el operador sobre el segmento delimitado, se obtiene:

984|765|1320

3.3.4.6 Sustitución

Una vez que las nuevas soluciones son creadas usando recombinación y mutación, es necesario incluirlas en la población. Existen diversas maneras de lograrlo, considerando que los cromosomas padres ya han sido seleccionados de acuerdo a su aptitud, por lo que se espera que la descendencia se encuentre entre los más aptos en la población y que la población en promedio mejore su aptitud. Algunas técnicas de sustitución son descritas a continuación:

Supresión de todos: Esta técnica suprime a todos los miembros de la población actual y los reemplaza con el mismo número de cromosomas que han sido creados recientemente. Esta es probablemente una técnica común y quizás sea la elección de muchos debido a su simpleza de implementación. A diferencia de otros métodos no requiere parámetros.

Estado estable: Esta técnica suprime n elementos antiguos de la población y los reemplaza con n elementos nuevos. El número a suprimir y reemplazar (n), en cualquier instante por parte de la aplicación es un parámetro para esta técnica. Otra consideración es la de decidir qué elementos serán suprimidos, pudiendo ser los menos aptos, los padres de la nueva generación o una selección al azar, lo que se constituye en otro parámetro para esta técnica.

Estado estable sin duplicados: Es igual que la técnica anterior, pero el algoritmo se asegura que no sean añadidos a la población cromosomas duplicados. Esto agrega un overead adicional a la técnica, pero significa que al haber más diversidad de elementos el espacio de búsqueda explorado es mayor.

3.3.4.7 Criterio de término

El criterio de término para los Algoritmos Genéticos es el encargado de definir el momento en el cual debe interrumpir el ciclo de evolución y adoptar el individuo más apto como la solución encontrada por el Algoritmo Genético.

A continuación se describen algunos criterios de término comúnmente utilizados:

Criterio de convergencia de identidad: Este criterio consiste en detener el Algoritmo Genético cuando un determinado porcentaje de los individuos representa a la misma solución. Los operadores tienden a preservar y difundir el material genético de los cromosomas más aptos, por lo que es de esperar que luego de un gran número de generaciones, alguna solución con gran valor de aptitud se imponga y domine la población.

Criterio de convergencia de aptitud: Puede suceder que existan soluciones equivalentes o casi equivalentes a un problema, que obtengan valores de fitness parecidos. En estos casos, es probable que no haya una solución que se imponga en la población (y el criterio de término por convergencia de identidad nunca se cumpla). Este criterio no espera a que la población se componga mayoritariamente de una sola solución, sino que finaliza la ejecución del algoritmo cuando los valores de fitness de un determinado porcentaje de las soluciones son iguales, o difieren en un porcentaje dado. Por ejemplo, cuando el 90% de las soluciones tenga valores de aptitud que no difieran en más de un 1%.

Criterio de cantidad de generaciones: El criterio de término por cantidad de generaciones consiste simplemente en finalizar la ejecución una vez que ha transcurrido un número determinado de generaciones. Los métodos anteriores apuntan a esperar a que la evolución de la población llegue a su fin. Cuando alguno de ellos se cumple, es probable que las soluciones no sigan mejorando mucho más, no importan cuantas generaciones más se ejecuten. Sin embargo, los Algoritmos Genéticos pueden necesitar un número de generaciones elevado para llegar a la convergencia, dependiendo de las tasas de reproducción y mutación. Utilizando cualquiera de los dos criterios anteriores no puede estimarse un número máximo de generaciones, ya que esto no dependerá solamente de los parámetros del algoritmo genético sino también del azar. Esto puede ser un problema, en el caso que se quisiera comparar los tiempos de resolución de un problema mediante Algoritmos Genéticos con otros métodos. Por lo que este método permite determinar con precisión los tiempos de ejecución del algoritmo a costa de detener la evolución sin la certeza de que las soluciones seguirán mejorando.

4 Tabu Search

4.1 Inicios

Figura 4.1 Fred Glover

Tabu Search (TS) es una meta heurística propuesta a fines de los años 80 por Fred Glover. *Tabu Search* [1] es una estrategia para resolver problemas de optimización combinatoria, los que abarcan desde teoría de grafos y matrices hasta programación entera. Es un procedimiento altamente adaptativo que tiene la habilidad de hacer uso de otros métodos, tales como algoritmos de programación lineal y heurísticas especializadas, cuyo objetivo principal es superar la barrera de los óptimos locales durante la búsqueda de la mejor solución.

Tabu Search [1] tiene sus orígenes en estudios de combinatoria aplicados a la resolución de problemas no lineales a fines de los 70s, para luego ser aplicado a todo tipo de problemas desde planificación y balanceo de cargas en canales computacionales, hasta análisis de *cluster* y planificación espacial. Las últimas investigaciones y pruebas computacionales involucran problemas como el vendedor viajero (TSP, por sus siglas en inglés), coloreado de grafos, asignación de secuencias de trabajos a máquinas (*Job Shop*), diseño de circuitos integrados y problemas de planificación de tablas de campeonatos, entre los que se encuentra el *Traveling Tournament Problem* (TTP). Hay que destacar que *Tabu Search* tiene una excelente tasa de obtención de soluciones de alta calidad con un uso moderado de recursos computacionales para la mayoría de los problemas mencionados.

En resumen, *Tabu Search* es un mecanismo general de imposición de restricciones que es utilizado como guía para heurísticas definidas de problemas particulares, propuesto a fines de los 80s, con el objetivo de cruzar zonas de óptimos locales durante la búsqueda de soluciones. Estas restricciones se basan principalmente en la exclusión o prohibición directa de alternativas de búsqueda (clasificadas tabú), las que se definen en función de lo realizado en etapas anteriores de la misma. Es una metodología flexible y eficaz en la resolución de problemas, que ha sido utilizada con muy buenos resultados en problemas de optimización combinatoria y problemas reales.

4.2 Definición

Teniendo definido un conjunto de soluciones del problema a tratar, para minimizar una función de costo que se quiere optimizar. Nos encontramos con el concepto de vecindario, en el cual se realizan los movimientos mediante pequeñas modificaciones (o movimientos) para movernos entre las distintas soluciones alcanzables desde el punto actual.

Para poder utilizar el algoritmo TS [2] es imprescindible su componente principal, la lista tabú (es una lista de movimientos no permitidos), es una lista especial de memoria de corto plazo que guarda un historial selectivo compuesto por configuraciones previamente encontradas o atributos generales de dichas configuraciones. La meta de la lista tabú es evitar caer en ciclos y terminar en mínimos locales.

Las entradas en la lista tabú son generadas de a pares. En efecto, se hacen calzar los intercambios. Una vez realizado un intercambio, su pareja inversa es clasificada tabú por las siguientes *k* iteraciones (*k*, llamada *tabu tenure* es dependiente del problema), lo que significa que realizar la reversa del intercambio está prohibido durante ése período.

Tabu tenure juega un rol importante en el desempeño del algoritmo tabú. Si se sobreestima podría prohibir de forma incorrecta la exploración de un espacio de búsqueda no visitado, y por lo tanto la capacidad del algoritmo de visitar el vecindario se verá reducida. En cambio, si se desestima el algoritmo podría quedar atrapado en mínimos locales.

4.3 Formulación

Para describir cómo funciona *Tabu Search* (TS) [1], se presenta un ejemplo de optimización de un problema de combinatoria de la siguiente manera:

$$(P) \qquad Minimizar\ c(x)\colon x \in X\ en\ R_n$$

El objetivo de la función $c(x)$ puede ser lineal o no lineal, y la condición $x \in X$ se asume como una restricción especificada por valores discretos de x. Para algunas configuraciones (P) puede ser la modificación de algún problema, tal como X es un conjunto de vectores, los que normalmente representan soluciones factibles y $c(x)$ es una función de penalización diseñada para asegurar que las soluciones óptimas alcanzadas por (P) sean óptimas del problema y no locales.

Un largo rango de procedimientos, heurísticos y óptimos, para la resolución de varios problemas es factible de ser escrito de la forma (P) y ser caracterizados convenientemente como referencias a secuencias de "movimientos" que pueden llevarnos desde una "solución de prueba" (seleccionando $x \in X$) a otra. Definiremos un movimiento s consistente en una asignación en un subconjunto $X(s)\ de\ X$:

$$s\colon X(s) \to X$$

Asociándolo con $x \in X$, sería el set $S(x)$ que consiste de los diferentes movimientos $s \in S$ que pueden ser aplicados a x.

TS comienza la búsqueda desde una solución inicial cualquiera (incluida en el vecindario), generada de forma aleatoria o determinística. Luego se mueve iterativamente desde una solución a otra vecina de acuerdo a la definición de vecindad que se esté utilizando. Además, en ciertas etapas se realizan procesos de intensificación y diversificación. El algoritmo termina cuando se cumple el criterio de término definido, que suele involucrar una cantidad máxima de iteraciones totales, o una cantidad máxima de iteraciones sin mejoras.

Otros elementos importantes del TS son las estrategias de intensificación y diversificación. Las estrategias de intensificación se refieren al hecho de explorar regiones del espacio consideradas buenas. Están basadas en la modificación de las reglas de elección para estimular combinaciones de elementos y atributos de una buena solución ya encontrada. Pueden incluir desde un regreso a regiones consideradas buenas para realizar la misma

búsqueda TS, pero con una lista tabú de menor tamaño, hasta la implementación de algoritmos de optimización local particulares al problema. Por otro lado, la diversificación se refiere al hecho de permitir al algoritmo analizar distintas regiones del espacio de búsqueda, evitando volver siempre a las mismas zonas. En su caso más extremo, esto puede implicar reiniciar la búsqueda desde una nueva solución generada al azar. Las estrategias de diversificación son particularmente de ayuda cuando soluciones mejores pueden encontrarse sólo si se cruzan barreras de la topología del espacio de búsqueda.

Basado en [5], para poder aplicar la búsqueda local para TTP es necesario definir algunas características. Primero, consideraremos las definiciones acerca del espacio de búsqueda, la función de costo y el procedimiento computacional para obtener la solución inicial. Entonces, definiremos algunas estructuras del vecindario y analizaremos sus características e indicaciones.

4.3.1 Espacio de Búsqueda, Función de Costo y Estado Inicial

Como espacio de búsqueda, seleccionaremos todas las soluciones correctas para el TTP, incluyendo aquellas que violan las reglas de no repetición de partidos consecutivos entre dos equipos (que identificaremos como *H1*) y el no exceder los 3 partidos de local y visita para cada equipo durante la ronda (que identificaremos como *H2*). Por lo tanto, no es posible que un equipo juegue cero o dos encuentros en una misma ronda, pero puede ocurrir, por ejemplo, que dos equipos jueguen en rondas consecutivas violando la restricción *H1*.

La función de costo es una ponderación entre las violaciones a las restricciones y la suma de las distancias recorridas. Los pesos de las violaciones a las restricciones se ajustan respecto al peso que tienen respecto a la generación de soluciones no factibles para el TTP, por lo que algunas de ellas pueden llegar a pesar más que las distancias en la ponderación. De todas formas, los pesos variarán dinámicamente durante las búsquedas, pudiendo explorarse soluciones no factibles dentro del espacio de búsqueda.

4.3.2 Vecindarios (Neighborhood)

Al revisar lo expuesto en [5], nos encontramos que existen muchos vecindarios en el espacio de búsqueda definido, para lo cual se revisará la definición basada en los vecindarios definidos por Anagnostopoulos.

Se define el conjunto de equipos como $T = \{0, \dots, n-1\}$ y el conjunto de rondas $R = \{0, \dots, r-1\}$, el vecindario se define de la siguiente manera:

1.- N1, SwapHomes:

Atributos: $(t1, t2)$, dónde $t1, t2 \in T$.

Precondiciones: $t1 \neq t2$.

Efecto: Intercambio de calidad visitante/local en las dos rondas disputadas entre *t1* y *t2*.

Ronda	t1	t2	t3
1	t2	- t1	- t4
2	- t5	t7	- t8
3	- t2	t1	- t6

↓

Ronda	A	B	C
1	- t2	t1	- t4
2	- t5	t7	- t8
3	t2	- t1	- t6

Figura 4.2 N1, SwapHomes

2.- N2, SwapTeams:

Atributos: $(t1, t2)$, dónde $t1, t2 \in T$.

Precondiciones: $t1 \neq t2$.

Efecto: Intercambio de los juegos disputados por *t1* y *t2*.

Ronda	A	B	C
1	- F	C	- B
2	D	E	- D
3	- H	F	- F

↓

Ronda	A	B	C
1	- B	C	- F
2	- D	E	D
3	- F	F	- H

Figura 4.3 N2, SwapTeams

3.- N3, SwapRounds:

Atributos: $(r1, r2)$, dónde $r1, r2 \in R$.

Precondiciones: $r1 \neq r2$.

Efecto: Todos los partidos programados para *r1* serán movidos a *r2* y viceversa.

Ronda	t1	t2	t3
1	- t6	t3	- t2
2	t4	t5	- t4
3	- t8	t9	- t6

↓

Ronda	t1	t2	t3
1	- t6	t3	- t2
2	- t8	t9	- t6
3	t4	t5	- t4

Figura 4.4 N3, SwapRounds

4.- N4, SwapMatches:

Atributos: $(r, t1, t2)$, dónde $r \in R \; y \; t1, t2 \in T$.

Precondiciones: $t1 \neq t2$ y, $t1 \; y \; t2$ no juegan al mismo tiempo en r.

Efecto: Los oponentes de $t1 \; y \; t2$ son intercambiados en la ronda r. $t1 \; y \; t2$ permanecen en su misma ubicación (local o visita), pero sus oponentes posiblemente cambien de locación

Nota: Es necesario realizar cambios posteriores para ajustar la solución y no caer en un espacio de solución no factible.

Ronda	t1	t2	t3
1	- t6	t3	- t2
2	t4	- t5	- t7
3	- t8	t9	- t6

↓

Ronda	t1	t2	t3
1	- t6	t3	- t2
2	- t5	t4	- t7
3	- t8	t9	- t6

Figura 4.5 N4, SwapMatches

5.- N5, SwapMatchRound:

Atributos: $(t, r1, r2)$, dónde $t \in T$ *y* $r1, r2 \in R$.

Precondiciones: $r1 \neq r2$ y, t no juega con el mismo oponente en $r1$ *y* $r2$.

Efecto: Los oponentes de t en las rondas $r1$ *y* $r2$ son intercambiados. t permanece en su misma ubicación (local o visita), pero sus oponentes posiblemente cambien de locación

Nota: Es necesario realizar cambios posteriores para ajustar la solución y no caer en un espacio de solución no factible.

Ronda	t1	t2	t3
1	- t6	t3	- t2
2	t4	t5	- t4
3	- t8	t9	- t6

↓

Ronda	t1	t2	t3
1	- t6	t5	- t2
2	t4	t3	- t4
3	- t8	t9	- t6

Figura 4.6 N5, SwapMatchRound

5 Desarrollo

En este capítulo se explicará la metodología a utilizar y se detallará el algoritmo *Tabu Search* que será implementado para resolver el problema del TTP. Este algoritmo tiene sus bases en el utilizado por Cardemil en [3], el que será implementado y modificado en base a las definiciones del trabajo a realizar en el Proyecto.

5.1 Decisiones de Diseño

Se utilizarán dos metodologías para abordar el problema a resolver, una de ellas es la investigativa, para recolectar toda la información necesaria y establecer el marco teórico del proyecto y la otra es el desarrollo en espiral, para realizar el software, sus pruebas y posterior ajuste de código y parámetros.

El método investigativo utilizado busca plantear una hipótesis para la cual se obtiene información que debe ser probada mediante experimentación. En particular se utilizará sólo para poder obtener la información necesaria que permita establecer de manera fehaciente el marco teórico y estado del arte en que se basará el estudio que está siendo llevado a cabo en este trabajo. Algunas de las fases importantes que es necesario tener en cuenta:

- Recolección del material de estudio para analizarlo, el que puede ser *papers*, tesis u otro tipo de publicaciones.

- Filtrar dicho material, para luego realizar una selección de los *papers* que serán de utilidad para la definición del marco teórico.

- Generar un estado del arte que contenga el material relevante respecto al cual se está realizando el estudio en cuestión.

Finalmente el desarrollo espiral, más conocido como método iterativo incremental, en el que se realizan análisis y mediciones como guías para el proceso de mejora, lo que radica en una diferencia mayor entre las mejoras iterativas y el desarrollo rápido de aplicaciones, principalmente por dos razones:

- Provee de soporte para determinar la efectividad de los procesos y de la calidad del producto.

- Permite estudiar y después mejorar y ajustar el proceso para el ambiente en particular.

Estas mediciones y actividades de análisis pueden ser añadidas a los métodos de desarrollo rápido existentes.

De hecho, el contexto de iteraciones múltiples conlleva ventajas en el uso de mediciones. Las medidas a veces son difíciles de comprender en lo absoluto, aunque en los cambios relativos en las medidas a través de la evolución del sistema puede ser muy informativo porque proveen una base de comparación. Así el observador puede decidir

respecto a las características del producto como el tamaño, la complejidad, el acoplamiento y la cohesión incrementan o disminuyen en el tiempo. También puede monitorearse el cambio relativo de varios aspectos de un producto o pueden proveer los límites de las medidas para apuntar a problemas potenciales y anomalías.

5.2 Algoritmo Tabu Search para Resolver el TTP

5.2.1 Esquema del Algorimo Implementado

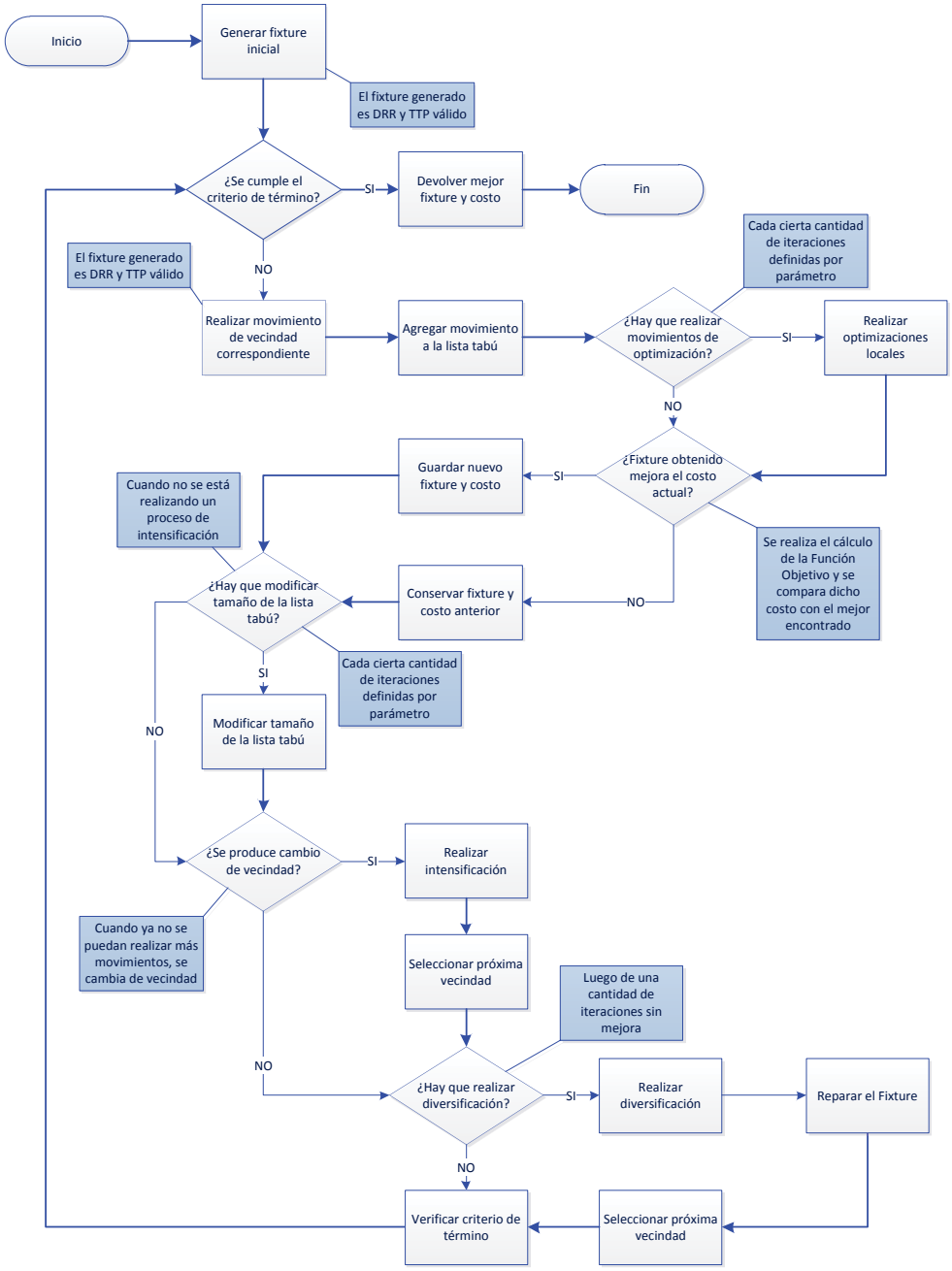

Figura 5.1 Diagrama algoritmo implementado

El algoritmo comienza cuando se genera el fixture inicial y se obtiene el valor de dicho fixture mediante el cálculo de la función objetivo del problema (estos serán los valores globales hasta que se consiga una mejor solución), luego se realiza el movimiento de vecindad correspondiente y se guarda el movimiento en la lista tabú, el que debe ser un fixture válido y mejore o iguale a la mejor solución global encontrada realizando optimizaciones locales si corresponde y la verificación de la solución, para actualizar los valores globales con la mejor solución encontrada en cada ejecución del algoritmo.

Luego, se verifica si es necesario modificar el tamaño de la lista tabú, lo que implica aumentar o disminuir la capacidad de memoria que tiene esta lista y con ello los movimientos prohibidos para el algoritmo, lo siguiente es la verificación de rutinas de intensificación y diversificación, las que revisan de forma intensa el espacio de búsqueda en que se encontró una solución (intensificación) y la diversificación del espacio de búsqueda, lo que implica realizar movimientos que involucran el paso por espacios de búsqueda no factibles hasta llegar otra vez a uno factible (diversificación) y continuar desde allí la búsqueda; hay que saber además que ambas técnicas cambian la vecindad que se encuentra guiando la búsqueda.

Finalmente se verifica si se ha cumplido con el criterio de término del algoritmo, que dará por agotada la búsqueda ya sea porque agotó la cantidad de iteraciones a realizar o lleva muchas iteraciones sin lograr mejorar la solución global.

5.2.2 Espacio de Soluciones

Denotaremos como F al conjunto de soluciones factibles, que serán representadas mediante matrices. Dado que el TTP es un torneo DRR, se tiene que la cantidad R de rondas del torneo es $2\ x\ (n-1)$ y que la celda M_{re} toma valores enteros (positivos y negativos) entre 1 y n. Asumiremos que un valor positivo en la celda M_{re} indica que el equipo e juega de local la ronda r contra el equipo M_{re}, mientras que un valor negativo indica que dicho equipo e juega de visitante en la ronda r con el equipo $-M_{re}$. Entonces, diremos que una solución factible para el TTP con n equipos es cualquier matriz M^x de dimensión $R\ x\ n$ que satisfaga las siguientes restricciones:

1.- Restricciones generales para un torneo DRR (sin espejamiento)

(1) $M_{r,e} \neq M_{s,e}\ \forall\ r \neq s\ (e = 1 \dots n;\ r, s = 1 \dots R)$

*Todo equipo juega dos veces (partido y revancha) contra todos los demás.

(2) $\left[(i \neq j) \Longrightarrow \left|M_{r,i}\right| \neq \left|M_{r,j}\right|\right]\ (i, j = 1 \dots n)$

*En cada ronda, todos juegan contra un equipo distinto.

(3) $\left[M_{i,r} = j \Leftrightarrow M_{j,r} = -i\right]\ \forall r\colon 1 \leq r \leq R, (i, j = 1 \dots n)$

*Si A juega de local contra B, entonces B juega de visitante contra A.

2.- Restricciones particulares para TTP

(4) $\left|M_{r,e}\right| \neq \left|M_{r+1,e}\right|$ $(e = 1 \dots n; r = 1 \dots R - 1)$

*Ningún par de equipos juega entre sí en dos rondas seguidas.

(5) $\left[M_{r,e} > 0 \wedge M_{r+3,e} > 0\right] \Longrightarrow \left[M_{r+1,e} < 0 \vee M_{r+2,e} < 0\right]$ $(e = 1 \dots n; r = 1 \dots R - 3)$

*Ningún equipo juega más de 3 partidos de local.

(6) $\left[M_{r,e} < 0 \wedge M_{r+3,e} < 0\right] \Longrightarrow \left[M_{r+1,e} > 0 \vee M_{r+2,e} > 0\right]$ $(e = 1 \dots n; r = 1 \dots R - 3)$

*Ningún equipo juega más de 3 partidos de visita.

5.2.3 Solución Inicial

Se implementarán dos algoritmos para generar la solución inicial. Ambos generan fixtures en forma aleatoria, pero el primero de ellos lo hace siempre siguiendo un mismo "esquema" o "patrón" establecido. Éste es una variante del expuesto en [3] propuesto por Scheuder, J.A.M para generar fixtures de una vuelta. Por otro lado, el segundo algoritmo que se basa en el expuesto en [3] propuesto por Dinitz, J. & Stinson D. genera fixtures aleatorios sin seguir algún patrón en especial. Ambos algoritmos generan fixtures válidos para torneos DRR, pero no siempre son válidos para TTP.

Por lo tanto, al resultado obtenido por cualquiera de estos algoritmos se les aplica una serie de mejoras (basadas en lo descrito en 5.2.7) y correcciones en los HAP de los equipos que violen las restricciones establecidas por el TTP. En caso que no se pueda corregir rápidamente, se repite el proceso hasta obtener un fixture correcto, es decir, que cumpla con las 6 restricciones planteadas anteriormente para DRR y TTP en la sección 5.2.2.

5.2.4 Función Objetivo

Se utilizará como función de evaluación la suma de los recorridos de cada equipo a lo largo del torneo, ya que justamente es el objetivo del *Traveling Tournament Problem* (TTP) minimizar dicha suma, respetando las restricciones expuestas.

Primero, se define una matriz de distancias $DIST$ de dimensiones $n\ x\ n$, por lo tanto el valor denotado por $DIST_{ij}$ representa la distancia entre el estadio del equipo i y el estadio del equipo j.

Además, hay que recordar que el TTP especifica que todos los equipos comienzan (ronda 0) y terminan (ronda R+1) el torneo en su propio estadio, asumiremos que $M_{0,k} > 0$ y que $M_{R+1,k} > 0$ $(\forall k: 1, \dots, n)$. Luego, definimos a T_{kr} como la distancia que debe recorrer el equipo k del estadio donde juega la ronda $r - 1$ al estadio donde juega la ronda r, de la siguiente manera:

$$T_{k,r} = \begin{cases} DIST_{|M_{r-1,k}|,|M_{r,k}|} & si\ \left(M_{r-1,k} < 0\right) \wedge \left(M_{r,k} < 0\right)\ (i) \\ DIST_{|M_{r-1,k}|,k} & si\ \left(M_{r-1,k} < 0\right) \wedge \left(M_{r,k} > 0\right)\ (ii) \\ DIST_{k,|M_{r,k}|} & si\ \left(M_{r-1,k} > 0\right) \wedge \left(M_{r,k} < 0\right)\ (iii) \\ 0 & si\ \left(M_{r-1,k} > 0\right) \wedge \left(M_{r,k} > 0\right)\ (iv) \end{cases}$$

(i) El equipo k juega de visitante en las rondas $r-1$ y r. Entonces la distancia recorrida es la que separa a los estados de los dos rivales de k en las rondas $r-1$ y r.

(ii) El equipo k juega de visitante en la ronda $r-1$ y de local en la ronda r. Luego, la distancia buscada es la que separa al estadio de k con el estadio de su rival en la ronda $r-1$.

(iii) El equipo k juega de local en la ronda $r-1$ y de visitante en la ronda r. Luego, la distancia buscada es la que separa al estadio de k con el estadio de su rival en la ronda r.

(iv) El equipo k juega de local en las rondas $r-1$ y r. Por lo tanto, permanece en su estadio y la distancia que recorre entre esas dos rondas es cero, por tratarse de la misma ubicación.

Finalmente, definimos la función de evaluación $f: F \to N$ de la siguiente forma:

$$f(x) = \sum_{e=1}^{n} C_e$$

Donde C_e indica la distancia total recorrida por el equipo e a lo largo de todo el torneo según el fixture x. El valor de C_e $(e = 1 \ldots n)$ se calcula de la siguiente manera:

$$C_e = \sum_{r=1}^{R+1} T_{e,r}$$

Esquema Algoritmo Cálculo de la Función Objetivo

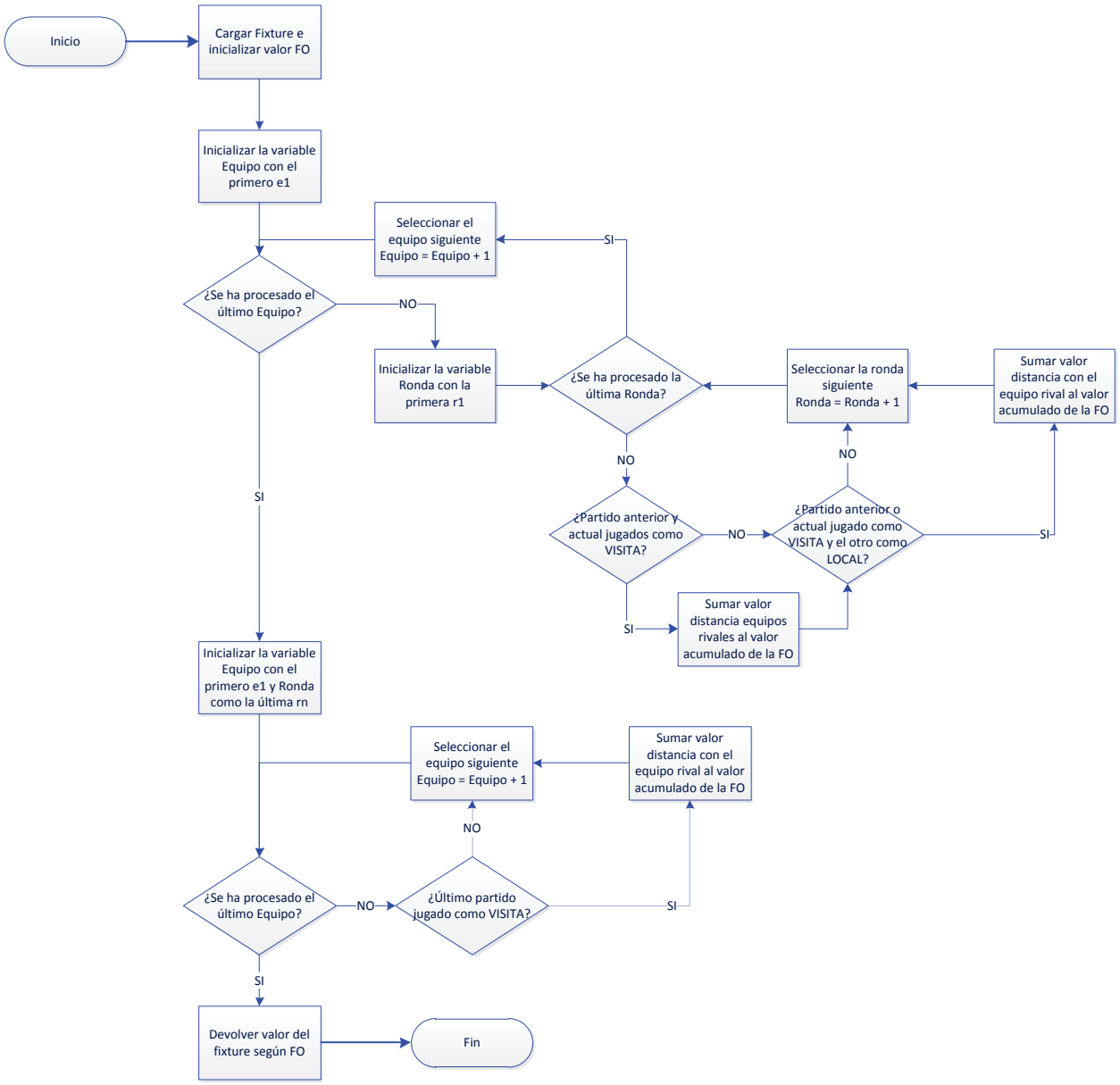

Figura 5.2 Diagrama algoritmo cálculo de la función objetivo

5.2.5 Vecindarios

A diferencia del esquema clásico de TS, en lugar de definir un solo vecindario que dirige la búsqueda en el espacio de soluciones, se definirán varios, los cuales se van alternando en distintas etapas del proceso de exploración, aunque algunos se utilizan durante mayor tiempo que otros (especialmente el vecindario principal o *IPR*).

Esta idea es aplicada no sólo para reducir el riesgo de quedar atrapados en mínimos locales, sino que a la vez se realiza una diversificación indirecta de la búsqueda. En las próximas secciones se detallarán los vecindarios utilizados. Se asumirá a S como el conjunto de soluciones factibles de torneos DRR sin considerar las restricciones particulares del TTP.

De esta forma, tenemos que $F \subset S$. Pero para todos los vecindarios que se describirán a continuación, se considerarán vecinos sólo aquellos fixtures válidos para el TTP. En consecuencia, durante el proceso normal de búsqueda cuando no estamos usando diversificación, sólo se estarán explorando regiones factibles. Pero en las etapas de diversificación, el algoritmo sí se moverá intencionalmente por regiones no factibles, con el objetivo de llegar a otras regiones factibles que, de otra manera, podrían no ser visitadas nunca.

5.2.5.1 Intercambio Parcial de Rondas – IPR

La vecindad $IPR(x)$ de un fixture $x \in S$ consiste en todos los fixtures que se pueden obtener aplicando a x un movimiento *IPR*. Un movimiento *IPR* es una permutación de cuatro partidos entre dos rondas (r_1, r_2) que involucra sólo a cuatro equipos (e_1, e_2, e_3, e_4). Para que un movimiento *IPR* (definido por r_1, r_2, e_1, e_2, e_3, e_4) pueda ser aplicado a un fixture x, éste debe satisfacer las siguientes propiedades:

$$x.rival(e_1, r_1) = x.rival(e_2, r_2) = e_3$$

$$x.rival(e_1, r_2) = x.rival(e_2, r_1) = e_4$$

Donde $x.rival(e, r)$ indica el equipo contra el que juega e en la ronda r en el fixture x, es decir $\left|M_{r,e}^{x}\right|$. Si se cumplen las condiciones expuestas, entonces tenemos que en x están definidos los siguientes cuatro encuentros:

Equipo/ Ronda	Ronda 1 (r_1)	Ronda 2 (r_2)
Equipo 1 (e_1): A	- C	D
Equipo 2 (e_2): B	- D	C
Equipo 3 (e_3): C	- F	- E
Equipo 4 (e_4): D	B	- G

Figura 5.3 Encuentros seleccionados para aplicar vecindad IPR a un fixture *x*

El movimiento consiste en cambiar las rondas de estos partidos manteniendo la estructura de todos los demás. Esto significa que de estos cuatro partidos, los que se jugaban en la ronda r_1 pasan a jugarse en la ronda r_2, mientras que los que se jugaban en la ronda r_2 lo harán en la ronda r_1. En definitiva, el resultado de aplicar un movimiento *IPR* a un fixture x (que cumple con las propiedades señaladas) es un nuevo fixture y que tiene las siguientes características:

$$y.rival(e_1, r_1) = y.rival(e_2, r_2) = e_4 = x.rival(e_1, r_2) = \left|M^x_{r2,e1}\right|$$

$$y.rival(e_1, r_2) = y.rival(e_2, r_1) = e_3 = x.rival(e_1, r_1) = \left|M^x_{r1,e1}\right|$$

El resto de los encuentros se mantiene igual, entonces tenemos que según las modificaciones realizadas, ahora en y están definidos los siguientes cuatro encuentros:

Equipo/ Ronda	Ronda 1 (r_1)	Ronda 2 (r_2)
Equipo 1 (e_1): A	D	- C
Equipo 2 (e_2): B	C	- D
Equipo 3 (e_3): C		
Equipo 4 (e_4): D	B	- G

Figura 5.4 Resultado de la aplicación de la vecindad IPR a un fixture *x*

Esto finalmente se traduce en el intercambio de una sección de dos rondas en el fixture, es decir, se intercambian las celdas correspondientes a las dos columnas tal como se puede ver en la Figura.

Aunque se puede observar fácilmente que al aplicar un movimiento *IPR* sobre un fixture cualquiera, el resultante siempre será un fixture que cumpla el esquema DRR; es importante destacar que esto no siempre es cierto para que el fixture sea considerado válido para TTP. En particular, al aplicar un movimiento *IPR* podemos obtener un vecino que viola cualquiera de las restricciones definidas por 4, 5 y 6 en la sección 5.2.2.

Es importante mencionar que al realizar un movimiento *IPR* se consideran todas las variantes posibles respecto de la localia de los cuatro partidos que se alteran originalmente (esto es 2^4 combinaciones posibles), seleccionado entre las que resulten factibles aquella con la que se obtiene el mejor resultado. También cabe destacar que es posible (especialmente para las instancias con pocos equipos) que no se pueda realizar ningún movimiento *IPR* válido. Cuando esto sucede, automáticamente el algoritmo deja de utilizar esta vecindad como guía en el proceso de búsqueda y continúa usando alguna de las que se describen más adelante.

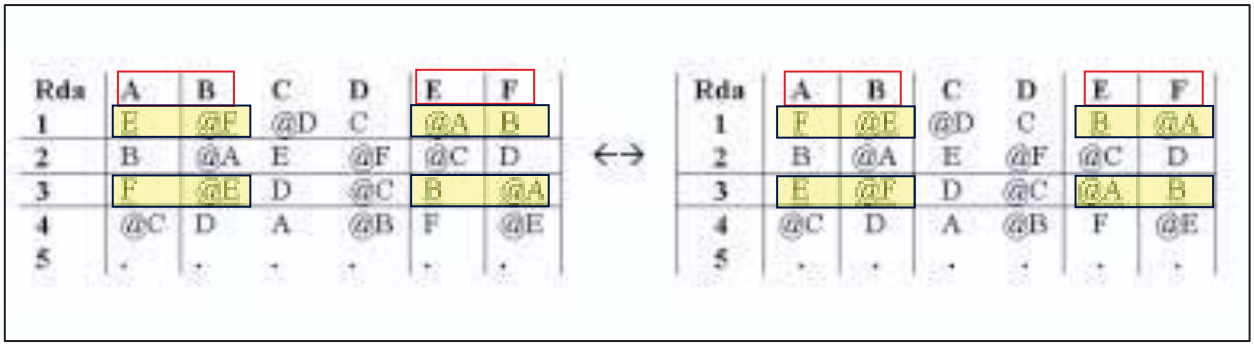

Figura 5.5 Ejemplo de vecindad IPR

Esquema Algoritmo Vecindad IPR

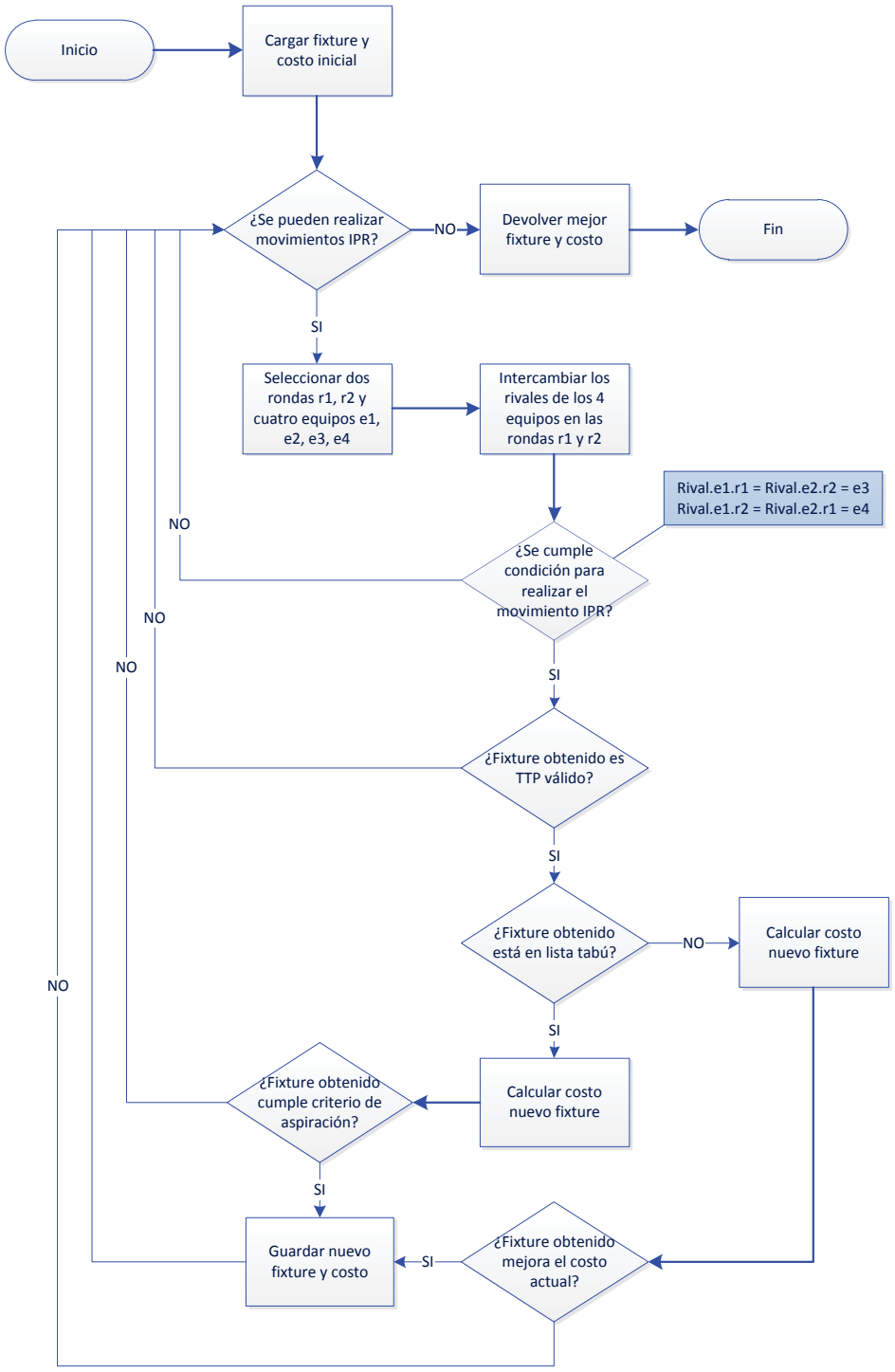

Figura 5.6 Diagrama algoritmo vecindad IPR

5.2.5.2 Intercambio de Rondas – IR

La vecindad $IR(x)$ de un fixture $x \in S$ consiste en todos los fixtures que se pueden obtener a partir de x, permutando un par de rondas. Es decir, que simplemente se permutan dos filas de la matriz M^x asociada al fixture x.

Está claro que el fixture resultante luego de una permutación de este tipo no viola ninguna de las tres primeras condiciones (correspondientes a DRR) sobre M^x, pero si puede violar cualquiera de las tres últimas (correspondientes a TTP).

Ronda	A	B	C
1	- F	C	- B
2	D	E	- D
3	- H	F	- F

↓

Ronda	A	B	C
1	- F	C	- B
2	- H	F	- F
3	D	E	- D

Figura 5.7 Representación de la aplicación de una vecindad IR a un fixture *x*

Esquema Algoritmo Vecindad IR

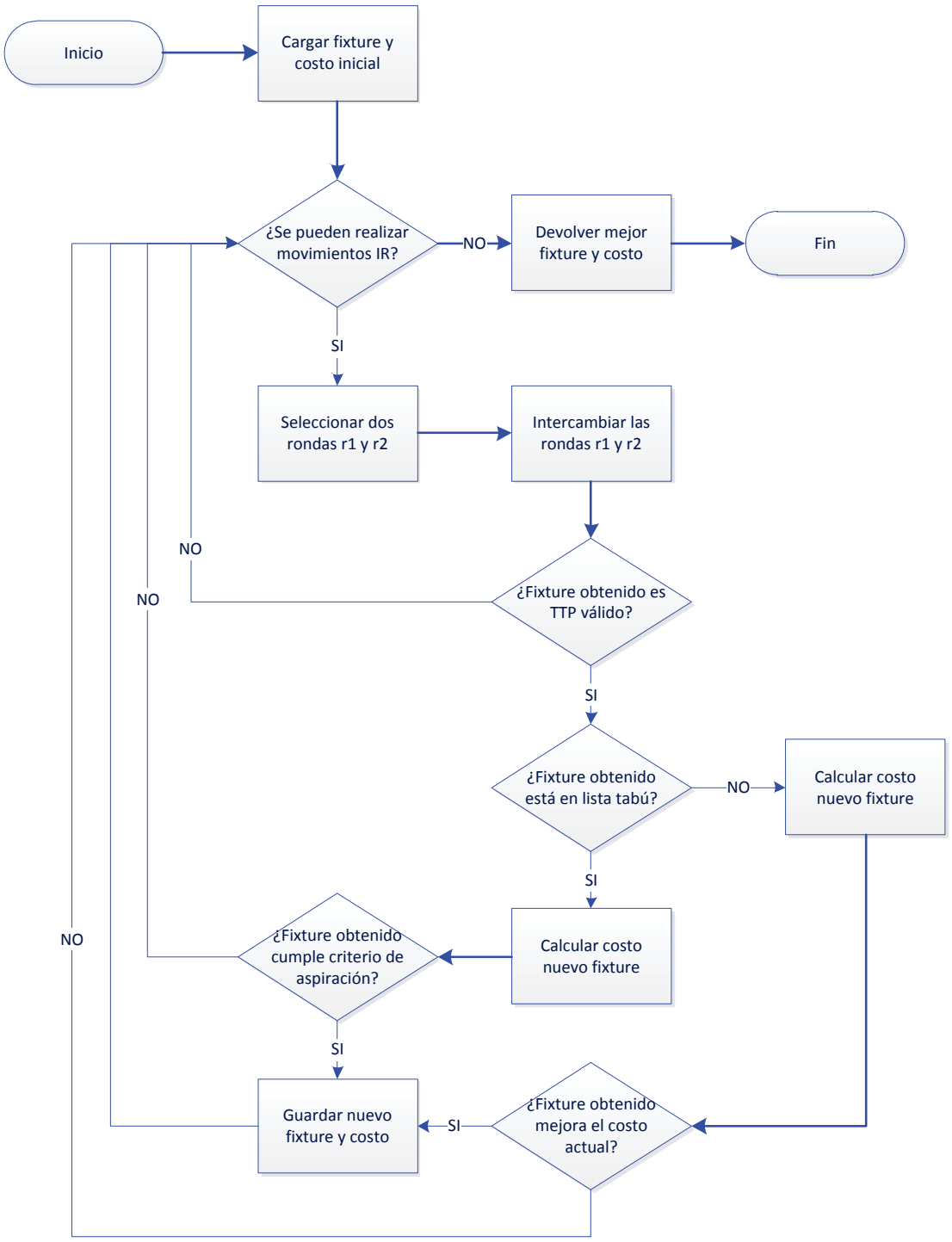

Figura 5.8 Diagrama algoritmo vecindad IR

5.2.5.3 Intercambio de Equipos – IE

La vecindad $IE(x)$ de un fixture $x \in S$ consiste en todos los fixtures que se pueden obtener a partir de x, permutando un par de equipos. Es decir, que simplemente se permutan dos columnas de la matriz M^x asociada al fixture x.

Esta permutación es equivalente a renombrar los equipos, por lo que esta vecindad no puede generar fixtures no factibles, si es que se realiza desde uno factible.

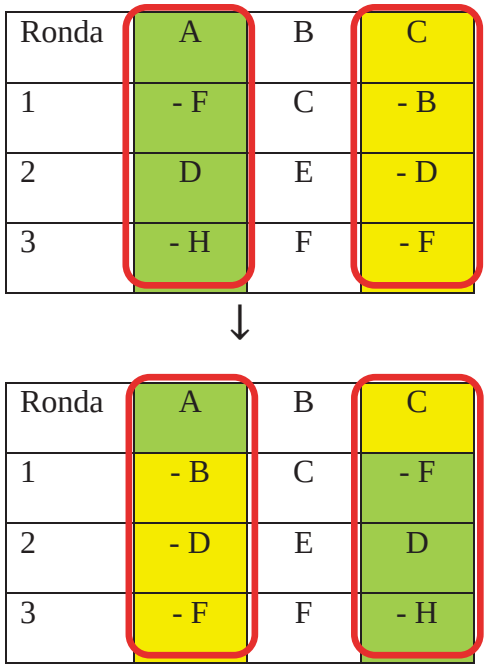

Figura 5.9 Representación de la aplicación de una vecindad IE a un fixture *x*

Esquema Algoritmo Vecindad IE

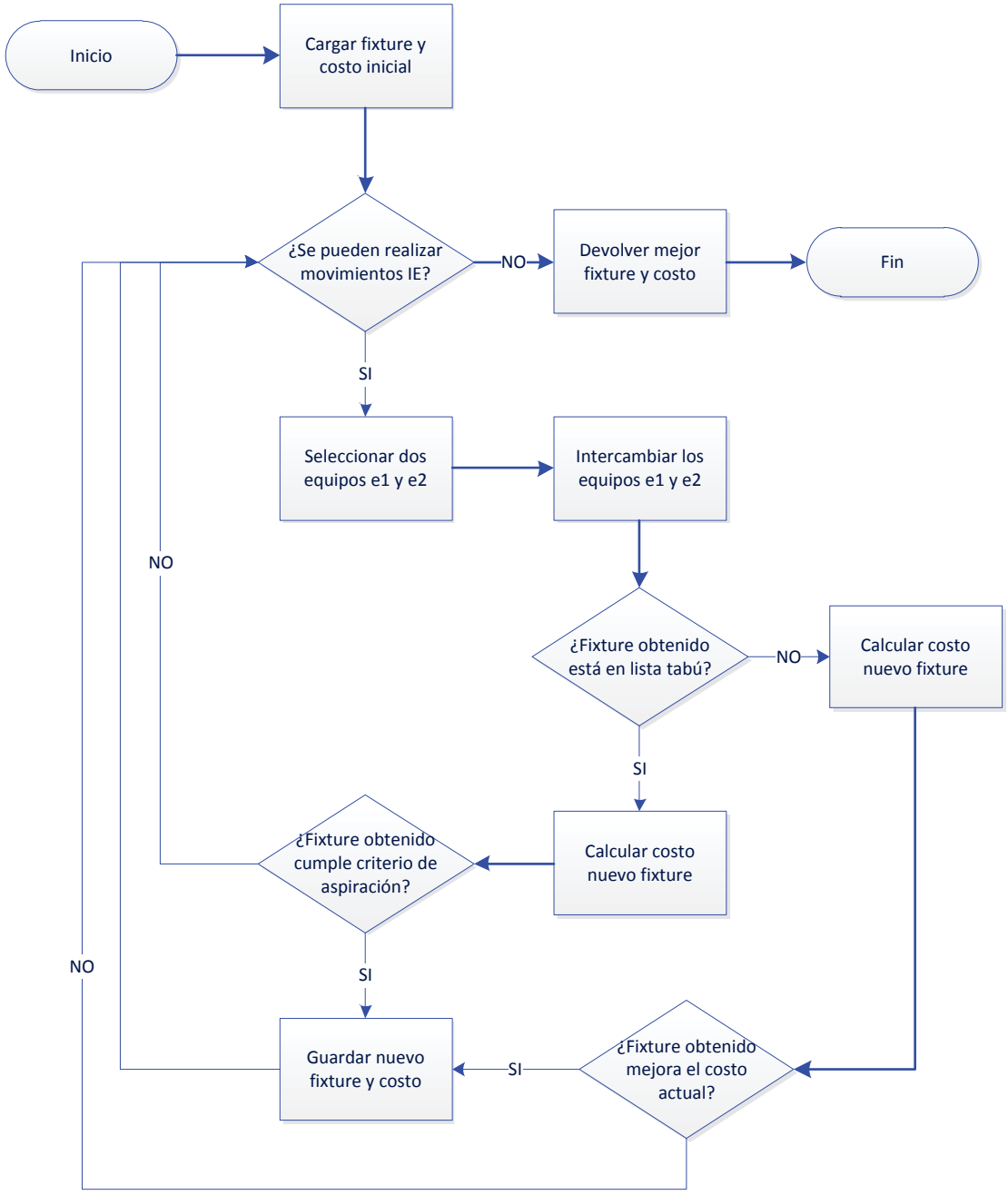

Figura 5.10 Diagrama algoritmo vecindad IE

5.2.5.4 Intercambio de Localias – IL

La vecindad $IL(x)$ de un fixture x^{es}, consiste de todos los fixtures que se pueden obtener a partir de x, cambiando la localía de los dos partidos entre un mismo par de equipos e_1 y e_2. Es decir, que si en x, e_1 jugaba de local contra e_2 en la ronda r_1, y de visitante en la ronda r_2, las únicas diferencias entre x y el vecino que se obtiene permutando las localías de

los equipos e_1 y e_2 es simplemente que en este nuevo fixture, e_1 juega contra e_2 de visitante en la ronda r_1 y de local en la ronda r_2.

Está claro que el fixture resultante, luego de una permutación de este tipo, sólo puede llegar a violar las restricciones 5 o 6 (sección 5.2.2) pues la "estructura" de la matriz se mantiene igual.

Ronda	A	B	C
1	B	- A	- E
2	- H	F	- D
3	- B	A	- F

↓

Ronda	A	B	C
1	- B	A	- E
2	- H	F	- D
3	B	- A	- F

Figura 5.11 Representación de la aplicación de una vecindad IL a un fixture *x*

Esquema Algoritmo Vecindad IL

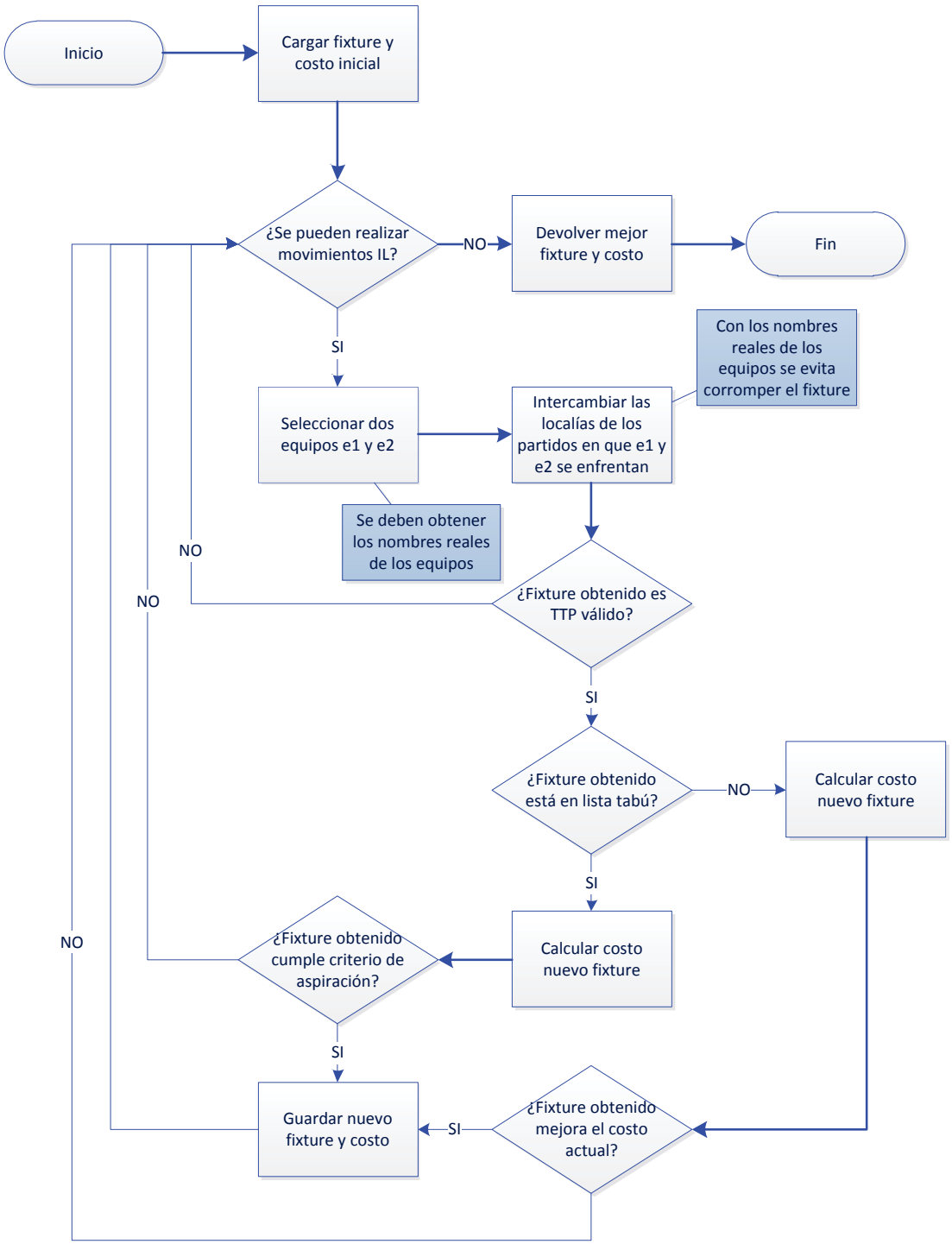

Figura 5.12 Diagrama algoritmo vecindad IL

5.2.6 Manejo de la Lista Tabú

La lista Tabú registra cuales son los movimientos considerados prohibidos. Según cual sea la vecindad que se esté utilizando para dirigir la búsqueda, en la lista Tabú se guardan los últimos t movimientos realizados. Durante el proceso normal de búsqueda, el valor t se modifica cada cierta cantidad de iteraciones, las que se definen como parámetro, eligiendo su valor en forma aleatoria en un rango determinado (el que también se establece como parámetro). Cada vez que se realiza un movimiento, éste se incorpora a la lista, y a su vez (si la lista está completa) se elimina de ésta el movimiento que mayor tiempo ha permanecido en ella. En resumen, podemos decir que la lista Tabú tiene un comportamiento FIFO (el primer elemento en llegar es el primero en salir).

5.2.7 Optimizaciones Locales

Sumado al proceso de búsqueda tabú que realiza la exploración del espacio de soluciones utilizando como guía (alternadamente) las distintas vecindades anteriormente descritas, el algoritmo cada cierta cantidad de iteraciones (que se definen mediante parámetro) intenta efectuar algunas pequeñas optimizaciones locales.

Para esto, se evalúa el costo de todos los fixtures válidos para el TTP que se obtienen mediante las operaciones que se describen a continuación, y si se obtienen mejoras se aplica la operación sobre la solución actual y se repite el proceso hasta que no se obtengan más mejoras. Estas operaciones se van aplicando en el mismo orden en el que se mencionan:

1.- Permutación de 2 rondas: ver la definición de la vecindad IR.

2.- Permutación de 2 equipos: ver la definición de la vecindad IE.

3.- Permutación de localias: (entre los dos partidos que juegan entre sí un par de equipos) ver la definición de la vecindad IL.

4- Inversión de *Tours*: se invierte la localia de todos los partidos (*tour* o campeonato) de cada uno de los equipos.

5.2.8 Criterio de Aspiración

El criterio de aspiración utilizado consiste en permitir realizar un movimiento que está considerado Tabú, cuando la aplicación del mismo genera una solución cuyo costo es inferior al mejor costo obtenido hasta ese momento.

Además, es importante destacar que, si en algún momento la lista Tabú llegara a prohibir todos los posibles movimientos, el algoritmo inmediatamente cambia la dirección de la búsqueda seleccionando otra vecindad.

5.2.9 Intensificación

El objetivo principal de la intensificación es explotar regiones del espacio de soluciones que se consideran "buenas". La estrategia de intensificación utilizada en el algoritmo es detallada a continuación:

5.2.9.1 Disminución del tamaño de la lista Tabú

Esta estrategia consiste en realizar la búsqueda de la forma habitual durante un número acotado de iteraciones pero utilizando un valor más pequeño para el largo de la lista Tabú, el cuál varía aleatoriamente en el rango (4,8) (definido empíricamente por [3] y comprobado en la ejecución del algoritmo) cada vez que se inicia un proceso de intensificación.

Esta estrategia se utiliza cada vez que se cambia la vecindad guía de la búsqueda, reiniciándola a partir de algunas de las mejores soluciones obtenidas hasta ése momento.

5.2.10 Diversificación

La etapa de diversificación se realiza después de cierta cantidad de iteraciones sin mejora. Ésta consiste en realizar varios movimientos IPR consecutivos para obtener soluciones no factibles, las que luego son corregidas mediante los algoritmos de optimización local descritos en la sección "*5.2.7 Optimizaciones Locales*", lo que debiera resultar en la obtención de soluciones factibles a partir de este movimiento de diversificación.

De esta forma, como se muestra en la Figura 5.13, podemos alcanzar otras regiones de soluciones factibles del espacio de búsqueda, atravesando espacios de soluciones no factibles. Siendo ésta la forma que utiliza el algoritmo para alcanzar soluciones inalcanzables desde el punto de vista local y cumple su objetivo en pos de asegurar la búsqueda de mínimos globales.

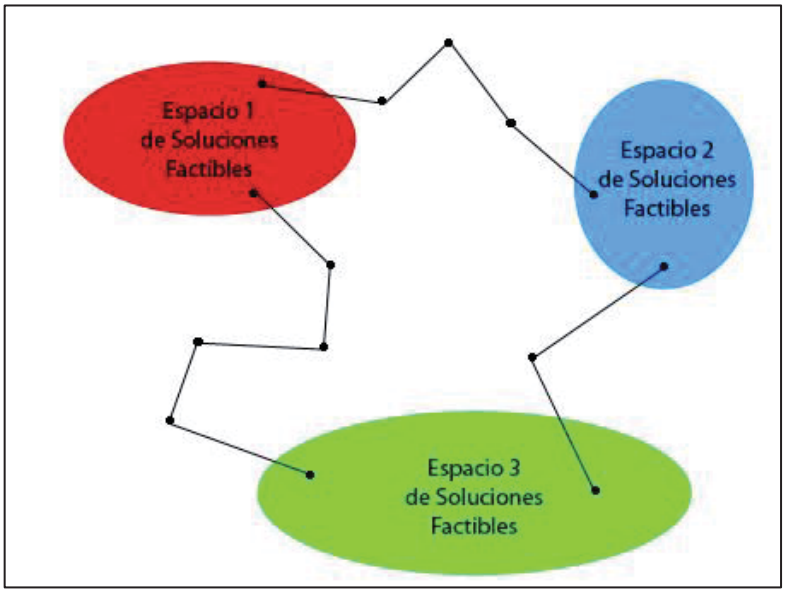

Figura 5.13 Diversificación de soluciones en el espacio de búsqueda

Además, hay que recalcar que el hecho de variar con cierta frecuencia la vecindad guía de la búsqueda (aunque algunas vecindades se utilicen con menor frecuencia y duración que otras) provoca que el proceso de búsqueda realice de forma indirecta pequeñas diversificaciones, lo que disminuye en gran medida que el algoritmo quede atrapado en mínimos locales.

Esquema Algoritmo de Diversificación (movimientos IPR)

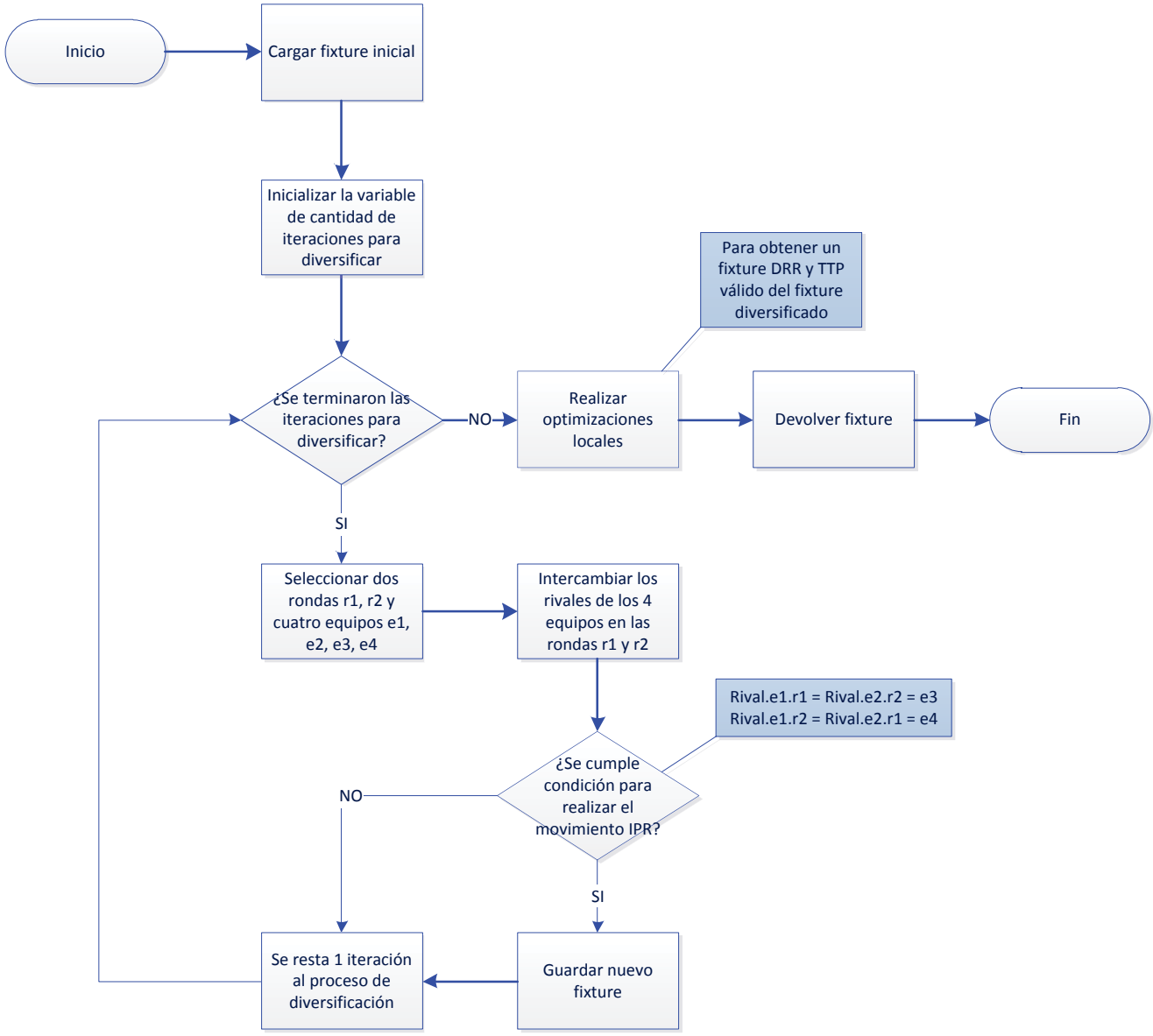

Figura 5.14 Diagrama del algoritmo de diversificación

5.2.11 Criterio de Término

El criterio de término utilizado por el algoritmo está relacionado con dos de los parámetros de entrada, uno de ellos es la cantidad máxima de iteraciones y el otro la cantidad admisible de iteraciones sin mejora.

Cuando se cumple cualquiera de los dos criterios mencionados, el algoritmo finaliza su ejecución y se entrega el fixture hallado junto con el valor obtenido al evaluarlo con la función objetivo tras la ejecución del algoritmo.

6 Diseño Experimento y Resultados

En este capítulo se presentan las distintas representaciones que se pueden encontrar en la literatura para el problema del TTP, además de las instancias utilizadas para probar el algoritmo y finalmente los resultados de las pruebas.

6.1 Representaciones

Las soluciones del problema TTP han sido representadas de múltiples formas, las que están basadas en el método seleccionado de solución del problema por parte de cada uno de los diferentes investigadores que ha abordado el problema.

6.1.1 Formato Anagnostopoulos

Es el formato estándar de representación de soluciones del TTP más utilizado. Su representación se basa en números enteros que representan a cada uno de los equipos, los que son utilizados en el encabezado de las filas de la matriz para señalarlos, las columnas que corresponden a las rondas y las celdas contienen a los oponentes, indicados como local o visita, en que los equipos visitantes corresponden a números enteros negativos.

Tabla 6.1 Instancia NL4 utilizando el Formato Anagnostopoulos

	R1	R2	R3	R4	R5	R6
E1	3	2	4	-3	-2	-4
E2	4	-1	-3	-4	1	3
E3	-1	4	2	1	-4	-2
E4	-2	-3	-1	2	3	1

6.1.2 Formato Trick

Es uno de los formatos estándar de representación de soluciones del TTP. Su representación se basa en las iniciales de los equipos, las que son utilizadas en el encabezado de las columnas de la matriz para señalar cada equipo, las rondas son los encabezados de las filas y las celdas contienen a los oponentes, indicadas como local o visita, en que los equipos visitantes tienen antepuesto el símbolo @.

Tabla 6.2 Instancia NL4 utilizando el Formato Trick

	R1	R2	R3	R4	R5	R6
E1	E3	E2	E4	@E3	@E2	@E4
E2	E4	@E1	@E3	@E4	E1	E3
E3	@E1	E4	E2	E1	@E4	@E2
E4	@E2	@E3	@E1	E2	E3	E1

6.1.3 Formato Urrutia

Este formato considera un formato de texto en vez de las matrices utilizadas por las otras representaciones, no es muy amigable a la vista por ser una línea consecutiva que indica los encuentros entre un equipo vs el otro indicando la localidad de cada uno de los equipos enfrentados.

6.1.4 Formato Di Gaspero

A diferencia del formato de Anagnostopoulos, Di Gaspero utiliza signos que indican la localia de los equipos, siendo positivo para oponentes enfrentados de local y negativo para oponentes enfrentados de visita. Además hay que notar que la numeración de los equipos parte desde 00 y no desde 1, aunque se conserva la notación de números enteros.

Tabla 6.3 Instancia NL4 utilizando el Formato Di Gaspero

	R1	R2	R3	R4	R5	R6
E1	+02	+01	+03	-02	-01	-03
E2	+03	-00	-02	-03	+00	+02
E3	-00	+03	+01	+00	-03	-01
E4	-01	-02	-00	+01	+02	+00

6.1.5 Cuadrados Latinos

La propuesta adicional de este trabajo fue probar el modelo con un nuevo formato de representación que se basa en cuadrados latinos, que consiste de una matriz de $n\,x\,n$ elementos, en que cada posición puede adoptar cualquier valor entero entre 1 y n con la condición de que ninguno aparezca más de una vez en la misma fila o columna.

La nueva representación reduce el espacio de búsqueda y puede tener como consecuencia el aumento de eficiencia sobre los algoritmos de búsqueda.

Se define M una matriz de soluciones donde cada elemento (i, j) corresponde a la ronda en la que juega el equipo Ti con el equipo Tj.

Las características relevantes de esta matriz, son las siguientes:

- La diagonal principal de la matriz es 0, debido a que los equipos no se enfrentan a ellos mismos.
- Para cada $(k, x) \Rightarrow k$ es local, $\forall x \neq k$.
- Para cada $(x, k) \Rightarrow k$ es visita, $\forall x \neq k$.

Tabla 6.4 Instancia NL4 utilizando Cuadrados Latinos

	E1	E2	E3	E4
E1	0	5	6	4
E2	2	0	1	6
E3	3	4	0	5
E4	1	3	2	0

6.2 Instancias

Se proponen dos clases de problemas para probar el algoritmo para el TTP. La primera es una instancia artificial para probar los aspectos TSP del TTP. La segunda son una serie de instancias de la *Major League Baseball*.

6.2.1 Instancias Circulares (CIRC)

Hay argumentos acerca de la complejidad de los problemas que el TTP resuelve como una versión más completa del *Traveling Salesman Problem* (TSP). Para lo que no es tan claro que el TTP sea sencillo, incluso si el TSP involucrado fuese trivial. Se exploraron instancias en las que el TSP es fácil de resolver (cuya solución es única), pero el TTP en cambio es desafiante.

El nodo n de la instancia circular (conocida como CIRC n) contiene distancias generadas por el grafo circular n con distancias unitarias. En este grafo, los nodos son etiquetados 0, 1,…., $n-1$; hay un camino desde i a $i+1$ y desde $n-1$ al nodo 0, cada una con distancia 1. La distancia desde i a j (con $i > j$) es la longitud del camino más corto en este grafo, e igualmente el mínimo de $i-j$ y $j-i+n$.

En este grafo, 0, 1,…., $n-1$ es el recorrido óptimo para el TSP. ¿Esto significa que el TTP será tan simple de resolver?

Se sugiere probar dos tipos de instancias por cada tamaño: una en que no hay restricciones respecto al HAP (*Home Away Pattern*) y otra con la restricción del TTP (no más de 3 partidos de local o visita). Las que son conocidas como CIRC *n constrained* (restrictiva) y *unconstrained* (no restrictiva) respectivamente.

6.2.2 Instancias de la National League (NL)

La definición de estas instancias fue un esfuerzo para encontrar tableros deportivos para la *Major League Baseball* (NL). Desafortunadamente, la MLB tiene demasiados equipos para el actual estado del arte de encontrar soluciones óptimas. La MLB se encuentra dividida en dos ligas: la *National League* (NL) y la *American League* (AL). La mayoría de los encuentros que disputa cada equipo son contra equipos correspondientes a su misma liga, así que es razonable acotar el análisis a ligas individuales. La *National League* está compuesta por 16 equipos y la *American League* por 14 equipos. Las ligas no se encuentran definidas geográficamente: cada liga tiene equipos en ambas costas y en el centro de Estados Unidos. Cada liga además contiene un equipo Canadiense (Montreal en la NL y Toronto en la AL).

Se generaron matrices de distancias para la *National League* utilizando "distancias aéreas" entre las ciudades. Esto, para generar instancias pequeñas en las que se seleccionaron algunos equipos. Para realizar esto, se crearon las instancias NL4, NL6, NL8, NL10, NL12, NL14 y NL16, en las que el número indica la cantidad de equipos que compite en la instancia.

6.3 Resultados

A continuación se presentan los resultados obtenidos, correspondientes a la ejecución del algoritmo diseñado. El objetivo es mostrar los distintos resultados obtenidos (tiempos de ejecución, óptimos obtenidos, valores y gráficos de desempeño), para poder establecer la línea y nivel de dificultad de problemas que pueden ser resueltos mediante este.

Primero revisaremos los resultados generales, para luego revisar por separado los correspondientes a cada notación y finalmente la comparación de los resultados de ambas notaciones.

6.3.1 Resultados Generales

La tabla y gráficos a continuación presentan la consolidación de los resultados correspondientes a la ejecución de 20 iteraciones de cada instancia (las que corresponden a la suma de las iteraciones de Anagnostopoulos y Cuadrados Latinos).

Tabla 6.5 Tabla consolidada de resultados algoritmo TTP

INSTANCIA	LÍMITE INFERIOR	LÍMITE SUPERIOR	PROMEDIO ITERACIONES	TIEMPO PROMEDIO	¿ÓPTIMO?
NL-4	8.276	8.276	8.276	8,33 seg	SI
NL-6	25.603	26.296	25.950	1,68 min	NO
NL-8	47.406	48.802	48.357	9,11 min	NO
NL-10	77.595	78.811	78.499	11,81 min	NO
NL-12	143.957	149.750	145.920	1,56 hrs	NO
CIRC-4	20	24	22	8,14 seg	SI
CIRC-6	70	74	72	1,42 min	NO
CIRC-8	132	180	164	7,14 min	SI
CIRC-10	240	354	316	11,39 min	SI
CIRC-12	424	588	522	1,20 hrs	SI

Los resultados obtenidos en la tabla anterior (óptimos, límites inferior y superior), corresponden a valores promedio de la ejecución de 20 iteraciones realizadas por instancia (10 iteraciones de cada notación: Anagnostopoulos y Cuadrados Latinos), en los que se puede apreciar aumento exponencial del tiempo de ejecución respecto a la cantidad de equipos que participan en cada instancia, lo que indica el incremento sustantivo en la complejidad del problema a medida que se agregan más equipos al análisis, razón por la que no se utilizaron instancias con más equipos: NL14, CIRC14, NL16, CIRC16 y CIRC20. El gráfico a continuación refleja de forma nítida la explicación.

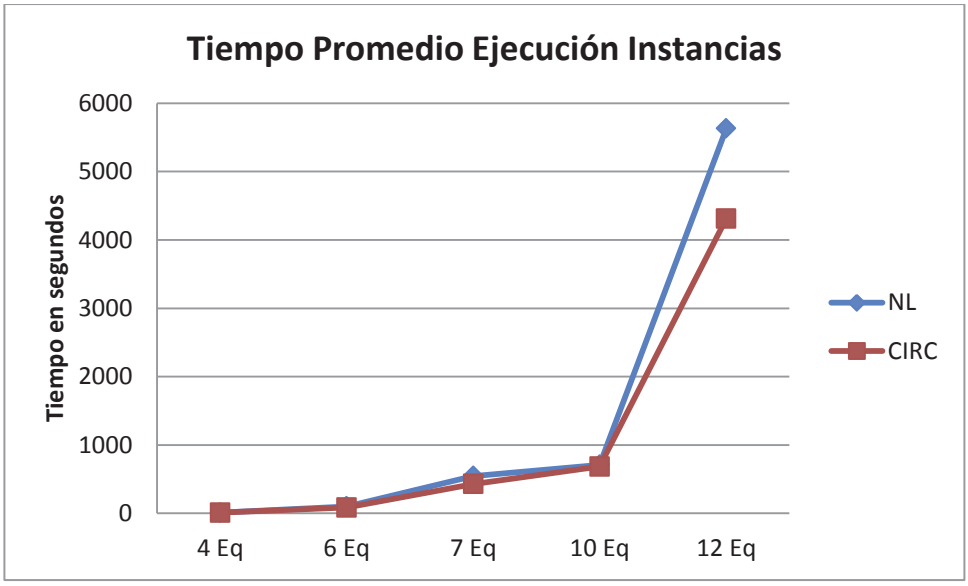

Figura 6.1 Gráfico comparativo tiempo promedio de ejecución instancias

6.3.2 Resultados Anagnostopoulos

Estas tablas presentan los resultados consolidados correspondientes a la ejecución de 10 iteraciones por cada instancia y cada notación.

Tabla 6.6 Tabla parámetros de ejecución algoritmo – Notación Anagnostopoulos

PARÁMETROS DE EJECUCIÓN	
Cantidad iteraciones	100.000
Iteraciones intensificación	2.000
Iteraciones diversificación	100
Tamaño Tabu List (TL)	intervalo [n-3, n+3], n corresponde al número de equipos
Tamaño TL intensificación	intervalo [4, 8]
Iterac. Mod tamaño TL	2.000
Secuencia vecindades	IE, IPR, IR, IPR, IL, IPR, IL

Tabla 6.7 Tabla resultados instancia NL – Notación Anagnostopoulos

Instancia	Proyecto	Literatura
NL4	8276	8276
NL6	25603	23978
NL8	47406	41505
NL10	77595	68691
NL12	146639	143655

Tabla 6.8 Tabla resultados instancia CIRC – Notación Anagnostopoulos

Instancia	Proyecto	Literatura
CIRC4	20	20
CIRC6	70	64
CIRC8	132	132
CIRC10	240	242
CIRC12	424	404

Se puede notar que utilizando esta notación se lograron prácticamente todos los valores óptimos señalados en las publicaciones especializadas, lo que se explica por la simplicidad del manejo de los datos y que las operaciones de vecindario sacan más provecho de este tipo de notación en particular.

6.3.3 Resultados Cuadrados Latinos

Estas tablas presentan los resultados consolidados correspondientes a la ejecución de 10 iteraciones por cada instancia y cada notación.

Tabla 6.9 Tabla parámetros de ejecución algoritmo – Notación Cuadrados Latinos

PARÁMETROS DE EJECUCIÓN	
Cantidad iteraciones	100.000
Iteraciones intensificación	2.000

Iteraciones diversificación	100
Tamaño Tabu List (TL)	intervalo [n-3, n+3], n corresponde al número de equipos
Tamaño TL intensificación	intervalo [4, 8]
Iterac. Mod tamaño TL	2.000
Secuencia vecindades	IE, IPR, IR, IPR, IL, IPR, IL

Tabla 6.10 Tabla resultados instancia NL – Notación Cuadrados Latinos

Instancia	Proyecto	Literatura
NL4	8276	8276
NL6	26296	23978
NL8	48802	41505
NL10	78811	68691
NL12	143957	143655

Tabla 6.11 Tabla resultados instancia CIRC – Notación Cuadrados Latinos

Instancia	Proyecto	Literatura
CIRC4	24	20
CIRC6	74	64
CIRC8	180	132
CIRC10	354	242
CIRC12	572	404

Si bien, con las instancias CIRC se lograron obtener buenas soluciones, no superan a las obtenidas con la notación Anagnostopoulos, debido a la forma en que están diseñadas las vecindades. Dicho factor es el que evita obtener el desempeño deseado para esta notación compacta y simple, debido a que la forma de manejo de los datos fácilmente puede llegar a comportarse con una complejidad alta de $O(n^4)$.

6.3.4 Comparación Resultados Ambas Notaciones

Cada notación tiene sus particularidades positivas como restricciones, lo que se hace más evidente por desde el punto de vista del análisis de complejidad del algoritmo operado con Cuadrados Latinos versus el que utiliza la notación de Anagnostopoulos.

Revisemos las gráficas que comparan el desempeño de las soluciones correspondientes a las distintas notaciones, para cada instancia:

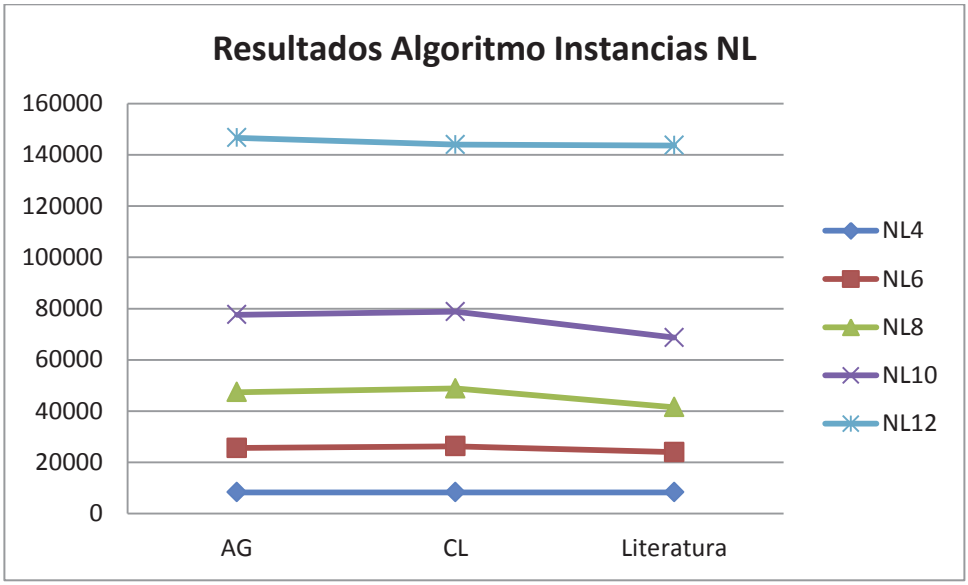

Figura 6.2 Gráfico comparativo de resultados del algoritmo instancias NL

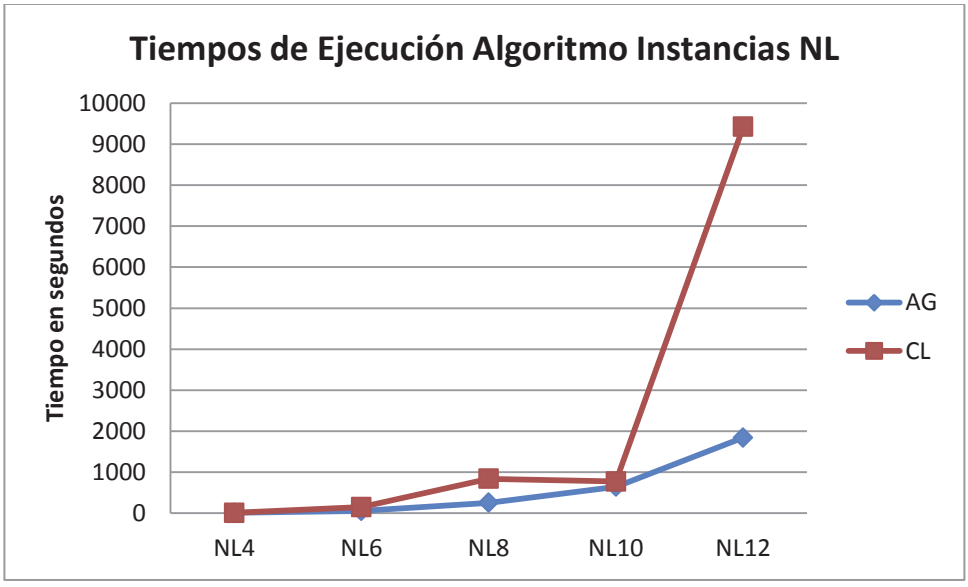

Figura 6.3 Gráfico comparativo de tiempos de ejecución del algoritmo instancias NL

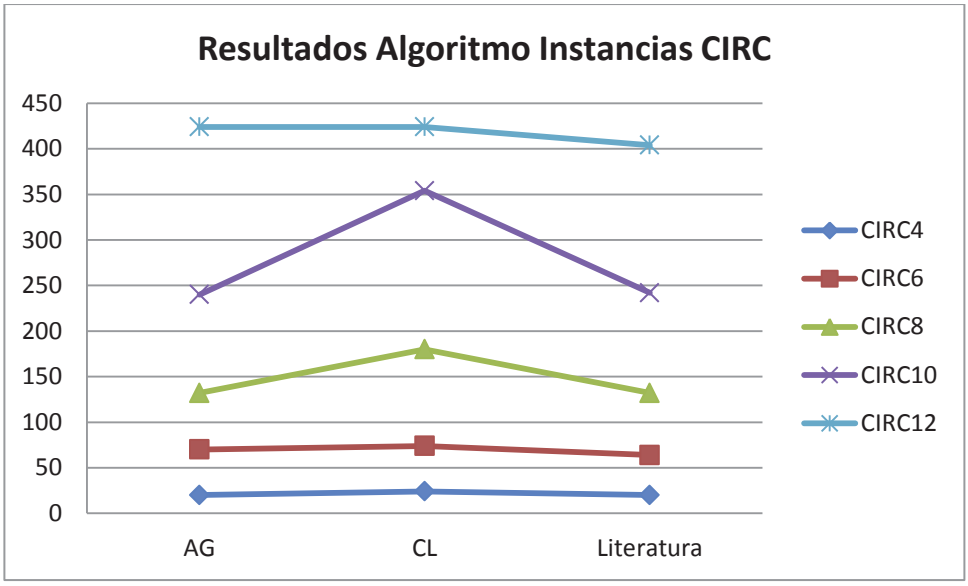

Figura 6.4 Gráfico comparativo de resultados del algoritmo instancias CIRC

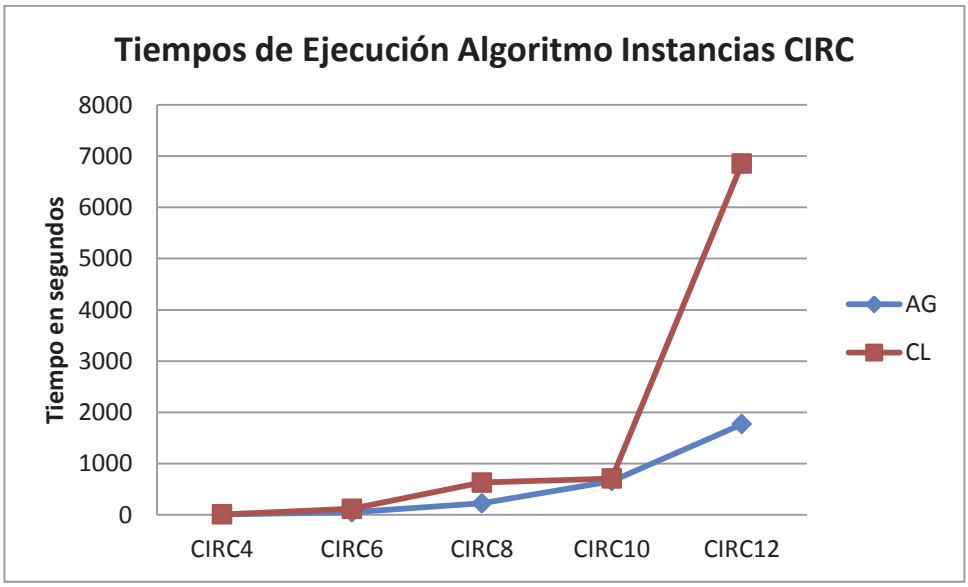

Figura 6.5 Gráfico comparativo de tiempos de ejecución del algoritmo instancias CIRC

Se reafirma el postulado que indica que la notación de Cuadrados Latinos es la que peor se comporta en cuanto al tiempo empleado y calidad para obtención de soluciones para cada instancia, mostrando una baja de desempeño notable en resultados al contrastarlo con la literatura o el algoritmo operando con la notación de Anagnostopoulos. Esto vuelve a indicar que el tipo de vecindades utilizadas no es el apropiado para el uso de la notación de Cuadrados Latinos. Aunque la baja de performance más notoria ocurre con las instancias circulares (CIRC).

7 Conclusiones

Este trabajo comprendió el estudio del TTP y del algoritmo heurístico *Tabu Search*, los que fueron descritos y desarrollados mediante el modelamiento y la resolución del problema desde el punto de vista teórico-práctico.

Se puede concluir que la investigación llevada a cabo para resolver el TTP mediante *Tabu Search* y las pruebas con el software de experimentación han resultado satisfactorias para probar que es posible encontrar tableros válidos de campeonatos deportivos y, que el algoritmo *Tabu Search* logra moverse cómodamente a través de la topología de los espacios de búsqueda de las diversas instancias utilizadas para resolver el problema y entregar soluciones de excelente calidad con tiempos no muy elevados y recursos computacionales relativamente corrientes, tomando en cuenta la dificultad que implica buscar buenas soluciones o el óptimo de la función objetivo de un problema NP-Completo.

Para realizar esta tarea, el algoritmo utiliza movimientos de vecindades para moverse dentro del espacio de búsqueda entre soluciones factibles, con la excepción de la rutina de diversificación, en que es necesario atravesar espacios de soluciones no factibles para encontrar espacios de soluciones factibles no explorados. La característica particular de este algoritmo es que se apoya en el uso de una lista tabú dinámica, que se comporta de forma especial cuando es necesario intensificar la búsqueda en un espacio de buenas soluciones.

Las vecindades son el motor de la búsqueda de soluciones y por ende la columna vertebral del algoritmo, por lo que se convierten en el factor crítico a tomar en cuenta para la mejora de los resultados actuales y el tiempo de ejecución. Al respecto es necesario tomar en cuenta que la complejidad de las operaciones que realiza la vecindad incide directamente en el tiempo de ejecución del algoritmo y la calidad de las soluciones que se obtienen. Por ejemplo, revisemos las siguientes vecindades: IR (Intercambio de Rondas) e IPR (Intercambio Parcial de Rondas); la vecindad IR opera con relativa simplicidad en ambas notaciones, lo que la hace una vecindad rápida de utilizar, pero que no consigue grandes mejoras sobre las soluciones encontradas; a diferencia de la anterior, la vecindad IPR es la más compleja de todas, lo que se traduce en más tiempo de proceso para encontrar una mejor solución y por ende es la vecindad más lenta, pero su lentitud se compensa con la calidad de soluciones que es capaz de encontrar, lo que puede significar mejoras interesantes respecto a la solución actual, además de destacar que es la única vecindad que se utiliza para las rutinas de diversificación.

Por lo tanto, el balance de ejecución de las vecindades del algoritmo es uno de los temas fundamentales y es por eso que el software de experimentación logró obtener excelentes resultados, alcanzando un error promedio de un 5% sobre las soluciones publicadas en la literatura, recalcando que varias de ellas alcanzaron valores óptimos.

Finalmente se puede establecer que el TTP es un problema complejo y que puede ser enfocado de distintas perspectivas de solución, por lo que se considera abierta la puerta para su investigación, para lo que se sugiere revisar los siguientes temas en trabajos futuros:

- Implementar el algoritmo *Tabu Search* mejorado i-TSAB.

- Generar estadísticas de búsqueda realizadas por el algoritmo:
    - Cantidad de iteraciones de intensificación.
    - Cantidad de iteraciones de diversificación.
    - Vecindad más utilizada.
- Implementar otras notaciones (Trick, Urrutia, Di gaspero) y ver el impacto que tienen en el desempeño del algoritmo.
- Utilizar este algoritmo en combinación con Algoritmos Genéticos. Por ejemplo, entre generaciones (antes de realizar la selección) puede utilizarse *Tabu Search* para mejorar las soluciones ya encontradas.
- Incorporar al algoritmo nuevas vecindades que ayuden a mejorar la precisión de la revisión del espacio de búsqueda. Sobre todo para la notación de Cuadrados Latinos.
- Incorporar nuevas restricciones o relajar las existentes y luego ver el impacto que tienen en el desempeño del algoritmo.

8 Referencias

[1] **Fred Glover..** *"Tabu Search"*, U.S. West Chair in Systems Science, Center for Applied Artificial Intelligence, University of Colorado. Año 1989.

[2] **Jean-Philippe Hamiez and Jin-Kao Hao..** *"Solving the Sports League Scheduling Problem with Tabu Search"*, LGI2P / Ecole des Mines d'Alès – EERIE Parc Scientifique Georges Besse – 30035 Nîmes Cedex 01 (France); LERIA / Université d'Angers – 2, Bd. Lavoisier – 49045 Angers Cedex 01 (France).

[3] **Andrés Cardemil, Guillermo Durán..** *"Un Algoritmo Tabu Search para el Traveling Tournament Problem"*, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos, Aires Buenos Aires, Argentina; Departamento de Ingeniería Industrial, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Santiago, Chile.

[4] **Kelly Easton, George Nemhauser, and Michael Trick..** *"The Traveling Tournament Problem Description and Benchmarks"*, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia USA, 30332; Graduate School of Industrial Administration, Carnegie Mellon, Pittsburgh, PA USA, 15213.

[5] **Luca di Gaspero and Andrea Schaerf..** *"A Composite-Neighborhood Tabu Search Approach to the Traveling Tournament Problem"*, Dipartimento di Ingegneria Elettrica, Gestionale e Meccanica, Universita di Udine, via delle Scienze 208, I-33100, Udine, Italy. Año 2006.

9 ANEXO A: Bibliografía Relacionada

1.- **Mieke Adriaen, Nele Custers, Greet Vanden Berghe..** *"An Agent Based Metaheuristic for the Travelling Tournament Problem"*, KaHo Sint-Lieven, Information Technology.

2.- **Ryuhei Miyashiro, Tomomi Matsui, Shinji Imahori..** *"An Approximation Algorithm for the Traveling Tournament Problem"*, METR 2008–42. Año 2008.

3.- **Graham Kendall, Wim Miserez, and Greet Vanden Berghe..** *"A Constructive Heuristic for the Travelling Tournament Problem"*, University of Nottingham, School of Computer Science and IT, Jubilee Campus, Nottingham NG8 1BB, UK.

4.- **Kelly Easton, George Nemhauser, and Michael Trick..** *"Solving the Traveling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach"*, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia USA, 30332.

5.- **Thierry Benoist, François Laburthe, Benoît Rottembourg..** *"Lagrange Relaxation and Constraint Programming Collaborative schemes for Traveling Tournament Problems"*, Bouygues eLab, 1 avenue Eugène Freyssinet, 78061 Saint Quentin-en-Yvelines Cedex, France.

6.- **Sebastián Urrutia, Celso C. Ribeiro, and Rafael A. Melo..** *"A New Lower Bound to the Traveling Tournament Problem"*.

7.- **Fabrício Lacerda Biajoli and Luiz Antonio Nogueira Lorena..** *"Mirrored Traveling Tournament Problem: An Evolutionary Approach"*, Instituto Nacional de Pesquisas Espaciais – INPE Laboratório Associado de Computaçâo e Matemática Aplicada – LAC Av. dos Astronautas, 1.758 - Sâo José dos Campos-SP, Brasil.

8.- **Glenn Langford..** *"An Improved Neighbourhood for the Traveling Tournament Problem"*, Año 2010.

9.- **A. Anagnostopoulos, L. Michel, P. Van Hentenryck, and Y. Vergados..** *"A Simulated Annealing Approach to the Travelling Tournament Problem"*,Brown University, Box 1910, Providence, RI 02912.

10.- **Alain Hertz, Eric Taillard, Dominique de Werra..** *"A Tutorial on Tabu Search"*, EPFL, Département de Mathématiques, MA-Ecublens, CH-1015 Lausanne; Université de Montréal, Centre de Recherche sur les Transports, Montréal, Canada H3C 3J7.

11.- **Pascal Van Hentenryck and Yannis Vergados..** *"Traveling Tournament Scheduling: A Systematic Evaluation of Simulated Annealling"*, Computer Science Department, Brown University, Providence, RI 02912, USA.

10 ANEXO B: Fixtures Obtenidos

Se listan algunos de los fixtures obtenidos por el algoritmo. Estos corresponden a las mejores soluciones obtenidas para cada una de las instancias utilizadas en este trabajo.

Tabla 10.1 Fixture instancia NL4 – Notación Anagnostopoulos

INSTANCIA NL4: → 8.276 ←	NOTACIÓN: ANAGNOSTOPOULOS
	- 2. - 4. - 1. 2. 4. 1.
	3. 1. 2. - 3. - 1. - 2.
	- 4. - 2. - 3. 4. 2. 3.
	1. 3. 4. - 1. - 3. - 4.

Tabla 10.2 Fixture instancia NL6 – Notación Anagnostopoulos

INSTANCIA NL6: → 25.603 ←	NOTACIÓN: ANAGNOSTOPOULOS
	- 5. - 1. 4. 2. - 4. 1. 5. 6. - 2. - 6.
	- 3. - 2. - 5. 3. 5. 2. - 6. - 1. 6. 1.
	- 1. - 3. - 6. 4. 6. 3. - 4. - 2. 1. 2.
	2. - 4. 1. - 5. - 1. 4. - 2. - 3. 5. 3.
	6. 5. 2. - 6. - 2. - 5. 3. 4. - 3. - 4.
	- 4. - 6. 3. 1. - 3. 6. - 1. - 5. 4. 5.

Tabla 10.3 Fixture instancia NL8 – Notación Anagnostopoulos

INSTANCIA NL8: → 47.406 ←	NOTACIÓN: ANAGNOSTOPOULOS
	- 2. - 1. - 8. 7. - 6. - 5. - 4. 2. 1. 8. - 7. 6. 5. 4.
	3. 2. 1. - 8. - 7. - 6. 5. - 3. - 2. - 1. 8. 7. 6. - 5.
	- 4. - 3. - 2. 1. - 8. - 7. - 6. 4. 3. 2. - 1. 8. 7. 6.
	5. 4. 3. - 2. - 1. - 8. 7. - 5. - 4. - 3. 2. 1. 8. - 7.
	- 6. - 5. - 4. 3. - 2. - 1. - 8. 6. 5. 4. - 3. 2. 1. 8.
	7. 6. 5. - 4. - 3. - 2. 1. - 7. - 6. - 5. 4. 3. 2. - 1.

- 8. - 7. - 6. 5. - 4. - 3. - 2. 8. 7. 6. - 5. 4. 3. 2.
1. 8. 7. - 6. - 5. - 4. 3. - 1. - 8. - 7. 6. 5. 4. - 3.

Tabla 10.4 Fixture instancia NL10 – Notación Anagnostopoulos

INSTANCIA NL10: → 77.595 ←	**NOTACIÓN: ANAGNOSTOPOULOS**
- 2. - 1. - 10. 9. - 8. - 7. - 6. 5. - 4. 2. 1. 10. - 9. 8. 7. 6. - 5. 4.	
3. 2. 1. - 10. - 9. - 8. 7. 6. 5. - 3. - 2. - 1. 10. 9. 8. - 7. - 6. - 5.	
- 4. - 3. - 2. 1. - 10. - 9. - 8. 7. - 6. 4. 3. 2. - 1. 10. 9. 8. - 7. 6.	
5. 4. 3. - 2. - 1. - 10. 9. 8. 7. - 5. - 4. - 3. 2. 1. 10. - 9. - 8. - 7.	
- 6. - 5. - 4. 3. - 2. - 1. - 10. 9. - 8. 6. 5. 4. - 3. 2. 1. 10. - 9. 8.	
7. 6. 5. - 4. - 3. - 2. 1. 10. 9. - 7. - 6. - 5. 4. 3. 2. - 1. - 10. - 9.	
- 8. - 7. - 6. 5. - 4. - 3. - 2. 1. - 10. 8. 7. 6. - 5. 4. 3. 2. - 1. 10.	
9. 8. 7. - 6. - 5. - 4. 3. 2. 1. - 9. - 8. - 7. 6. 5. 4. - 3. - 2. - 1.	
- 10. - 9. - 8. 7. - 6. - 5. - 4. 3. - 2. 10. 9. 8. - 7. 6. 5. 4. - 3. 2.	
1. 10. 9. - 8. - 7. - 6. 5. 4. 3. - 1. - 10. - 9. 8. 7. 6. - 5. - 4. - 3.	

Tabla 10.5 Fixture instancia NL12 – Notación Anagnostopoulos

INSTANCIA NL12: → 146.639 ←	**NOTACIÓN: ANAGNOSTOPOULOS**
- 10. - 9. - 4. 11. - 6. - 5. - 8. 3. - 2. - 1. - 12. 2. 9. 4. - 11. 6. 1. 8. - 3. 10. 5. 12.	
11. - 10. 5. - 12. - 7. - 6. 9. - 4. 3. - 2. - 1. - 3. 4. - 5. 12. 7. 2. - 9. 10. - 11. 6. 1.	
12. - 11. - 6. 1. - 8. - 7. - 10. 5. - 4. - 3. - 2. 4. 11. 6. - 1. 8. 3. 10. - 5. - 12. 7. 2.	
1. - 12. 7. - 2. - 9. - 8. 11. 6. 5. - 4. - 3. - 5. 12. - 7. 2. 9. 4. - 11. - 6. - 1. 8. 3.	
- 2. - 1. - 8. 3. - 10. - 9. - 12. 7. - 6. - 5. - 4. 6. 1. 8. - 3. 10. 5. 12. - 7. 2. 9. 4.	
3. - 2. - 3. - 4. 11. - 10. 1. 8. 7. - 6. - 5. - 7. 2. 9. 4. - 11. 6. - 1.	

- 8. - 9. 10. 5.
- 4. - 3. - 10. 5. - 12. - 11. - 2. 9. - 8. - 7. - 6. 8. 3. 10. - 5. 12. 7. 2. - 9. 4. 11. 6.
5. - 4. 11. - 6. - 1. - 12. 3. - 10. 9. - 8. - 7. - 9. 10. - 11. 6. 1. 8. - 3. 4. - 5. 12. 7.
6. - 5. - 12. 7. - 2. - 1. - 4. 11. - 10. - 9. - 8. 10. 5. 12. - 7. 2. 9. 4. - 11. - 6. 1. 8.
7. - 6. 1. - 8. - 3. - 2. 5. 12. 11. - 10. - 9. - 11. 6. - 1. 8. 3. 10. - 5. - 12. - 7. 2. 9.
- 8. - 7. - 2. 9. - 4. - 3. - 6. 1. - 12. - 11. - 10. 12. 7. 2. - 9. 4. 11. 6. - 1. 8. 3. 10.
9. - 8. - 9. - 10. 5. - 4. 7. 2. 1. - 12. - 11. - 1. 8. 3. 10. - 5. 12. - 7. - 2. - 3. 4. 11.

Tabla 10.6 Fixture instancia CIRC4 – Notación Anagnostopoulos

INSTANCIA CIRC4: → 20 ←	**NOTACIÓN: ANAGNOSTOPOULOS**
4. 3. 2. - 4. - 3. - 2.	
3. 4. - 1. - 3. - 4. 1.	
- 2. - 1. - 4. 2. 1. 4.	
- 1. - 2. 3. 1. 2. - 3.	

Tabla 10.7 Fixture instancia CIRC6 – Notación Anagnostopoulos

INSTANCIA CIRC6: → 70 ←	**NOTACIÓN: ANAGNOSTOPOULOS**
- 5. - 6. - 1. 5. 4. 1. - 2. 6. 2. - 4.	
3. 1. - 2. - 6. - 5. 2. - 3. - 1. 6. 5.	
- 4. - 2. - 3. 1. - 6. 3. 4. 2. - 1. 6.	
2. 3. - 4. - 2. - 1. 4. 5. - 3. - 5. 1.	
- 6. - 4. - 5. 3. - 2. 5. 6. 4. - 3. 2.	
1. 5. - 6. - 4. - 3. 6. - 1. - 5. 4. 3.	

Tabla 10.8 Fixture instancia CIRC8 – Notación Anagnostopoulos

INSTANCIA CIRC8: → 132 ←	NOTACIÓN: ANAGNOSTOPOULOS
- 1. - 7. - 8. 7. - 6. - 5. - 4. 2. 1. 8. - 2. 6. 5. 4.	
2. 8. - 5. - 8. - 7. 6. 5. - 3. - 2. - 1. 3. 7. - 6. 1.	
- 3. - 1. - 2. 1. - 8. - 7. - 6. 4. 3. 2. - 4. 8. 7. 6.	
4. 2. 3. - 2. - 1. - 8. 7. - 5. - 4. - 3. 5. 1. 8. - 7.	
- 5. - 3. - 4. 3. - 2. - 1. - 8. 6. 5. 4. - 6. 2. 1. 8.	
6. 4. - 1. - 4. - 3. 2. 1. - 7. - 6. - 5. 7. 3. - 2. 5.	
- 7. - 5. - 6. 5. - 4. - 3. - 2. 8. 7. 6. - 8. 4. 3. 2.	
8. 6. 7. - 6. - 5. - 4. 3. - 1. - 8. - 7. 1. 5. 4. - 3.	

Tabla 10.9 Fixture instancia CIRC10 – Notación Anagnostopoulos

INSTANCIA CIRC10: → 240 ←	NOTACIÓN: ANAGNOSTOPOULOS
- 1. - 9. - 10. 9. - 8. - 7. - 6. 5. - 4. 2. 1. 10. - 2. 8. 7. 6. - 5. 4.	
2. 10. 1. - 10. - 9. - 8. 7. 6. 5. - 3. - 2. - 1. 3. 9. 8. - 7. - 6. - 5.	
- 3. - 1. - 2. 1. - 10. - 9. - 8. 7. - 6. 4. 3. 2. - 4. 10. 9. 8. - 7. 6.	
4. 2. 3. - 2. - 1. - 10. 9. 8. 7. - 5. - 4. - 3. 5. 1. 10. - 9. - 8. - 7.	
- 5. - 3. - 4. 3. - 2. - 1. - 10. 9. - 8. 6. 5. 4. - 6. 2. 1. 10. - 9. 8.	
6. 4. 5. - 4. - 3. - 2. 1. 10. 9. - 7. - 6. - 5. 7. 3. 2. - 1. - 10. - 9.	
- 7. - 5. - 6. 5. - 4. - 3. - 2. 1. - 10. 8. 7. 6. - 8. 4. 3. 2. - 1. 10.	
8. 6. 7. - 6. - 5. - 4. 3. 2. 1. - 9. - 8. - 7. 9. 5. 4. - 3. - 2. - 1.	
- 9. - 7. - 8. 7. - 6. - 5. - 4. 3. - 2. 10. 9. 8. - 10. 6. 5. 4. - 3. 2.	
10. 8. 9. - 8. - 7. - 6. 5. 4. 3. - 1. - 10. - 9. 1. 7. 6. - 5. - 4. - 3.	

Tabla 10.10 Fixture instancia CIRC12 – Notación Anagnostopoulos

INSTANCIA CIRC12: → 424 ←	NOTACIÓN: ANAGNOSTOPOULOS
- 2. - 11. - 12. 11. - 10. - 9. - 8. 7. - 6. - 5. - 4. 1. 2. 12. - 1. 10. 9. 8. - 7. 6. 5. 4.	

- 3. 12. 1. - 12. - 11. - 10. 3. 8. 7. - 6. - 5. 2. 9. - 1. - 2. 11. 10. - 9. - 8. - 7. 6. 5.
- 4. - 1. - 2. 1. - 12. - 11. - 10. 9. - 8. - 7. - 6. 3. 4. 2. - 3. 12. 11. 10. - 9. 8. 7. 6.
5. 2. 3. - 2. - 1. - 12. 11. 10. 9. - 8. - 7. 4. - 5. - 3. - 4. 1. 12. - 11. - 10. - 9. 8. 7.
- 6. - 3. - 4. 3. - 2. - 1. - 12. 11. - 10. - 9. - 8. 5. 6. 4. - 5. 2. 1. 12. - 11. 10. 9. 8.
7. 4. 5. - 4. - 3. - 2. 1. 12. 11. - 10. - 9. 6. - 7. - 5. - 6. 3. 2. - 1. - 12. - 11. 10. 9.
- 8. - 5. - 6. 5. - 4. - 3. - 2. 1. - 12. - 11. - 10. 7. 8. 6. - 7. 4. 3. 2. - 1. 12. 11. 10.
- 9. 6. 7. - 6. - 5. - 4. 9. 2. 1. - 12. - 11. 8. 3. - 7. - 8. 5. 4. - 3. - 2. - 1. 12. 11.
- 10. - 7. - 8. 7. - 6. - 5. - 4. 3. - 2. - 1. - 12. 9. 10. 8. - 9. 6. 5. 4. - 3. 2. 1. 12.
11. 8. 9. - 8. - 7. - 6. 5. 4. 3. - 2. - 1. 10. - 11. - 9. - 10. 7. 6. - 5. - 4. - 3. 2. 1.
- 12. - 9. - 10. 9. - 8. - 7. - 6. 5. - 4. - 3. - 2. 11. 12. 10. - 11. 8. 7. 6. - 5. 4. 3. 2.
1. 10. 11. - 10. - 9. - 8. 7. 6. 5. - 4. - 3. 12. - 1. - 11. - 12. 9. 8. - 7. - 6. - 5. 4. 3.

Tabla 10.11 Fixture instancia NL4 – Notación Cuadrados Latinos

INSTANCIA NL4: → 8.276 ←	**NOTACIÓN: CUADRADOS LATINOS**
	0. 5. 6. 4.
	2. 0. 1. 6.
	3. 4. 0. 5.
	1. 3. 2. 0.

Tabla 10.12 Fixture instancia NL6 – Notación Cuadrados Latinos

INSTANCIA NL6: → 26.296 ←	**NOTACIÓN: CUADRADOS LATINOS**

0.	**3.**	**6.**	**7.**	**4.**	**2.**
8.	**0.**	**10.**	**1.**	**5.**	**6.**
9.	**7.**	**0.**	**2.**	**8.**	**4.**
10.	**4.**	**5.**	**0.**	**9.**	**3.**
1.	**2.**	**3.**	**6.**	**0.**	**7.**
5.	**9.**	**1.**	**8.**	**10.**	**0.**

Tabla 10.13 Fixture instancia NL8 – Notación Cuadrados Latinos

INSTANCIA NL8: → 48.802 ←							**NOTACIÓN: CUADRADOS LATINOS**
0.	**6.**	**5.**	**10.**	**14.**	**9.**	**4.**	**1.**
13.	**0.**	**3.**	**11.**	**9.**	**7.**	**8.**	**12.**
12.	**10.**	**0.**	**2.**	**7.**	**8.**	**6.**	**11.**
3.	**14.**	**9.**	**0.**	**8.**	**13.**	**5.**	**4.**
11.	**2.**	**4.**	**1.**	**0.**	**5.**	**10.**	**6.**
2.	**4.**	**1.**	**6.**	**12.**	**0.**	**14.**	**10.**
7.	**1.**	**13.**	**12.**	**3.**	**11.**	**0.**	**2.**
8.	**5.**	**14.**	**7.**	**13.**	**3.**	**9.**	**0.**

Tabla 10.14 Fixture instancia NL10 – Notación Cuadrados Latinos

INSTANCIA NL10: → 78.811 ←									**NOTACIÓN: CUADRADOS LATINOS**
0.	**1.**	**16.**	**9.**	**10.**	**12.**	**2.**	**13.**	**6.**	**14.**
5.	**0.**	**2.**	**8.**	**18.**	**9.**	**13.**	**6.**	**17.**	**3.**
18.	**4.**	**0.**	**12.**	**7.**	**1.**	**6.**	**14.**	**13.**	**8.**
4.	**10.**	**3.**	**0.**	**13.**	**15.**	**14.**	**5.**	**18.**	**7.**
8.	**16.**	**9.**	**11.**	**0.**	**17.**	**1.**	**12.**	**4.**	**15.**
3.	**7.**	**11.**	**6.**	**14.**	**0.**	**18.**	**2.**	**8.**	**10.**
7.	**11.**	**15.**	**17.**	**5.**	**16.**	**0.**	**10.**	**12.**	**4.**

11.	**15.**	**17.**	**16.**	**3.**	**4.**	**8.**	**0.**	**7.**	**1.**
15.	**14.**	**10.**	**1.**	**2.**	**5.**	**3.**	**9.**	**0.**	**16.**
17.	**12.**	**5.**	**2.**	**6.**	**13.**	**9.**	**18.**	**11.**	**0.**

Tabla 10.15 Fixture instancia NL12 – Notación Cuadrados Latinos

INSTANCIA NL12: → 143.957 ←	**NOTACIÓN: CUADRADOS LATINOS**

0.	**6.**	**11.**	**5.**	**21.**	**15.**	**20.**	**1.**	**19.**	**13.**	**7.**	**14.**
17.	**0.**	**16.**	**10.**	**4.**	**20.**	**3.**	**8.**	**2.**	**22.**	**12.**	**11.**
18.	**5.**	**0.**	**15.**	**9.**	**1.**	**19.**	**2.**	**7.**	**14.**	**6.**	**10.**
16.	**21.**	**4.**	**0.**	**3.**	**8.**	**2.**	**22.**	**12.**	**17.**	**11.**	**9.**
10.	**15.**	**20.**	**1.**	**0.**	**13.**	**7.**	**14.**	**6.**	**18.**	**5.**	**19.**
4.	**9.**	**3.**	**19.**	**2.**	**0.**	**14.**	**17.**	**11.**	**16.**	**10.**	**7.**
9.	**1.**	**8.**	**13.**	**22.**	**12.**	**0.**	**18.**	**5.**	**21.**	**15.**	**6.**
3.	**19.**	**13.**	**7.**	**12.**	**6.**	**11.**	**0.**	**21.**	**4.**	**20.**	**16.**
8.	**13.**	**22.**	**14.**	**17.**	**18.**	**16.**	**10.**	**0.**	**9.**	**3.**	**4.**
2.	**7.**	**12.**	**6.**	**11.**	**5.**	**10.**	**15.**	**20.**	**0.**	**19.**	**3.**
22.	**14.**	**17.**	**18.**	**16.**	**21.**	**4.**	**9.**	**1.**	**8.**	**0.**	**13.**
12.	**18.**	**21.**	**20.**	**8.**	**22.**	**17.**	**5.**	**15.**	**1.**	**2.**	**0.**

Tabla 10.16 Fixture instancia CIRC4 – Notación Cuadrados Latinos

INSTANCIA CIRC4: → 24 ←	**NOTACIÓN: CUADRADOS LATINOS**

0.	**5.**	**6.**	**1.**
2.	**0.**	**1.**	**6.**
3.	**4.**	**0.**	**5.**
4.	**3.**	**2.**	**0.**

Tabla 10.17 Fixture instancia CIRC6 – Notación Cuadrados Latinos

INSTANCIA CIRC6: → 74 ←	**NOTACIÓN: CUADRADOS LATINOS**

0. 3. 2. 7. 4. 9.
8. 0. 5. 4. 9. 10.
10. 7. 0. 1. 8. 4.
5. 6. 9. 0. 10. 3.
6. 1. 3. 2. 0. 7.
1. 2. 6. 8. 5. 0.

Tabla 10.18 Fixture instancia CIRC8 – Notación Cuadrados Latinos

INSTANCIA CIRC8: → 180 ←	**NOTACIÓN: CUADRADOS LATINOS**
0. 13. 14. 10. 6. 2. 5. 1.	
4. 0. 3. 11. 8. 12. 9. 7.	
7. 10. 0. 2. 5. 1. 4. 11.	
3. 6. 8. 0. 9. 13. 14. 12.	
11. 2. 12. 1. 0. 7. 10. 4.	
8. 5. 9. 4. 14. 0. 6. 10.	
12. 1. 13. 7. 3. 11. 0. 2.	
9. 14. 6. 5. 13. 3. 8. 0.	

Tabla 10.19 Fixture instancia CIRC10 – Notación Cuadrados Latinos

INSTANCIA CIRC10: → 354 ←	**NOTACIÓN: CUADRADOS LATINOS**
0. 14. 9. 4. 17. 3. 7. 11. 1. 8.	
5. 0. 13. 6. 3. 15. 2. 1. 10. 18.	
18. 4. 0. 12. 7. 2. 1. 8. 14. 6.	
13. 17. 3. 0. 11. 16. 8. 5. 18. 7.	
6. 12. 15. 2. 0. 10. 5. 9. 4. 1.	
12. 7. 11. 1. 8. 0. 9. 4. 17. 14.	
15. 11. 16. 10. 14. 18. 0. 6. 12. 4.	

2.	**16.**	**10.**	**14.**	**18.**	**13.**	**17.**	**0.**	**7.**	**3.**	
16.	**8.**	**5.**	**9.**	**13.**	**6.**	**3.**	**15.**	**0.**	**11.**	
10.	**9.**	**17.**	**15.**	**16.**	**5.**	**13.**	**12.**	**2.**	**0.**	

Tabla 10.20 Fixture instancia CIRC12 – Notación Cuadrados Latinos

INSTANCIA CIRC12: → 572 ←											**NOTACIÓN: CUADRADOS LATINOS**
0.	**6.**	**11.**	**5.**	**21.**	**15.**	**20.**	**14.**	**19.**	**13.**	**7.**	**1.**
17.	**0.**	**16.**	**10.**	**4.**	**9.**	**14.**	**8.**	**13.**	**22.**	**12.**	**18.**
18.	**5.**	**0.**	**15.**	**20.**	**3.**	**19.**	**13.**	**7.**	**12.**	**6.**	**10.**
16.	**21.**	**4.**	**0.**	**14.**	**8.**	**2.**	**22.**	**1.**	**17.**	**11.**	**20.**
10.	**15.**	**9.**	**3.**	**0.**	**13.**	**7.**	**1.**	**6.**	**18.**	**5.**	**19.**
4.	**20.**	**14.**	**19.**	**2.**	**0.**	**1.**	**17.**	**11.**	**16.**	**10.**	**7.**
9.	**3.**	**8.**	**13.**	**22.**	**12.**	**0.**	**18.**	**5.**	**10.**	**15.**	**6.**
3.	**19.**	**2.**	**7.**	**12.**	**6.**	**11.**	**0.**	**21.**	**4.**	**20.**	**16.**
8.	**2.**	**22.**	**12.**	**17.**	**18.**	**16.**	**10.**	**0.**	**9.**	**3.**	**4.**
2.	**7.**	**1.**	**6.**	**11.**	**5.**	**21.**	**15.**	**20.**	**0.**	**19.**	**3.**
22.	**1.**	**17.**	**18.**	**16.**	**21.**	**4.**	**9.**	**14.**	**8.**	**0.**	**13.**
12.	**11.**	**21.**	**9.**	**8.**	**22.**	**17.**	**5.**	**15.**	**14.**	**2.**	**0.**