Estudi d'algorismes d'optimització basats en colònia de formigues per a la resolució del problema ACSP: implementació d'un nou prototipus d'algorisme ACO.

Esteve Mir Espàrrech
Grau en Enginyeria Informàtica – Itinerari Computació

David Isern Alarcón

23 Desembre 2015

FITXA DEL TREBALL FINAL

Títol del treball:	*Estudi d'algorismes d'optimització basats en colònia de formigues per a la resolució del problema ACSP: implementació d'un nou prototipus d'algorisme ACO.*
Nom de l'autor:	*Esteve Mir Espàrrech*
Nom del consultor:	*David Isern Alarcón*
Data de lliurament (mm/aaaa):	*23/2015*
Àrea del Treball Final:	*Computació – Intel·ligència Artificial*
Titulació:	*Grau en Enginyeria Informàtica*

Resum del Treball (màxim 250 paraules):

El desenvolupament d'aquest projecte es fonamenta en dues idees principals: oferir una visió general i informativa de l'estat actual de la intel·ligència computacional d'eixam i del potencial que ofereixen els mecanismes basats en el comportament col·lectiu de sistemes naturals, i alhora realitzar una comparativa entre dos algorismes de colònia de formigues diferents envers la resolució del problema complex d'optimització per restriccions ACSP.

Per a realitzar aquest estudi comparatiu s'implementaran dos algorismes ACO diferents: l'algorisme Max-Min Ant System, reconegut pel seu rendiment i capacitat de cerca de múltiples solucions, i un nou algorisme desenvolupat especialment per a la resolució del problema ACSP, que s'anomena Forgetful Ant System, i que utilitza un nou mecanisme d'oblit de nodes per a evitar casos d'estancament en la construcció de les solucions.

La metodologia de treball i el disseny del nou algorisme estan basats en els resultats de l'estudi dut a terme per Guang-Feng Deng i Woo-Tsong Lin a *Ant Colony Optimization-based algorithm for airline crew scheduling problem,* on el problema ACSP és resolt a través de la seva transformació en un problema d'optimització per restriccions i cerca de camins mínims derivat del problema conegut com a Traveling Salesman Problem.

Abstract (in English, 250 words or less):

This project has been developed starting on two main ideas: provide an informative overview of the state of the art in computational swarm intelligence and of the inherent potential shown by new mechanisms based on collective behavior of natural systems, and also perform a comparison between two different ant colony algorithms towards ACSP complex constrained optimization problem solving.

In order to carry out this comparative study, two different ACO algorithms will be implemented: the Max-Min Ant System, renowned for its performance and ability to search for multiple solutions and a new algorithm especially developed to solve the ACSP problem, called Forgetful Ant System, which uses a new node forget mechanism to avoid stagnation cases during solution construction processes.

Methodology and new algorithm design are based on the results of the study conducted by Guang Feng Deng and Lin Tsong-Woo in *Ant Colony Optimization-based algorithm for airline crew scheduling problem*, where ACSP problem is solved through its transformation to a search for the minimum path and constrained optimization problem derived from the problem known as Traveling Salesman.

Paraules clau (entre 4 i 8):

Ant Colony; Optimization; Airline Crew Scheduling Problem; Swarm Intelligence;

Agraïments

A tota la meva família i a la meva parella per haver-me acompanyat sempre i en especial aquests últims anys d'estudi. Pel suport, els ànims i l'estima. Mariona, ja en som dos! Txell, esperem veure un TFG d'aquí poc...

A tots els meus companys i amics per l'interès mostrat en el projecte i per les qüestions inestimables sobre la integració de les formigues en l'àmbit informàtic...

Al David Isern, pel temps dedicat, els consells i tot el procés de guiatge que ha fet possible la realització d'aquest projecte.

Moltes gràcies a tots!

Índex

Índex de figures

Índex de taules

1. Introducció

1.1 Context i justificació del Treball

L'objecte principal d'aquest treball és realitzar la implementació d'un nou algorisme de tipus ACO[1] a través de l'estudi del funcionament i comportament d'algorismes meta-heurístics en la resolució del problema d'optimització ACSP[2]. Com a mesura de la bonança dels resultats s'utilitzarà el rendiment[3].

Els algorismes que s'utilitzaran formen part de la família de mètodes coneguts com a colònia de fòrmigues que, al seu torn, es basen en tècniques de la branca d'intel·ligència artificial anomenada intel·ligència d'eixam. Aquesta centra el seu estudi en el comportament col·lectiu de sistemes naturals o artificials que tenen en comú la descentralització i la auto-organització.

Les tècniques dels algorismes ACO es basen en la utilització d'agents simples anomenats formigues que, de forma individual, construeixen solucions candidates per a problemes complexos d'optimització en combinatòria. Per cada solució o camí construït, l'agent marca aquest camí amb senyalitzadors indirectes anomenats feromones. La decisió de quina direcció escollir en cada cas es determina en funció de diferents paràmetres de l'algorisme. En general, i principalment en les modalitats més simples, es considera el camí més concorregut com la millor solució del problema. La no existència d'una garantia pel que fa a la troballa del camí òptim condueix a la utilització de diversos mètodes de cerca local en les interseccions on els agents han de prendre les decisions.

Per aquest treball s'ha escollit de forma específica l'algorisme MMAS[4] com a punt de partida, tant per a la seva representativitat en el marc evolutiu d'aquest tipus de tècniques probabilístiques com per la millora del rendiment i de les capacitats de cerca que ofereix respecte als algorismes AS originals. Tot i que més endavant es detallaran les seves propietats i particularitats, a continuació es concreten alguns dels seus trets distintius:

- **MMAS**: Implementa diverses millores en relació a l'algorisme base AS[5]; la més significativa és la restricció en el dipòsit de feromones. En cada iteració s'estableix un interval $[min, max]$ i només es permet dipositar feromones a aquells agents amb una solució vàlida dins el mateix [2].

[1] Airline Crew Scheduling Problem (ACSP).
[2] Ant COlony Optimization (ACO).
[3] Per a mesurar l'eficàcia s'utilitzarà el sistema MBF o *mean best fitness*.
[4] Max-Min Ant System (MMAS).
[5] Ant System (AS).

L'ACSP és un exemple de problema d'optimització en què s'intenta buscar la distribució òptima dels membres de les tripulacions de vols d'aero-línies per tal de reduir els costos derivats dels seus sous. L'objectiu és minimitzar aquestes despeses satisfent alhora els requisits de planificació, rutes establertes, polítiques laborals, regulacions governamentals i polítiques pròpies de cada empresa. Guang-Feng Deng i Woo-Tsong Lin [1] demostren el potencial dels algorismes tipus ACO envers les aproximacions SCP[1] o SPP[2] per a la resolució d'aquest problema.

1.2 Motivació

En el context de creixent desenvolupament tecnològic i científic que viu la nostra societat avui dia, la intel·ligència artificial és una de les àrees de coneixement que té un impacte directe o indirecte més important en àmbits tan diversos com les finances, la indústria o la investigació, entre d'altres. És també aquesta àrea multidisciplinària que engloba camps tant diversos com la filosofia, les matemàtiques i les ciències de la computació, la que ofereix un ventall de possibilitats que van des del desenvolupament de sistemes d'ús quotidià fins a l'imaginari de ciència ficció.

En aquest sentit, són els algorismes que imiten el comportament de sistemes naturals complexos els que sovint demostren com algunes conductes que a priori semblen aleatòries, es converteixen en cooperatives i intel·ligents, centrades en un propòsit concret. Dels que s'utilitzen i encara s'estudien avui dia, els algorismes de colònia de formigues en són un exponent clar, que demostra que partint d'una metodologia de comportament col·lectiu a primera vista senzilla es poden obtenir resultats viables per a problemes inherentment difícils.

Partint d'aquesta base, la motivació principal d'aquest projecte de final de grau és estudiar la capacitat d'aquests sistemes artificials per a resoldre problemes complexos i aconseguir millorar o optimitzar-ne el funcionament a través de la implementació d'un nou algorisme derivat. La realització d'aquest estudi es durà a terme contraposant els algorismes presentats en la resolució d'un dels problemes d'optimització complexos encara vigent i representatiu de la magnitud d'aquest tipus de casos, el problema ACSP.

Per a aquest estudi concret s'han tingut en compte diversos articles d'intel·ligència artificial centrats en la investigació de les capacitats dels algorismes ACO per a la resolució de casos amb característiques similars. Principalment l'article de Guang-Feng Deng i Woo-Tsong Lin [1] sobre les

[1] Set Cover Problem (SCP).
[2] Set Partition Problem (SPP).

possibilitats d'utilització d'algorismes ACO en el mateix problema ACSP, i l'article de Abbas Afshar, Fariborz Massoumi, Amin Afshar i Miguel A. Mariño [5], que analitza l'estat actual de la utilització dels algorismes de colònia de formigues en sistemes de gestió de recursos hídrics.

La intenció final és poder comparar el comportament dels algorismes enfront un problema establert, indicant les seves febleses/fortaleses en funció de diferents paràmetres i metodologies de comportament.

1.3 Objectius

Amb el desenvolupament d'aquest treball es pretenen assolir els següents objectius:

- Exposar els conceptes generals de la intel·ligència artificial i la intel·ligència computacional.
- Introduir i exposar els conceptes d'intel·ligència d'eixam en el marc dels algorismes d'optimització i la seva utilització en la resolució de problemes d'optimització en combinatòria complexos.
- Analitzar l'evolució dels algorismes ACO i l'estat de l'art.
- Desenvolupar un algorisme ACO derivat per a la resolució del problema d'optimització ACSP.
- Detallar el funcionament i les característiques dels algorismes utilitzats.
- Implementar els prototips per a realitzar els dos experiments.
- Comparar el rendiment i l'eficàcia dels algorismes presentats en la resolució del problema ACSP.
- Exposar les conclusions obtingudes dels experiments.

1.4 Metodologia de treball

Guang-Feng Deng i Woo-Tsong Lin analitzen [1, pag. 4] el rendiment d'un algorisme ACO en funció de les diferents configuracions de paràmetres de les instàncies en el problema ACSP. Els paràmetres són:

- Límit d'exploració pels agents

- La importància relativa entre els camins de feromones i la funció heurística del propi algorisme ACO
- El *coeficient de persistència de rastreig* ρ o coeficient de duració del rastre de feromones
- El nombre d'agents

La influència d'aquests paràmetres és estudiada en funció de tres criteris:

- El temps de CPU requerit per a arribar a una solució
- El nombre d'iteracions de l'algorisme
- El resultat

Aquests són calculats amb la mesura de l'MBF, que utilitza la mitjana obtinguda de la realització de *n* execucions.

En aquest treball s'utilitzarà aquesta metodologia per a mesurar la bonança dels resultats obtinguts i per a dur a terme la comparativa entre els algorismes presentats. En la fase d'experimentació es detallaran de forma específica els paràmetres, l'entorn d'execució i el procediment seguit.

1.5 Relació de tasques

La planificació de les diverses etapes del projecte s'ha dut a terme tenint en compte la possible dificultat de cadascuna, el temps total disponible i una distribució setmanal. A continuació es presenta un resum de les tasques que s'han realitzat durant tot el procés:

1. Definició del context del treball i justificació
2. Definició d'objectius
3. Breu descripció de la metodologia de treball
4. Planificació i distribució de tasques
5. Descripció de les tasques
6. Intel·ligència artificial - computacional. Presentació i conceptes generals.
    a. Documentació
    b. Redacció de continguts
        1. Descripció general
        2. Tipus d'algorismes
        3. Aplicacions
        4. *State of the art*
    c. Revisió i modificació

7. Problemes d'optimització. Algorismes d'optimització amb colònies de formigues (ACO). Meta-heurística. Conceptes generals.
    a. Documentació
    b. Redacció de continguts
        1. Descripció general
        2. Tipus d'algorismes
        3. Aplicacions
        4. *State of the art*
    c. Revisió i modificació
8. Presentació i definició dels experiments
    a. Problema ACSP – Airline Crew Scheduling Problem
        i. Documentació
        ii. Redacció de continguts
            1. Bases generals
            2. Implementació
        iii. Revisió i modificació
    b. Plantejament de la metodologia
        i. Documentació
        ii. Redacció de continguts
            1. Mesures de rendiment/MBF
            2. Selecció entorn de proves / Dades inicials problema
        iii. Revisió i modificació
9. Algorisme MMAS
    a. Documentació
    b. Redacció de continguts
        1. Breu història i aplicacions
        2. Característiques
        3. Funcionament
        4. Implementació prototip
        5. Fase de proves/Establiment paràmetres generals
    c. Revisió i modificació
10. Presentació prototipus alternatiu-derivat
    a. Documentació
    b. Redacció de continguts

1. Característiques
2. Funcionament
3. Implementació prototip
4. Fase de proves/Establiment paràmetres generals

c. Revisió i modificació

11. Fase d'experimentació

a. Configuració entorn de proves

b. Execució dels prototips

c. Obtenció, presentació i redacció de resultats

d. Anàlisi dels resultats / Redacció conclusions

e. Revisió dels continguts

12. Presentació

a. Disseny

b. Redacció / creació

c. Revisió

13. Revisió del projecte

14. Entrega de documentació

La planificació temporal del projecte ha estat la que es mostra en la següent taula:

Setmana	**Tasques**
5-12/10	1,2,3,4,5
12-19/10	6
19-26/10	7
26/10-2/11	8
2-9/11	9
9-16/11	10
16-23/11	10
23-30/11	11
30/11-7/12	12
7-14/12	13
14-21/12	14

2. Intel·ligència artificial – Intel·ligència computacional

2.1 Breu introducció a la intel·ligència artificial

És globalment acceptat que l'adopció del terme *intel·ligència artificial* per a definir el camp de recerca amb el mateix nom va sorgir arrel de l'estudi anomenat *Dartmouth Summer Research Project On Artificial Intelligence* dut a terme el 1956 al Dartmouth College de Hanover. La trobada va ser proposada pels matemàtics Claude E. Shannon, Marvin L. Minsky i John McCarthy, juntament amb l'enginyer elèctric Nathaniel Rochester, i tenia com a propòsit analitzar la conjectura que:

> [...] Every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it [6, pag. 1]

Tot i que en aquesta reunió es van tractar problemes tan rellevants d'aquesta matèria com les xarxes neuronals, la capacitat d'auto-aprenentatge o la optimització de *computadors automàtics* [6, pag. 1], no va ser fins l'any 1958 en un simposi sobre la *Mechanization of Thought Processes* al Regne Unit [26, pag. 73], que es va fer patent l'auge d'aquesta àrea de coneixement com a nexe d'unió d'altres subàrees tant diverses com les matemàtiques, la neurologia, l'estadística o la lingüística.

Malgrat això, també és necessari considerar totes les influències que la intel·ligència artificial havia rebut abans de la seva consideració com a camp d'estudi multidisciplinari. Alguns exemples els podem trobar en Aristòtil, amb la introducció de la lògica sil·logística que establí els principis del raonament vàlid, Ramon Llull, amb la seva màquina de *raonament lògic Ars Magna* o Thomas Hobbes, amb la idea d'*animal artificial* en el seu llibre *Leviathan.* Però també en Leonardo Da Vinci, Blaise Pascal, Gottfried Wilhelm Leibniz, Francis Bacon, George Boole, Santiago Ramon y Cajal, Rudolf Carnap, John Von Neumann, Alfred Tarski i molts altres autors d'una gran varietat de disciplines diferents [28].

Cal valorar especialment l'aportació realitzada per Alan Mathison Turing, un matemàtic anglès considerat el pare de les *màquines de computació lògiques* (LCM – logical computing machine), que ja pensava en la intel·ligència de les computadores i en el seu futur a principis del segle XX. En el seu tractat *Computing Machinery and* Intelligence publicat per *Mind* el 1950 es podia llegir la seva visió:

> [...] I believe that in about fifty years' time it will be possible to programme computers, with a storage capacity of about 10^9, to make them play the imitation game so well that an average interrogator will not have more than 70 per cent. Change of making the right identification after five minutes of questioning. The original question, 'Can Machines think?' I believe to be too meaningless to deserve discussion [22, pag. 11]

Turing es referia al test que va idear com a prova de l'habilitat d'una màquina per a mostrar un comportament intel·ligent, i que ell va anomenar *imitation game* [1].

Va ser entre les dècades de 1950 i 1970 quan la intel·ligència artificial va progressar més ràpidament. Les investigacions en reconeixement de patrons, heurística, representació semàntica, llenguatge natural o teoria de jocs, paral·lelament al desenvolupament de màquines de computació més sofisticades, varen possibilitar la introducció de noves àrees de coneixement. Fites d'aquesta època són la creació del llenguatge de programació Lisp (1958-John McCarthy), la creació dels programes de raonament lògic SAINT (James Slagle -1963), ANALOGY (1968 – Tom Evans) i STUDENT (1967, Daniel Bobrow), la creació del projecte Dendral[2], la creació del sistema de diàleg en llenguatge natural SHRDLU[3] o la introducció del sistema *Image Understanding*[4] [28][26].

Durant els anys següents es continuà investigant en àrees conegudes i també recent creades de la intel·ligència artificial. Alguns dels avenços més significatius van estar relacionats amb:

- Les xarxes semàntiques per a la representació del coneixement i la seva utilització en el raonament dels sistemes intel·ligents.
- La introducció de les xarxes bayesianes com a models de decisió probabilístics.
- La introducció de nous sistemes per a l'aprenentatge automàtic: xarxes neuronals, arbres de decisió, coneixement no supervisat, raonament CBR[5], etc.

[1] Una computadora hauria de tenir capacitats en processament de llenguatge natural, representació del coneixement, raonament automàtic i coneixement màquina o *Machine Learning.*

[2] Per crear un algorisme que a partir d'un espectrograma de massa d'un compost químic i dels seus components atòmics en discernís la seva estructura [26, pag. 255].

[3] Terry Winograd va presentar aquest sistema com a part de la seva dissertació pel seu doctorat al Massachusset's Institute of Technology, MIT [26, pag. 238].

[4] DARPA va llançar aquest programa el 1976 [26, 338].

[5] Case-based reasoning: procediment de solucionar nous problemes basant-se en les solucions de problemes similars [87].

- La millora del processament de llenguatge natural i escenes naturals a través de la creació de noves gramàtiques, mètodes estadístics i visió computeritzada.
- La introducció *d'Intelligent System Architectures*[1].

Amb alguns d'aquests desenvolupaments es comencen a construir els primers sistemes basats en metodologies d'intel·ligència artificial a nivell comercial. La introducció dels agents intel·ligents com a entitats racionals i autònomes marca, a partir del 1995, la nova tendència d'investigació en aquesta matèria.

Actualment seria difícil especificar tots els camps en què la intel·ligència artificial està present, principalment per l'ampli ús de noves tecnologies però també perquè els progressos teòrics en els últims anys han anat de la mà de les millores en les capacitats dels sistemes reals. Les àrees d'intel·ligència artificial s'han anat tornant més integradores, desenvolupant-se juntament amb altres disciplines de coneixement. Alguns exemples d'aplicacions modernes els podríem trobar en el reconeixement de veu, els sistemes autònoms de planificació, els jocs, la mineria de dades o en els algorismes de traducció [28].

2.2 Intel·ligència computacional

Andries P. Engelbrecht [7] defineix el concepte d'intel·ligència computacional com:

> [...] the study of adaptive mechanisms to enable or facilitate intelligent behaviour in complex and changing environments [7, pag. 4]

Wlodzislaw Duch [32] en canvi, propugna que la intel·ligència computacional ha de ser entesa com:

> [...] a branch of computer science studying problems for which there are no effective computational algorithms [32, pag. 6]

D'altre banda, podem trobar més de deu publicacions periòdiques, llibres i articles sobre intel·ligència computacional [32] amb un corpuscle diferenciat d'acceptacions diverses del terme. La veritat és que avui dia encara no hi ha consens sobre si la intel·ligència computacional és una part de la intel·ligència artificial, si aquesta última ha de ésser considerada una extensió de la primera o si totes dues són acceptacions d'una mateixa àrea de coneixements.

La majoria dels estudis actuals però, mantenen una tendència a considerar la intel·ligència computacional com una subbranca de la intel·ligència artificial, centrada en l'estudi de mecanismes que dotin d'habilitats d'aprenentatge i

[1] Majoritàriament arquitectures en què una interfície comuna controla sub-programes especialitzats en tasques concretes. Preludi de les arquitectures de computadors actuals

adaptació als sistemes intel·ligents que ho requereixin. Capacitats d'adaptació a noves situacions, de descobriment, generalització, abstracció i associació de conceptes són algunes de les habilitats requerides pels diferents paradigmes existents [7]. En aquest treball d'investigació s'utilitzarà aquesta acceptació del terme com a punt de partida per a l'estudi i desenvolupament del nou algorisme ACO.

2.2.1 Paradigmes de la intel·ligència computacional

Un dels objectius a llarg termini de la intel·ligència computacional és la creació de sistemes cognitius[1] que permetin dur a terme tasques complexes (com el reconeixement de patrons, percepció de l'entorn, habilitat d'aprenentatge, memorització, generalització, etc) d'una forma similar al funcionament que demostra el sistema cognitiu humà durant la seva evolució des del període sensomotriu fins al període d'operacions formals[2]. La idea subjacent és aproximar els comportaments i capacitats dels sistemes artificials als que demostren els cervells humans, especialment pel que fa a l'especialització en l'anàlisi de patrons naturals, segmentació d'escenes visuals i auditives, control de moviment i mapatge de percepcions a accions [32].

Per a dur a terme aquestes opcions s'han desenvolupament i provat amb diversos graus d'èxit diverses aproximacions basades majoritàriament en el funcionament de sistemes biològics [1][7][9][34][35][36]. A continuació s'exposaran alguns dels paradigmes més rellevants en aquest camp.

2.2.1.1 Xarxes neuronals artificials

S'estima que en el còrtex del cervell humà hi ha entre 10 i 500 mil milions de neurones amb uns 60 milions de sinapsis que formen el nexe d'unió entre elles. Aquesta interconnexió massiva és la que habilita el pensament paral·lel i no linear [38], del que deriva la gran capacitat de *computació* del sistema neuronal humà.

Les xarxes neuronals artificials intenten simular aquest mecanisme a través de la creació d'agrupacions de neurones artificials[3] que treballen de forma similar i

[1] Sistemes dotats d'atenció, memòria-aprenentatge, llenguatge, percepció, solució de problemes, planificació, intel·ligència, etc [88].
[2] Jean Piaget en la seva teoria del desenvolupament cognitiu exposa que el sistema cognitiu humà evoluciona passant per una sèrie d'etapes marcades per canvis graduals que modelen la interacció amb l'entorn i les capacitats intel·lectuals, d'acció i percepció de l'individu [33].
[3] Artificial neuron (AN).

aproximada a com ho fan les biològiques. Cadascuna rep senyals de l'entorn o d'altres neurones, les processa i les transmet a les neurones que hi estan connectades. El processament i la força de la senyal transmesa es controlen a través d'una funció anomenada funció d'activació fAN[1], que treballa a partir dels pesos assignats a cadascuna de les entrades. En la figura següent podem veure l'esquema bàsic d'una AN:

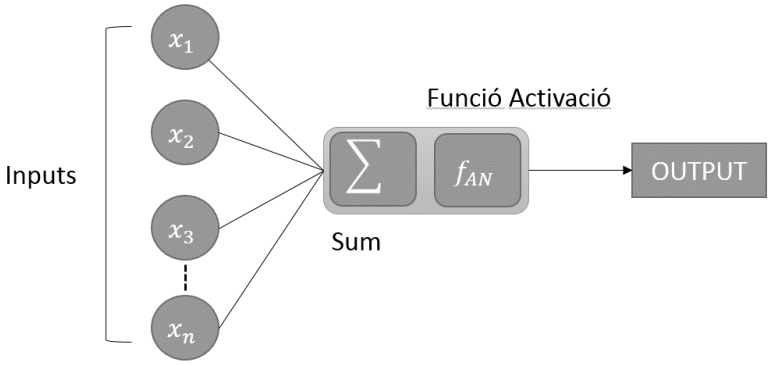


Fig. 1 – Esquema d'una neurona artificial

Aquest comportament permet als sistemes basats en xarxes neuronals reforçar certes connexions a partir dels paràmetres establerts, aconseguint un mecanisme d'aprenentatge que es pot utilitzar per predir comportaments a partir de regles i dades de mostra [7]. Si la comparació entre una sortida i les dades de mostra no és correcte, s'afinen els paràmetres per a ajustar el comportament del sistema.

Tot i que en l'actualitat existeixen moltes variacions diferents de xarxes neuronals, Engelbrecht anomena els tipus bàsics següents com a base d'evolució a models més complexos [7, pag. 40]:

- Xarxes neuronals de capa única[2].
- Xarxes neuronals multicapa predictives o anticipatives[3].
- Xarxes neuronals temporal[4].
- Xarxes neuronals auto-organitzatives[5].
- Combinació de xarxes neuronals supervisades i no supervisades[6].

[1] Són mapatges que incrementen els valors monòtonament: $\mathrm{fAN}(-\infty) = 0$ o $\mathrm{fAN}(-\infty) = -1$, i $\mathrm{fAN}(\infty) = 1$ [7, 52].
[2] Single-layer NNS (SLNNS).
[3] Multilayer Feedforward NNS (MFNNS).
[4] Temporal NNS (TNNS).
[5] Self-organizing NNS (SONNS).
[6] Combined supervised and unsupervised NNS (CSUNNS).

Models creats a partir d'aquesta aproximació s'utilitzen en camps tals com les finances, l'anàlisi de senyals, robòtica, síntesi de veu i diagnosi mèdica, entre d'altres [37].

2.2.1.2 Computació evolutiva

Tal i com descriu Kenneth A. De Jong [39, pag. 13]:

> […] there are at least two possible interpretations of the term *evolutionary system* [...] to describe a system that changes incrementally over time […] or to mean a Darwinian evolutionary system […]

Igual que De Jong, Thomas Weise [40] també afirma que tot i no haver-hi un consens general sobre què representa completament aquest últim punt, sí que existeix una acceptació general de que un sistema d'aquestes característiques podria ésser condensat en els components següents [41][42]:

- Una o més poblacions d'individus competeixen per recursos limitats.
- Les poblacions canvien dinàmicament a causa de la mort i naixement d'individus.
- El concepte d'aptitud reflecteix l'habilitat d'un individu per a sobreviure i reproduir-se.
- El concepte d'herència variacional: la descendència s'assembla als progenitors però no és idèntica.

Un sistema evolutiu es pot veure a través d'aquestes idees com un model que, a partir d'unes condicions inicials fixades, evoluciona en el temps entre diferents estats. A partir d'aquesta trajectòria és possible estudiar diversos aspectes del procés, com poden ser el seu comportament o la convergència a un punt comú, entre d'altres [39, pàg. 13]. Aquest és el punt de partida de la computació evolutiva.

Els algorismes evolutius utilitzen aquesta idea introduint el canvi substancial de ser *goal-driven*[1]. Aquests parteixen d'una població d'individus o fenotip anomenats cromosomes, cadascun amb característiques concretes o gens [7, pàg. 41]. La representació o codificació d'aquests individus (binària, decimal, etc.) depèn del problema a tractar i de les operacions que s'hagin d'aplicar. A

[1] Literalment, desenvolupament guiat per objectius.

partir d'aquest punt els algorismes segueixen l'esquema de funcionament següent [40, pàg. 96]:

1. S'avalua cada individu a través d'una funció d'avaluació que mesura la qualitat de la solució candidata aportada.
2. Es seleccionen aquells individus que proporcionen les millors solucions per al problema que s'avalua, creant un subconjunt de la població inicial.
3. Es crea una nova generació d'individus a través d'operacions de mutació i d'encreuament entre els individus. La funció d'aquestes operacions és simular els mecanismes biològics evolutius presents en els sistemes naturals.
4. Es torna a començar el procés.

El procés sol acabar després de l'execució d'un nombre determinat d'iteracions o de l'assoliment d'un criteri de bonança prèviament definit. La figura següent ens mostra el procés d'una forma gràfica:

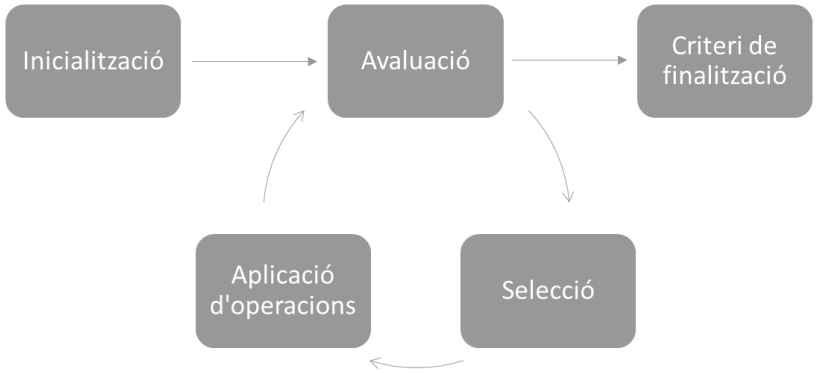


Fig. 2. Esquema de funcionament dels algorismes evolutius.

Tot i que actualment existeixen moltes classes d'algorismes evolutius, les següents són algunes de les més anomenades en la literatura actual:

- ***Genetic Algorithms [45]:*** Algorismes evolutius que imiten el model d'evolució genètica o procés de selecció natural.
- ***Genetic Programming [46]:*** Programació basada en la metodologia dels algorismes genètics on els individus són els mateixos programes o sets d'instruccions.
- ***Evolutionary Programming [47]:*** Tot i que similar a la programació genètica, la programació evolutiva no modifica l'estructura del programa o instruccions a optimitzar, sinó que es centra en l'evolució dels paràmetres.

- ***Differential evolution [44]:*** Algorismes d'optimització que també utilitzen el paradigma de la computació evolutiva. A diferència dels algorismes genètics comuns, els individus de cada població competeixen amb una sèrie de vectors de prova generats específicament.

La computació evolutiva s'utilitza actualment en una gran varietat de camps diferents entre els que podem trobar l'optimització de problemes de combinatòria, classificació, agregació o mineria de dades, entre d'altres.

2.2.1.3 Intel·ligència d'eixam

El present treball es centra precisament en aquest paradigma de la intel·ligència computacional, pel que a continuació s'introduiran els conceptes bàsics i en temes subsegüents s'exposaran d'una forma més detallada les bases, els mecanismes i el funcionament dels algorismes pertanyents a aquesta subàrea de la CI[1].

El concepte *swarm intelligence* va ésser utilitzat per primera vegada per Beni i Wang [48] en el context dels sistemes robòtics cel·lulars, on diferents entitats simples havien d'auto-organitzar-se en un espai dimensional finit per a generar patrons a partir de les interaccions amb els seus veïns més propers.

Actualment, tal i com ja afirmaven Dorigo, Bonabeau i Theraulaz el 1999, la definició també engloba:

> [...] any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies [11, pag. 20]

Els algorismes basats en eixam, inspirats en l'estudi de colònies, eixams o societats d'organismes, són capaços de produir solucions robustes a problemes complexos [12] utilitzant les interaccions i el comportament col·lectiu que mantenen aquests sistemes naturals, on la complexitat dels sistemes es genera a partir de la interacció entre individus que aparentment tenen comportaments individualitzats i auto-organitzatius [15].

Tot i que aquestes entitats que formen les agrupacions d'individus són relativament poc sofisticades i amb capacitats limitades, mostren un comportament col·lectiu intel·ligent i adaptatiu. En la majoria d'aproximacions, aquest mecanisme d'*intel·ligència emergent* s'utilitza per a resoldre problemes complexos.

[1] Computational Intelligence (CI).

Actualment existeixen moltes metodologies i variacions d'algorismes basats en intel·ligència d'eixam. A continuació s'anomenen algunes de les més rellevants [1][9][17][19][21][23][31]:

- **Ant Algorithms**: Són els tipus d'algorismes estudiats en aquesta projecte. Es basen en els patrons de comportament que mostren les colònies de formigues en entorns naturals. A través d'un mecanisme de comunicació indirecta basat en rastres de feromones per marcar el camí dels individus entre el niu i la font d'aliment, es poden trobar solucions òptimes de recorregut per a problemes que puguin ser representats en forma de graf.
- **Bee Algorithms**: Inspirat en el comportament de les colònies d'abelles, on existeixen grups d'individus amb diferents rols. Aquests algorismes s'utilitzen per a resoldre problemes d'optimització a partir dels recorreguts de les abelles en la seva cerca d'aliment.
- **Particle Swarm Optimization**: Simulen el comportament d'eixams de partícules que es mouen de forma grupal. Les partícules representen solucions candidates a un problema donat, i es mouen en l'espai de cerca d'aquest problema en funció de la influència de cada posició individual i de la posició de les altres partícules en cada instant.
- **Firefly Algorithm**: Aquest tipus d'algorismes d'optimització es basen en el comportament dels eixams de lluernes i l'atracció dels individus per les fonts de llum generades per altres membres del grup.
- **Bat Algorithm**: El transfons d'aquests algorismes d'optimització meta-heurística és la utilització del sistema d'ecolocalització que usen els microquiròpters[1] en la natura per a resoldre problemes complexos.
- **Cucko Search:** Es basen en el comportament que mostren els cucuts al deixar els seus ous en nius d'altres espècies d'ocells. S'utilitzen principalment per a resoldre problemes d'optimització.

Alguns dels camps d'aplicació d'aquestes metodologies inclouen l'aprenentatge computacional, sistemes dinàmics, optimització d'estructures o anàlisi d'imatges i dades, entre d'altres [12].

2.2.1.4 Sistemes immunes artificials

El sistema immunològic d'un organisme està format per diverses estructures biològiques i processos que serveixen per protegir aquest ens de malalties, patògens, virus i altres influències perjudicials. Per treballar de forma adequada,

[1] Subordre de ratpenats que utilitza ecolocalització [89].

el sistema ha de ser capaç de detectar correctament aquests agents i actuar per neutralitzar-los a través de la utilització de diferents mecanismes [50].

Els sistemes immunes artificials[1] intenten modelar alguns dels aspectes dels sistemes immunes naturals [7]:

- **La visió clàssica**: Existeix el mecanisme innat i no-canviant que detecta i destrueix organismes invasors, i el sistema adaptatiu que respon a cèl·lules foranies desconegudes i manté aquesta resposta en l'organisme per un període llarg de temps [49].
- **La selecció clonal**: Els anticossos generats per cèl·lules-B[2] que s'adapten de forma més correcta als antígens corresponents són estimulats per a *hipermutar* i crear nous anticossos. Això permet una resposta molt ràpida cap als antígens [49].
- **La teoria del perill**: El sistema immune ha de ser capaç de diferenciar de forma inequívoca els antígens perillosos dels no perillosos [7].
- **La teoria de la xarxa:** Les cèl·lules-B encarregades de crear els anticossos estan interconnectades de forma que s'estimulen mútuament al respondre a un antigen particular [51].

Segons Castro i Timmis [52], el marc d'implementació d'aquest tipus d'algorismes, per la seva mateixa naturalesa multi-objectiu i per la heterogeneïtat dels sistemes on s'ha demostrat la seva implementació, està definit de forma laxa, pel que sovint depèn principalment de l'enfocament del problema. Malgrat això, altres investigadors com Aickelin i Dasgupta [53], suggereixen que cal prendre decisions sobre quatre temes diferents per a implementar aquest tipus de sistemes: tipus de codificació, mesures de similitud, mecanismes de selecció i mecanismes de mutació.

Actualment els sistemes immunes artificials han demostrat la seva eficàcia en diversos camps d'actuació com poden ser la implementació de sistemes de prevenció d'intrusions, la resolució de problemes d'optimització o la resolució de problemes de criptoanàlisi.

2.2.1.5 Sistemes difusos

Els sistemes basats en lògica difusa imiten la forma en què prenen les decisions les persones, habilitant la utilització de valors aproximats que indiquen el grau de pertinença d'un element en un conjunt difús [7][54]. Els conjunts difusos són

[1] Artificial Immune System (AIS).
[2] Linfocits-B o B-Cell [90].

aquells en què els seus elements hi poden ésser continguts amb diversos graus de certesa [56].

Aquests sistemes sempre tenen tres parts diferenciades, com es mostra en la següent representació:

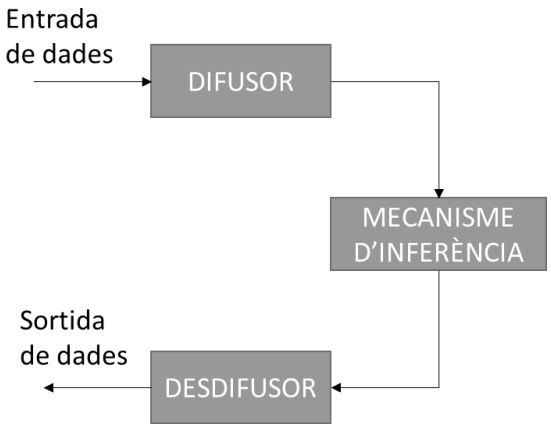


Fig. 3. Funcionament sistema difús

A cada dada d'entrada se li assigna un grau de pertinença a un conjunt a través d'una funció característica associada a aquest conjunt particular: per cada valor que pot prendre una variable x, la funció característica $\mu_{conjunt}(x)$ proporciona el grau de pertinença d'aquell valor al conjunt difús $conjunt$. Els valors d'entrada al bloc difusor sempre seran els valors concrets de la variable a analitzar i les dades de sortida seran els graus de pertinença als conjunts triats.

El sistema es defineix per una sèrie de regles d'inferència que regeixen la relació entre els conjunts difusos d'entrada i sortida. Finalment en el bloc difusor s'obtenen els resultats concrets mitjançant l'aplicació de mètodes de *des-difusió* [57].

La utilització de sistemes de lògica difusa en intel·ligència computacional permet dur a terme raonaments aproximats [7] de forma similar a com ho faria un cervell humà, a diferència de la classificació típicament binària de pertinença i no pertinença. Segons Mencar [55], la correcta interpretabilitat d'aquests sistemes s'aconsegueix a través d'una construcció precisa i acurada de les regles base del sistema.

La lògica difusa ha estat utilitzada en un gran nombre d'aplicacions com el reconeixement facial, el disseny de sistemes experts, la construcció de models de control, entre d'altres. Actualment continua formant part de molts estudis d'investigació i és un mecanisme àmpliament utilitzat en sistemes d'intel·ligència artificial.

2.3 Intel·ligència d'eixam: fonaments biològics

En el punt anterior dedicat a la intel·ligència d'eixam s'ha comentat a grans trets l'intent de modelar el comportament col·lectiu de societats d'individus o eixams amb l'objectiu d'utilitzar aquest mateix comportament en la resolució de diferents tipus de problemes. A continuació es farà un breu repàs a les bases biològiques d'aquests sistemes i a la seva utilització com a mitjà per a la resolució de problemes complexos.

El comportament col·lectiu en animals és un fenomen àmpliament observat i estudiat en sistemes biològics. Algunes agrupacions que mostren aquesta conducta són els estols d'ocells, eixams d'abelles, colònies de formigues, bancs de peixos o colònies de tèrmits, entre molts altres. Tots ells comparteixen un tret en comú: posseeixen un patró d'actuació col·lectiu que emergeix en absència d'un focus de control centralitzat. Com afirma Giardina [58], els individus actuen en funció d'informació local limitada que utilitzen per interactuar amb subjectes propers. Són aquestes accions individuals les que observades en conjunt mostren un comportament col·lectiu.

El fet que sorgeixi una estructura d'actuació a nivell global partint d'interaccions a *baix nivell* a través d'una sèrie de mecanismes dinàmics es coneix amb el nom de *self-organization*[1] [59]. En insectes socials, tal i com afirmen Dorigo, Bonabeau i Theraulaz [11], aquesta organització parteix de quatre pilars bàsics:

- **Feedback positiu o amplificació:** Es garanteix el comportament apropiat del grup a través de l'estimulació d'impulsos positius. En les formigues, per exemple, l'individu que ha trobat una font d'aliment pot advertir la resta de membres de la colònia a través d'un rastre de feromones específiques.
- **Feedback negatiu:** De forma inversa a l'anterior, es penalitzen les accions que impliquen un perjudici pel conjunt d'individus de la colònia. Aquestes penalitzacions poden venir donades per situacions de competició, exhauriment de l'aliment, mort d'individus, etc.
- **Amplificació de les fluctuacions:** Les actuacions dels individus a través d'accions aleatòries possibilita el descobriment de noves solucions que no podrien ser descobertes a través d'un comportament preestablert. Tornant al cas de les formigues, el moviment inicialment considerat aleatori[2] que segueixen pot actuar:

[1] Cal mencionar que aquest és un terme originalment desenvolupat en el context de les ciències físiques i químiques per a descriure els patrons macroscòpics observables derivats dels processos i les interaccions definides a nivell microscòpic.
[2] Estudis recents semblen indicar que les formigues no es mouen de forma aleatòria sinó seguint patrons estadístics [60].

> [...] as seeds from which structures nucleate and crow [...] foragers may get lost [..] and lost foragers can find new, unexploited food sources, and recruit nestmates to these food sources [11, pag. 23]

- **Interaccions múltiples:** El comportament col·lectiu requereix d'una interacció múltiple entre els individus, amb una densitat mínima de subjectes. Aquesta base es deriva lògicament del fet que els comportaments col·lectius s'han observat en sistemes amb molts individus que interactuen entre ells.

El requeriment d'interacció entre individus per a la generació d'un comportament comú implica dos mecanismes diferents de comunicació: directes i indirectes [61]. En el cas particular de les colònies de formigues (que són d'altra banda la base dels algorismes utilitzats en aquest projecte), la comunicació directa, com en moltes altres espècies d'insectes, es produeix a través del contacte físic o químic directe entre dos individus. D'altra banda, la comunicació indirecta[1], acostuma a ser un mecanisme més subtil que es produeix de forma asíncrona a través de l'entorn, utilitzant rastres químics de feromones.

El concepte d'*stigmergy*, tot i que no serveix per explicar completament els mecanismes de coordinació dels individus per ell mateix, proporciona una forma general d'explicar els comportaments individuals i a nivell de colònia [14]. A nivell de modelatge de sistemes no biològics, permet dotar de flexibilitat a les entitats individuals, intercanviant la coordinació síncrona a través de missatges per les interaccions indirectes a través de l'entorn. Cal tenir en compte que en sistemes en què hi participen múltiples entitats en coordinació, la comunicació sol convertir-se un dels punts crítics, al requerir sovint mecanismes complexos que afecten als costos i al rendiment del conjunt [7].

Com s'ha pogut veure, existeixen sistemes biològics en què els individus són capaços d'actuar de forma conjunta per a produir comportaments que responen als requeriments de la colònia. Tot i que l'objectiu de la intel·ligència d'eixam no és reproduir de forma exacta els patrons de conducta d'aquestes agrupacions de subjectes, si que utilitza mecanismes similars als que s'han vist fins ara per a obtenir fites preestablertes.

En aquest sentit, les aproximacions no biològiques acostumen a ser *goal-based*[2], ja que estan centrades en l'assoliment d'uns objectius concrets i la seva configuració n'és la demostració bidimensional: per una banda, l'establiment d'uns paràmetres específics és el que possibilita el correcte funcionament dels algorismes, ja siguin duració de feromones, nombre d'individus o comportament individual, entre d'altres. D'altra banda, el fet que siguin principalment aquests

[1] En anglès també es defineix aquest mecanisme com a *stigmergy.*
[2] Es focalitza el desenvolupament en els objectius a assolir.

condicionants els que controlen l'acompliment dels objectius, permet que els algorismes puguin ser redefinits de forma simple per a la resolució de diferents tipus de problemes. Cal tenir en compte com aquesta última vessant pot convertir-se en un handicap alhora d'afinar aquests *paràmetres de control*, no només per a la dificultat inherent de la qüestió sinó perquè cada paràmetre pot tenir una incidència diferent en funció de l'objectiu de l'algorisme [10].

En els últims anys hi ha hagut un auge en el desenvolupament d'algorismes inspirats en el comportament de sistemes naturals. Malgrat això, els avenços actuals no estan centrats en el desenvolupament de sistemes purs[1], sinó en la hibridació de sistemes ja coneguts per a millorar l'efectivitat i la robustesa de les solucions proposades per aquests algorismes [16][17].

3. Optimització per colònia de formigues (ACO).

3.1 Ant System: El primer algorisme

Els diferents aspectes de comportament de les colònies de formigues han inspirat des de fa anys diferents tipus d'algorismes per a resoldre problemes computacionals. Tot i que els més coneguts són els basats en *foraging* o cerca d'aliment a través de camins de feromones[2], també han estat implementats mecanismes de divisió de tasques (obreres, soldats, etc.) i transport cooperatiu, entre d'altres [62]. Tanmateix, l'únic que ha mostrat una connexió directa entre el rendiment a nivell individual i l'èxit a nivell de colònia [64] i que més àmpliament s'ha investigat i utilitzat a dia d'avui ha estat el de *foraging*. A partir d'ara es centrarà la informació en aquest mecanisme.

Marco Dorigo en la seva tesi doctoral de 1992 [65] va presentar el primer algorisme de tipus *ACO*[3]. Aquest, anomenat *AS*[4], utilitzava els mecanismes de

[1] En el sentit de la representativitat d'una sola espècie animal.
[2] Presentat com a model per primera vegada el 1989 [63].
[3] Ant Colony Optimization (ACO).
[4] Ant System (AS). Si bé realment es van desenvolupar tres tipus de variants d'aquest en funció dels paràmetres de configuració de feromones: *ant-density, ant-quantity* i *ant-cycle*, el propi Dorigo [62, pag. 70] considera l'Ant System com a sistema basat en *l'ant-cycle*. En aquest treball es prendrà la mateixa consideració.

retroalimentació positiva, computació distribuïda a través d'agents formiga i heurística àvida[1] per a descobrir ràpidament solucions acceptables.

De forma general, els agents formiga es comporten de forma similar a com ho fan les formigues reals. Considerem la figura següent (Fig. 4). En el punt a) podem apreciar com les formigues es veuen obligades a triar un dels dos camins; en b) les formigues escullen una de les dues opcions de forma aleatòria; ja que les formigues van a velocitat constant, les que fan el camí més curt tarden menys temps en arribar al destí i dipositen més feromones c). Aquestes feromones es mantenen en major mesura a pesar de la persistència limitada que tenen (ja que la persistència $p > tempsRecorregut$). Per *feedback* positiu, les formigues acabaran passant pel camí on hi hagi més feromones d).

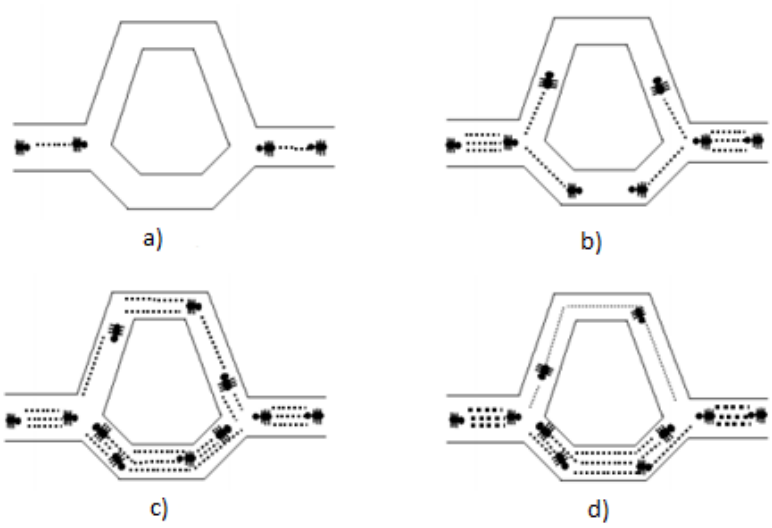


Fig. 4. Formigues: tria de camí i dipòsit de feromones.

El comportament de les formigues en l'algorisme AS funciona amb un procediment estocàstic[2] de construcció de la solució. Malgrat això, la política de construcció no és totalment arbitrària sinó que es basa en les regles o paràmetres imposats en la configuració del sistema, tals com duració de les feromones o nombre de formigues.

[1] Anomenada així ja que el sistema és capaç d'innovar de forma positiva a mesura que avança. El concepte d'avidesa representa la necessitat del sistema de trobar aliment, i per tant, d'arribar a solucions de forma ràpida.
[2] Que sorgeix a partir d'influències i efectes aleatoris [91].

3.1.1 Representació del problema

En general, un problema de minimització o optimització pot ser representat (S, f, Ω), on:

- S és el conjunt de solucions candidates
- f és una funció que assigna a cadascuna de les solucions anteriors una funció de cost $f(s,t)$
- Ω és el conjunt de restriccions que estableix el problema

L'objectiu final a l'aplicar un algorisme d'optimització és el de trobar una solució òptima global[1] $s_{opt} \in S$. El cas particular dels problemes d'optimització de combinatòria que han de ser aproximats a través de la utilització d'agents formiga poden ser representats amb els següents components [66]:

- Un conjunt finit $V = \{v_1, v_2, \dots, v_{nc}\}$ de nodes o vèrtexs.
- Un conjunt d'estats X definits en termes de seqüències $x = \{v_i, v_j, \dots, v_z\}$ sobre els elements de V. El nombre de components en les seqüències s'expressa $|x|$.
- Un nombre finit de restriccions Ω que defineix un conjunt d'estats factibles en el problema $\overline{X}$, on $\overline{X} \subseteq X$.
- Un conjunt de solucions factibles S^*, on $S^* \subseteq \overline{X}$ i $S^* \subseteq S$.
- Una funció de cost $f(s,t)$ associada a cada solució candidata $s \in S$.

Amb aquesta caracterització, cada agent formiga construeix una solució (ruta) movent-se d'un node a un altre en la direcció de l'aliment-objectiu o node final. El conjunt de nodes i arestes es representa en forma de graf $G = (V, A)$. Els vèrtexs V són el conjunt de punts de decisió pels agents, mentre que les arestes A representen les rutes que connecten aquests punts. Es consideren les distàncies entre els punts d_{ij} on $(i,j) \in A$.

3.1.2 Funcionament de l'algorisme AS

L'algorisme executa un màxim establert prèviament de t_{max} iteracions, i en cadascuna les m formigues realitzen n passos per a construir cada recorregut complet. En cada pas s'apliquen les regles de decisió probabilístiques basades

[1] Una solució amb un cost mínim que, a més, satisfaci les restriccions del problema.

en els dipòsits de feromones presents en els camins. Usualment cada aresta (i, j) té associat un camí de feromones $\tau_{ij}(t)$. El dipòsit de feromones en cada arc és proporcional a la qualitat de la solució que proporciona en el recorregut; com més curt és aquest, més quantitat de feromones hi dipositarà l'agent. Aquest canvi possibilita l'adquisició d'una experiència general de conjunt que permet als agents construir millors solucions en cada iteració posterior.

Dorigo i Stützle [66], enumeren les propietat que ha de tenir cada agent formiga de la forma següent:

- Ha d'explotar el graf $G = (V, A)$ per buscar solucions s amb cost mínim.
- Ha de tenir una memòria M per emmagatzemar informació del recorregut que ha dut a terme fins al moment. Aquesta memòria serà la que s'utilitzarà per a construir solucions factibles, per avaluar les solucions trobades i per desfer el camí seguit per a dipositar les feromones[1].
- Ha de tenir assignat un estat d'inici x_s^m i una o més condicions d'acabament e^m.
- Si es mou d'un estat $x_r = (x_r - 1, i)$ a un estat $x_r = (x_r, j)$ però no pot utilitzar un camí òptim, l'agent es podrà moure cap a aquest altre node j generant un estat no factible o no desitjable.
- Es mou aplicant una regla de decisió probabilística. La regla és una funció del contingut de feromones disponibles, de la memòria privada de la formiga que manté la seva història passada i de les restriccions del problema.
- El procediment de construcció del recorregut de cada agent m acaba quan una de les condicions d'acabament e^m es satisfà.
- Si s'afegeix un nou component a la solució s, es poden actualitzar els dipòsits de feromones associats a les connexions afectades per aquest component. Aquest mecanisme s'anomena *online step-by-step pheromone update.*
- Quan es construeix una solució, l'agent pot refer el camí de tornada i actualitzar els dipòsits de feromones en els arcs corresponents. Aquest procediment s'anomena *online delayed pheromone update.*

Cada node del graf que representa un problema està associat a una taula de decisió $D_i = [d_j^i(t)]_{|V_i|}$ que s'obté a partir dels dipòsits de feromones i dels valors heurístics locals de la forma següent:

[1] Els algorismes de formiga funcionen realitzant els dipòsits de feromones de forma retardada, per poder avaluar cada solució.

$$d_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\mu_{ij}]^{\beta}}{\sum_{l \in V_i}[\tau_{ij}(t)]^{\alpha}[\mu_{ij}]^{\beta}} \; \forall j \in V_i$$

On:

- τ_{ij} és la quantitat de feromones en l'arc (i,j) en l'instant t
- $\mu_{ij} = 1/d_{ij}$ és el valor heurístic de moure's del node i al node j
- V_i és el conjunt de vèrtexs veïns del node i
- $\propto$ i β són paràmetres que controlen el pes relatiu de les feromones i el valor heurístic envers el conjunt

La probabilitat que un agent formiga m esculli d'anar d'un node i a un node $j \in V_i^k$ mentre construeix el seu recorregut en la iteració i-èssima de l'algorisme és:

$$p_{ij}^{k}(t) = \frac{d_{ij}(t)}{\sum_{l \in V_i^k} d_{il}(t)}$$

On $V_i^k \subseteq V_i$ representa el conjunt de nodes adjacents al node *i* que encara no han estat visitats per l'agent formiga m.

Els paràmetres de control $\propto$ i β funcionen de la següent forma: si $\propto = 0$, els nodes més propers a l'actual tindran més probabilitat de ser seleccionats. D'altra banda, si $\beta = 0$, només es tindrà en compte l'amplificació de probabilitat produïda pels dipòsits de feromones. En la majoria d'aproximacions es fa patent la necessitat d'establir un compromís entre els valors d'aquestes variables per a obtenir resultats òptims [1][3][5][8][65].

Quan tots els agents formiga han completat els seus recorreguts es duu a terme el càlcul de l'evaporació i actualització dels dipòsits de feromones $\Delta\tau_{ij}(t)$ en cada arc que ha estat utilitzat:

$$\Delta\tau_{ij}(t) = \begin{cases} \dfrac{1}{L^m(t)} & si\ (i,j) \in T^m(t) \\ 0 & si\ (i,j) \notin T^m(t) \end{cases}$$

On:

- $T^m(t)$ és el recorregut generat per l'agent formiga m en la iteració t
- $L^m(t)$ és la longitud del recorregut anterior

És necessari tenir en consideració la direccionalitat dels arcs. En aquest cas particular es considera un problema simètric en què els arcs tipus (i,j) són considerats iguals als arcs (j,i). En problemes asimètrics els arcs anteriors són diferents i, per tant, també ho serà l'actualització dels seus dipòsits de feromones.

Finalment, l'addició de noves feromones i l'evaporació de les feromones es duu a terme a través de la següent regla aplicada a cadascun dels arcs:

$$\tau_{ij}(t) \leftarrow (1-p)\tau_{ij}(t) + \Delta\tau_{ij}(t)$$

On:

- $\Delta\tau_{ij}(t) = \Sigma_{m=1}^{k}\Delta\tau_{ij(t)}^{m}$ i k és una constant que representa el nombre de formigues a cada iteració

Dorigo [65] estableix que els paràmetres correctes de l'algorisme s'han de configurar inicialment de següent forma:

- Establir els dipòsits inicials de feromones $\tau_{ij}(0)$ a un valor enter constant i petit τ_0 en tots els arcs.
- Mantenir el nombre de formigues igual en cada iteració $m = k$.
- Establir els valors dels paràmetres $\propto$ i β en 1 i 5 respectivament.

Cal veure que tots els agents formiga actuen de forma paral·lela i independent per arribar a trobar múltiples solucions diferents. Usualment, la majoria d'aproximacions trobades per cada agent de forma individual no són prou bones per a ésser considerades solucions del problema, però a través de la seva interacció col·lectiva s'obtenen resultats amb la qualitat cercada.

La representació que s'acaba de definir per l'algorisme Ant System es va demostrar eficient en la resolució de problemes TSP[1] amb un nombre reduït de nodes per visitar (entre 30-75), en comparació amb altres aproximacions d'intel·ligència computacional tals com els algorismes genètics [67]. Malgrat això, en problemes amb un nombre de nodes creixent, l'algorisme requereix un nombre d'iteracions molt elevat per a mostrar una convergència a una possible solució. Per aquest motiu s'han anat desenvolupant algorismes més complexos amb aproximacions diferents. En el següent punt s'enumeren alguns dels més destacats.

3.2 Altres algorismes ACO

La majoria de variants simples de l'algorisme AS han estat desenvolupades a través de la modificació dels mecanismes d'actualització dels dipòsits de feromones, tal i com afirmen Dorigo i Blum [68, pag. 8]:

[1] Traveling Salesman Problem (TSP). El problema de l'agent viatjant es basa en trobar el camí mínim que hauria de realitzar un viatjant si ha de visitar tots els punts d'una ruta exactament una vegada i tornar al punt d'origen. Es té en compte que cada recorregut o arc entre nodes té un pes o valor associat [92].

> Variants of the ACO algorithm generally differ from each other in the pheromone update rule that is applied. A well-known example of an instantiation of update rule is the AS-update rule, that is, the update rule of Ant System […] An example of rule that is more used in practice is the IB-update rule (where IB stands for *iteration-best*) […] An stronger bias is introduced by the BS-update rule, where BS refers to the use of the *best-so-far* solution […] In practice, ACO algorithms that use variations of the IB-update or the BS-update rule and that additionally include mechanisms to avoid premature convergence achieve better results than algorithms that use the AS-update rule […]

Alguns dels exemples més clars els podem trobar en els algorismes següents:

- **Elitist Ant System (EAS)**: Introdueix el concepte d'estratègia elitista, permetent emfatitzar el *valor* del millor recorregut trobat al final de cada iteració. Només es té en compte aquest recorregut per a dur a terme l'actualització de feromones.
- **Ant System Rank-Based (ASrank)**: Es segueix una estratègia elitista com en el cas anterior, però no només es té en compte el millor recorregut sinó un conjunt d'aquests generat pels agents amb millor posició en un rànquing. Al final de cada iteració les formigues s'ordenen en funció de la longitud o valor de la seva solució. Les que han obtingut més bons resultats són valorades a l'alça en el rànquing general. Finalment només un nombre ω de formigues amb més rang és escollit per a dur a terme l'actualització global de feromones [70].
- **Max-Min Ant System (MMAS):** És un algorisme que centra el seu mecanisme d'actualització de feromones en només permetre que un dels agents formiga n'actualitzi els dipòsits en cada iteració. Els valors d'aquest dipòsits estan restringits a un interval concret $[\tau_{min}, \tau_{max}]$ i són inicialitzats a l'inici de l'algorisme al valor màxim permès, τ_{max} [2]. És l'algorisme que s'utilitza en el present treball per a realitzar la comparativa de resultats amb l'algorisme desenvolupat per a assolir solucions desitjables per al problema ACSP.

D'altra banda, també hi ha hagut avenços en altres aspectes dels algorismes, i aproximacions totalment diferents, com es mostra a continuació de forma resumida:

- **Ant Colony System (ACS)**: Informalment, l'algorisme ACS funciona distribuint les formigues en nodes d'acord amb una regla d'inicialització. Cada formiga construeix un camí cap al node objectiu aplicant una regla de transició entre estats, i actualitza la quantitat de feromones en les arestes visitades aplicant una regla local d'actualització. Quan tots els agents han acabat els respectius recorreguts, els dipòsits de feromones

són actualitzats a través d'una regla global. Dorigo i Giambardella [69] van introduir tres grans canvis en el disseny original per millorar el seu rendiment:

- La introducció d'una regla de transició entre estats per a proporcionar un mecanisme de balanceig entre l'exploració de nous nodes i la utilització del coneixement previ sobre el problema acumulat. La regla funciona afavorint la transició entre nodes propers i amb més quantitat de feromones.
- L'aplicació de la regla d'actualització global només en nodes que pertanyin al millor recorregut d'un agent. De forma similar a l'estratègia elitista, es pretén focalitzar la cerca en nodes veïns dels nodes pertanyents al millor recorregut dut a terme per un agent. Com es pot observar, però, això no implica en cap cas que el millor recorregut sigui l'òptim, ni tampoc desitjable.
- L'aplicació d'una nova regla d'actualització de feromones *local* de forma paral·lela a la construcció dels recorreguts per part dels agents formiga. El seu rol és dinamitzar l'elecció de recorreguts de forma individual entre els agents, fent els nodes més visitats lleugerament menys desitjables. D'aquesta manera, les formigues poden utilitzar la informació històrica de feromones de forma més eficient [69, pag. 7]

- **Hyper-cube framework per a algorismes ACO (HCF):** Tal i com afirma Dorigo [68, pag. 10]:

> Rather than being an ACO variant, the hyper-cube framework (HCF) for ACO is a framework for implementing ACO algorithms that comes with several benefits.

Aquests beneficis dels que parla estan basats principalment en la consideració de cadascuna de les solucions $s \in S$ com a un vector amb nombres binaris $\{0,1\}$ de longitud n dependent dels nodes visitats. Cada posició d'aquest vector indica si el node al que representa pertany o no al conjunt de nodes solució del problema.

Vist en una representació n dimensional com la que es mostra a la figura següent, es crea una àrea formada pels vèrtexs solució del problema (en què cada vèrtex és un conjunt de nodes inicials). Al mateix temps, el vector de valors de feromones $\vec{\tau} = (\tau_1, \dots, \tau_n)$ es converteix en una distribució de probabilitats sobre aquesta mateixa àrea de solucions, amb cadascun dels valors associat a una solució concreta. Quan es realitza l'actualització global de feromones, la distribució es mou en la direcció d'una nova distribució de probabilitat generada amb les noves solucions.

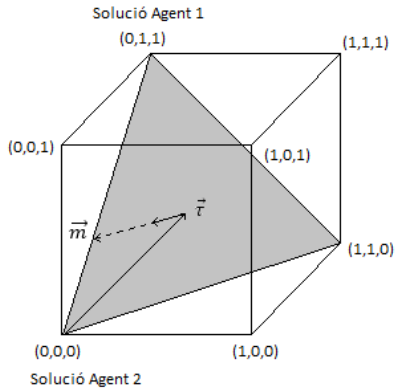


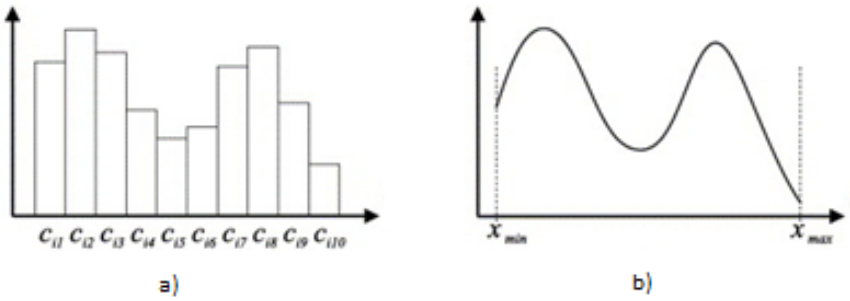
Fig. 5. Representació de l'espai 3-dimensional en una aproximació HCF.

Aquest mecanisme permet no només afavorir una observació més matemàtica del funcionament dels algorismes ACO [71, pag. 4], sinó també evitar casos d'estancament[1] a través d'un sistema d'actualització de feromones més eficaç, que permet re-avaluar elements-solució trobats anteriorment i guiar la cerca cap a àrees no visitades. $\vec{m}$ $\vec{\tau}$

- **Extended ACO for continuous domains (ACO_R):** Els algorismes que s'han vist fins ara estaven dissenyats per a problemes d'optimització en combinatòria amb dominis no continus. Malgrat això, molts problemes que es poden trobar en situacions reals requereixen un tractament diferent, ja que les solucions algorísmiques vistes fins ara només eren capaces d'aproximar solucions a partir de la transformació de rangs d'aquests dominis en sets finits [72, pag. 1]. Per aquest motiu, Socha i Dorigo [72] van proposar el 2006 una aproximació per abordar aquests problemes modificant el comportament dels algorismes ACO.

  La idea principal és que l'espai de cerca del model d'un problema d'optimització S també està definit sobre un conjunt finit de variables de decisió continues (com passava amb les aproximacions ACO normals per a dominis continus), tot i això, en aquest cas els agents utilitzen una funció de densitat de probabilitat, en comptes d'una distribució discreta de probabilitat. En la següent figura es poden apreciar les diferències:

[1] *Stagnation*: En el context dels algorismes formiga, aquest estancament significa que els agents tendeixen a convergir en una sola solució, no essent capaços de sortir d'aquest punt per a replantejar la cerca en un altre sentit.



a) Distribució discreta de probabilitat (l'eix X mostra el nombre de components que pertanyen a l'espai de solucions).

b) Funció contínua de densitat de probabilitat (l'eix X mostra el possible rang min-max establert per un espai de solucions).

Fig. 6. Diferències entre aproximacions ACO per a dominis continus.

Per a definir aquesta funció de densitat de probabilitat, Socha i Dorigo utilitzen una funció de *kernel Gaussià*, G_i, com a suma ponderada de diverses funcions Gaussianes d'una sola dimensió $g_l^i(x)$ (5, pag. 13):

$$G^i(x) = \sum_{l=1}^{k} \omega_l g_l^i = \sum_{l=1}^{k} \omega_l = \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(x-\mu_l^i)^2}{2\sigma_l^{i2}}}$$

On:

- k és el nombre de funcions Gaussianes
- L'equació està parametritzada amb els vectors $\vec{\omega}, \vec{\mu}$, i $\vec{\sigma}$. El primer vector representa els pesos, mentre que el segon és el vector de les mitjanes i l'últim el de les desviacions estàndard. La mida d'aquests vectors és igual al nombre de funcions Gaussianes que formen el *kernel Gaussià*.

Més informació sobre el funcionament complet d'aquest algorisme pot ésser consultada en l'article original de Socha i Dorigo [72].

- **Continuous Ortogonal Ant Colony (COAC):** La metodologia de disseny ortogonal presentada per Hu, Zhang i Li [3] en aquest algorisme, també està enfocada en la resolució de problemes amb dominis continus, amb la peculiaritat que la seva aproximació utilitza projeccions ortogonals com a mecanisme de cerca dels agents. La idea en aquest cas és que el nombre d'arestes en el graf que representa aquests problemes ja no és fixe i limitat, sinó que està establert en un domini $n - dimensional$.

  Amb aquesta aproximació les formigues són capaces de detectar les feromones en regions localitzades amb uns radis definits inicialment, sense utilitzar les regles locals vistes anteriorment. Cadascuna d'aquestes regions té diverses propietats: el radi de cerca, les coordenades al centre,

la quantitat de feromones i un rang de *desitjabilitat* que s'avalua amb una funció objectiu.

El funcionament general és el següent:

- En cada iteració existeixen μ regions en el domini que es generen de forma aleatòria. Els agents trien les regions que exploraran en funció de la quantitat de feromones en aquestes. Existeix un paràmetre q_0 que es pot establir per a determinar com actuaran les formigues en funció d'aquesta quantitat.
- Quan un agent selecciona una regió, aplica el mètode de disseny ortogonal per a seleccionar alguns punts en aquesta regió per explorar, com es pot apreciar en la següent figura. El punt seleccionat es converteix en el nou centre de la regió i s'ajusta un nou radi per la regió.

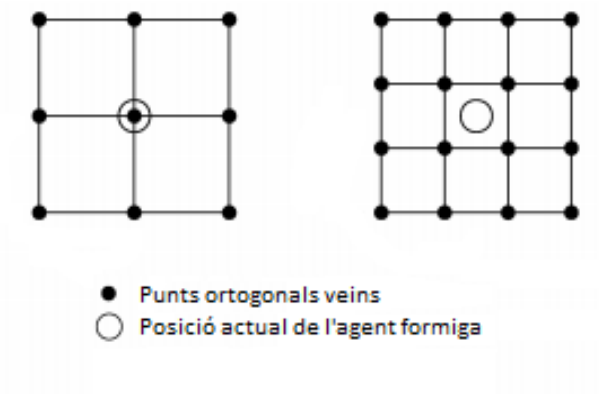


Fig. 7. Mètode de cerca ortogonal.

- Quan tots els agents han acabat l'exploració ortogonal s'avalua la *desitjabilitat* de cada regió a partir d'una aproximació elitista en l'elecció de les solucions dels agents. Només es selecciona un nombre establert de regions per a la següent iteració. Aquest procés s'anomena modulació global.

A continuació es pot veure un diagrama de flux del funcionament de l'algorisme:

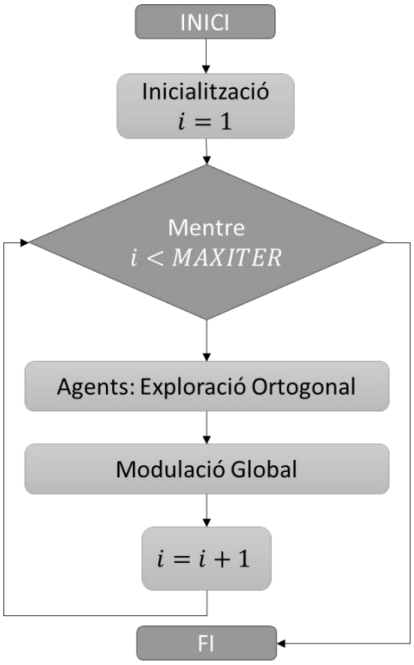


Fig. 8. Diagrama de flux algorisme COAC.

- **Multiple Objective ACO algorithms (MOACO):** Per a poder parlar correctament dels algorismes de tipus MOACO, primer cal descriure breument els problemes d'optimització multi-criteri. Aquests estan caracteritzats pel fet que més d'un objectiu ha de ser optimitzat de forma simultània, incrementant la dificultat per a trobar una solució viable. Donada aquesta complexitat intrínseca, i el fet que usualment no existeix una sola solució *bona*, els procediments meta-heurístics[1] com els algorismes de colònia de formigues, s'han mostrat especialment eficaços com a mecanisme per a afrontar aquesta problemàtica.

  El cas particular dels algorismes MOACO es centra en l'extensió d'alguns algorismes ACO modificats per a la resolució de problemes MOO[2] [74]. Tot i que existeixen moltes propostes diferents en la literatura dels algorismes ACO i que aquestes presenten característiques significativament diferents, es pot realitzar una classificació basada en la investigació de Bezerra, Goldbarg, Goldbarg, M. i Buriol [73] sobre tres temes generals:

  - **La construcció de la solució**: Donada l'heterogeneïtat dels objectius, es pot utilitzar una heurística centrada en aquests objectius i/o en les estructures de feromones. Això sovint implica realitzar un procés d'agregació sobre les diferents fonts d'informació a través de diversos mecanismes estadístics. Les

[1] Aquells que no tracten de buscar una solució directament sinó que la solució s'extreu del conjunt de solucions parcials obtingudes [29].
[2] Multiple Objective Optimization (MOO).

solucions trobades d'aquesta forma depenen de les operacions d'agregació escollides.

- **El nombre de colònies**: Es sol utilitzar més d'una colònia de formigues que s'especialitza en una de les parts del problema. Aquesta aproximació sol ser complexa, ja que es requereix com a mínim una estructura de feromones per colònia, s'han d'establir vectors definits només per a cada secció del problema i cal actualitzar les feromones de forma no compartida entre colònies (una colònia no pot intervenir en l'actualització d'una altra).
- **El criteri de selecció per a l'actualització de feromones**: En aquest cas existeixen dos camps principals d'actuació. Els primers es centren en actualitzar els dipòsits de feromones en funció de les millors solucions per objectiu (els objectius són independents entre si). Els segons només actualitzen els dipòsits de feromones amb les solucions que *no dominen* les solucions dels altres objectius (els objectius són dependents entre si). En qualsevol cas, la selecció final de les solucions que s'utilitzaran per actualitzar els dipòsits de feromones es fa seguint dos criteris diferents: (i) les solucions es comparen amb un *pool*[1] de solucions local o global de la colònia. (ii) Les solucions es comparen de forma històrica amb els resultats de la iteració anterior o amb els resultats històrics globals.

Tot i que l'objectiu d'aquest projecte no és detallar els tipus concrets d'algorismes MOACO que apareixen en la literatura d'aquest tema, a continuació es mostraran alguns dels que García-Martínez, Cordón i Herrera [75] consideren significatius en el seu estudi sobre la taxonomia d'aquests algorismes:

- **Multiple Objective Ant-Q Algorithm (MOAQ)**: Basat en una variant d'algorisme ACO anomenada ACO-Q que utilitza el reforç positiu.
- **Bi-criteria Ant Algorithm (BicriterionAnt):** Enfocat a la resolució de problemes amb dos criteris objectiu. Es caracteritza per utilitzar un recorregut de feromones per a cada objectiu.
- **Pareto ACO (P-ACO):** Utilitza el concepte d'*optimalitat* o eficiència de Pareto [76]. L'actualització de feromones es duu a terme amb els dos millors resultats per a cada objectiu.

[1] Entès, en aquest cas, com a dipòsit o *magatzem* d'emmagatzematge.

3.3 State of the art

Durant l'última dècada, els algorismes d'optimització per colònia de formigues han rebut una atenció considerable, no només per l'aproximació meta-heurística que utilitzen, sinó per què s'han demostrat altament eficaços per tractar problemes complexos de forma eficient. Fent una revisió de la literatura existent actualment, es poden entreveure dues tendències principals pel que fa a les investigacions en aquest camp. D'una banda es mostra una inclinació marcada a la utilització d'hibridació amb altres algorismes d'intel·ligència d'eixam, tècniques estadístiques i aproximacions d'intel·ligència computacional. Exemples recents es poden trobar en [4][10][25][31][77][78][79][80]. De l'altre, es pot comprovar com els algorismes ACO s'han popularitzat en moltes àrees d'estudi diferents, per una gran varietat de problemes d'optimització [81][82][83][84][85][86].

Si bé és cert que la investigació en aquest tipus d'algorismes ha avançat molt des de la seva concepció, actualment segueixen quedant algunes qüestions pendents de resolució. El principal escull continua essent la creació de nous algorismes derivats per a estendre la seva utilització a problemes d'optimització més complexos [5, pàg. 11], tot i que també, i tal i com afirmen Afshar, Massoumi i Mariño [5]:

> Despite the rapid development and application of ACO algorithms, their mathematical analysis remains partly unsolved […] Although, various studies have attempted to carry out convergence analysis, it remains an active and challenging topic.

Per tant, l'estat actual dels algorismes de colònia de formigues es troba en un punt intermedi entre la solidesa de la recerca duta a terme fins al moment i la investigació en noves metodologies que puguin expandir la utilització dels algorismes ACO a nous àmbits de treball i de coneixement. L'aspecte matemàtic, inherentment difícil a causa de les interaccions múltiples que es produeixen en els algorismes meta-heurístics i que causen fenòmens de naturalesa estocàstica, suposa també una base necessària que cal estudiar pel que fa a diversos aspectes. Alguns que assenyala Yang [9, pàg 13] com a destacats són:

- L'anàlisi específic del procés que segueixen aquests mecanismes per a arribar a obtenir les solucions d'un problema donat o procés de convergència: Es requereix una interfície unificada per a l'anàlisi dels algorismes.
- Les mesures de rendiment: No hi ha un criteri general de quins són els millors mecanismes per a dur a terme estudis comparables.
- Exploració i explotació: Cal definir un marc que permeti establir quin ha de ser el balanç òptim entre les accions d'exploració-diversificació i explotació-intensificació que duen a terme els agents.

D'aquesta forma, tot i que els algorismes de colònia de formigues continuen estan en el centre de les investigacions d'intel·ligència d'eixam i molts estudis concentren els esforços en millorar-ne la comprensió i l'aplicabilitat, encara queden diverses qüestions sense resposta que deixen oberta la porta a la recerca. Els algorismes realment intel·ligents (en referència al funcionament de la intel·ligència animal) encara han de ser desenvolupats.

4. El problema d'optimització ACSP: Airline Crew Scheduling Problem

4.1 Introducció

Dades recents de la IATA[1] [103] estimen que la indústria aèria va generar de forma directa i global uns 58.1 milions de llocs de treball i aproximadament 2.4 bilions de dòlars (US $)[2] en beneficis només durant l'any 2014; fet que representa un 3.4% del producte interior brut de tota l'economia mundial.

Tot i que el combustible suposa el cost més elevat per a les línies aèries, és un cost que pot ésser controlat només de forma indirecta, ja que depèn principalment del preu actual de mercat. D'altra banda, els costos derivats del personal representen la segona despesa més important per aquest tipus d'empreses i depenen de diferents factors que si poden ésser avaluats de forma individual o conjunta per a reduir el seu impacte en els beneficis finals de les companyies.

Així doncs, en un context actual en què les línies aèries ofereixen serveis *low-cost,* minimitzar aquests últims costos és una tasca essencial. Petites reduccions poden significar estalvis de l'ordre de milions. Com a resultat, el problema d'optimitzar els processos de distribució de tripulacions i agrupament-emparellament de vols, per la seva complexitat i importància, és una qüestió que encara rep una atenció constant tant en la indústria com en l'àmbit acadèmic [104]. En el present treball, s'utilitzarà aquest problema per a comparar el rendiment de l'algorisme MMAS amb una implementació d'ACO desenvolupada especialment per a aquesta problemàtica.

[1] Associació Internacional del Transport Aeri (IATA).
[2] Beneficis en *escala llarga* de parla no anglesa ($2.4 trilions).

4.2 Definició del problema ACSP

Tradicionalment i a causa de la seva complexitat, el problema de la distribució de tripulacions es divideix en dos sub-problemes diferents [93][98][99][105]:

- ***Crew Pairing-Scheduling Problem****:* És considera el problema principal i representa el fet de construir les rutes dels avions agrupant-les en emparellaments[1] o seqüències de vols que han de començar i acabar en una mateixa base d'operacions de l'aero-línia. Aquests *pairings* han de minimitzar els costos derivats de les tripulacions: estada en hotels, temps morts entre vols i temps de descans. A més, també han de satisfer les restriccions dels horaris de vol, les rutes de la companyia aèria, les regulacions de les aero-línies i la legislació vigent.
- ***Crew Rostering-Assignment Problem****:* Els emparellaments realitzats en la fase anterior són seqüenciats conjuntament amb altres activitats que han de dur a terme els membres de les tripulacions de forma individual (tasques als aeroports, tasques de reserva, vacances i altres requeriments) i són assignats a cadascun dels membres.

Tot i que ambdós sub-problemes depenen de regles i restriccions complexes, és el *crew-pairing* problem el que ha estat més estudiat. La raó és que s'ha demostrat que la major reducció de costos pot aconseguir-se obtenint únicament *pairings* productius que minimitzin aquests costos [97][99][101][106]. La comparativa dels algorismes d'aquest treball es realitzarà utilitzant el *crew-pairing problem*, seguint un plantejament similar al de Deng i Lin [1], que reformula el problema ACSP com un problema tipus TSP[2], com es veurà en el punt següent.

Per a comparar, avaluar i escollir els millors *pairings* entre tots els que es poden generar que compleixin les restriccions establertes, s'utilitza una funció de cost que cal minimitzar. Aquesta queda definida a través dels costos de tripulació:

- Pagament mínim per cada servei[3] en cada *pairing*.
- Despeses d'allotjament entre serveis.
- Despeses derivades de temps no útil entre serveis de vol[4] en cada *pairing*.

Així, per un problema amb n vols, m serveis de vol i o *pairings*, la funció es podria formular en base a [1, pàg. 2]:

[1] Anomenats Pairings en el context del problema.
[2] Traveling Salesman Problem, veure pàgina 25.
[3] Es considera servei el treball que duu a terme un tripulant des de que embarca en un vol fins que desembarca en l'últim vol d'un *pairing*.
[4] Es considera servei de vol el treball que duu a terme un tripulant des de que embarca en un vol fins que embarca en el punt de destí d'aquest mateix vol.

$$Min \sum_{p=1}^{m} C_{t_{basic}} + C_{t\_extra}\left[\left(\sum_{i=1}^{n} z_{ip} T_{temps\_vol\_i}\right) - T_{t\_minim\_vol}\right] + \sum_{p=1}^{m} \left(C_{t\_hotel} + C_{t\_desc\_extra} T_{desc\_entre\_vols} + C_{perdua}\left(T_{desc\_entre\_vols} - T_{desc\_minim}\right)\right)$$

On:

- C_{t_basic} és el pagament bàsic per un servei.
- C_{t_extra} és el bonus pagat a la tripulació per cada minut extra de vol no contemplat.
- $T_{temps\_vol\_i}$ representa el temps del vol i. La variable z_{ip} s'utilitza per saber si un vol i està en un servei de vol dins un *pairing* concret ($z_{ip} = 1$) o no ($z_{ip} = 0$).
- $T_{t\_minim\_vol}$ és el temps mínim pagat per un servei de vol.
- C_{t_hotel} és el cost de l'allotjament.
- $C_{t\_desc\_extra}$ és el pagament per cada minut extra de descans.
- $T_{desc\_entre\_vols}$ indica el temps de descans entre dos vols.
- T_{desc_minim} és el temps de descans mínim establert entre vols.
- C_{perdua} és el cost que ocasiona a la companyia cada minut de temps de la tripulació perdut.

Com s'ha comentat anteriorment, el problema també contempla una sèrie de restriccions dependents de les regulacions estatals i de companyia en cada línia aèria. En aquest cas s'utilitzarà una aproximació similar a l'establerta per Deng i Lin en el seu estudi [1]:

1. El vol següent a un altre vol haurà de sortir de la mateixa ciutat que l'anterior, per cada *pairing*.
2. En un servei hi haurà un màxim de $N_{\max\_vol}$ vols.
3. Existirà un temps màxim de descans entre vols $T_{\max\_desc}$. Si el temps d'espera entre vols supera aquest temps, la tripulació haurà d'anar a descansar a un hotel.

4. El temps total de treball en un servei haurà de ser inferior al temps total permès per servei: $T_{servei} < T_{\max\_servei}$. El temps d'un servei es calcularia idealment com:

$$T_{servei} = \sum_{i=1}^{n} T_{temps\_vol\_i} + T_{ij}^{descans}$$

5. El temps total de vol en un servei haurà de ser menor o igual que el màxim temps de vol permès per servei $T_{\max\_vol\_servei}$. Aquesta restricció es pot expressar com:

$$\sum_{i=1}^{n} T_{temps\_vol\_i} \leq T_{max\_vol\_servei}$$

6. Restriccions de descans:
   i. El temps de descans entre dos vols consecutius haurà de ser igual o major que el temps mínim de descans entre vols establert:

$$T_{desc\_minim}$$

   ii. La tripulació que treballi durant un període de 12 o més hores en un servei haurà de descansar un període no inferior a 24 hores consecutives:

$$T_{desc\_m} \geq 24 \ \ si \ \ T_{servei} \geq 12$$

4.2.1 Problema en base TSP

Per a aconseguir aquesta aproximació es considera el problema com un graf, on els nodes són vols i les arestes que els connecten representen la *satisfactibilitat* de les restriccions entre dos vols consecutius [109]. L'objectiu és aconseguir trobar agrupacions de camins (*pairings*) que satisfacin les restriccions del problema i que mantinguin valors mínims per a la funció de cost definida en l'apartat anterior.

Número de vol	*Ciutat sortida*	*Ciutat arribada*	*Hora sortida*	*Hora arribada*
0	A	B	08:00	09:00
1	A	D	09:30	10:10
2	D	B	10:35	11:35
3	B	C	09:20	10:20
4	C	B	10:40	11:40
5	B	A	12:00	13:20
6	A	B	13:40	15:00
7	B	A	15:20	15:50
8	B	A	15:30	16:30

Taula 1. Exemple de distribució de vols en una companyia aèria.

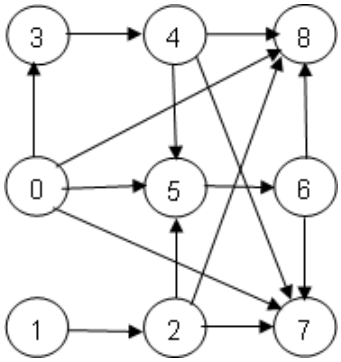


Fig. 9. Graf dirigit pel problema de la taula 1 amb una restricció de temps de descans de 20 min.

En la taula 1 i la Fig. 9 podem veure l'exemple que presenten Ogdemir i Mohan [109] per a mostrar la representació d'un problema ACSP en base TSP en funció d'una restricció en el temps de descans de 20 minuts. En la taula 2 es poden veure dues de les solucions possibles d'aquest problema, agrupades en *pairings* de vols que contenen totes les rutes que cal cobrir. Com es pot observar, qualsevol conjunt solució només és vàlid si inclou tots els vols possibles.

Solució	***Pairing***	***Vol***	***Ciutat sortida***	***Ciutat arribada***	***Hora sortida***	***Hora arribada***
1	1	1	A	D	9:30	10:10
		2	D	B	10:35	11:35
		5	B	A	12:00	13:20
	2	6	A	B	13:40	15:00
		8	B	A	15:30	16:30
	3	0	A	B	8:00	9:00
		3	B	C	9:20	10:20
		4	C	B	10:40	11:40
		7	B	A	15:20	15:50
2	1	1	A	D	9:30	10:10
		2	D	B	10:35	11:35
		5	B	A	12:00	13:20
	4	6	A	B	13:40	15:00
		7	B	A	15:20	15:50
	5	0	A	B	8:00	9:00
		3	B	C	9:20	10:20
		4	C	B	10:40	11:40
		8	B	A	15:30	16:30

Taula 2. Algunes solucions possibles per al problema definit pel graf de la fig. 9.

Tot i que en aquest cas concret d'exemple no s'han considerat totes les restriccions que s'imposaran en el problema que tractaran els algorismes ACO d'aquest treball, en la fig. 10 s'especifica el cicle de treball genèric que durà a

terme cada agent per a trobar i seleccionar cada solució. Cal tenir en compte que en aquest apartat encara no es consideren els dipòsits de feromones i les funcionalitats pròpies de cada tipus d'algorisme de colònia de formigues, sinó que es defineix un flux per a la construcció de solucions individuals.

Inici
// n: conjunt de vols pendents. Inicialment igual al conjunt total de vols m
// r: conjunt de vols que compleixen les restriccions
crea nou *pairing*
selecciona vol de m
afegeix vol al *pairing*
actualitza n
Mentre $(n! = \emptyset)$ **fes**
    r = selecciona vols que compleixin restriccions en funció del vol actual
    si $(r = \emptyset)$ **fes**
        crea nou pairing
        selecciona vol de n aleatòriament
        afegeix vol al pairing
        actualitza n
    fi si
    sinó si $(q_1 < q_0)$ **fes** //Explotació
        selecciona vol de r on $[\tau_{ij}(t)]^{\alpha}[\mu_{ij}]^{\beta}$ sigui màxim
        afegeix vol al *pairing*
        actualitza n
    fi si
    sinó fes $//(q_1 > q_0)$ Exploració controlada
        selecciona vol de r aleatòriament amb $p_{ij} = \frac{[\tau_{ij}(t)]^{\alpha}[\mu_{ij}]^{\beta}}{\Sigma_1^{mida\_r}[\tau_{ij}(t)]^{\alpha}[\mu_{ij}]^{\beta}}$
        afegeix vol al *pairing*
        actualitza n
fi mentre
Acaba

Fig. 10. Flux de treball d'un agent en la creació dels *pairings*

4.3 Metodologia

Un dels temes recurrents quan es tracta d'avaluar l'eficàcia dels algorismes de colònia de formigues és la dificultat d'especificar els paràmetres d'inicialització en funció del problema que cal tractar [62]. Algunes aproximacions que apareixen en la literatura existent configuren aquests paràmetres en funció de les observacions realitzades per altres investigadors en problemes de característiques similars.

De forma inherent, aquesta metodologia pot implicar dues problemàtiques diferents: la primera associada a la implementació de l'algorisme i la segona al

tipus de dades utilitzades per a realitzar la investigació. Per evitar la influència d'aquests factors en la comparativa i anàlisi de rendiment dels dos algorismes que s'estudiaran en aquest treball i per verificar la viabilitat del nou algorisme presentat, s'ha dividit la fase d'experimentació principal en dues sub-fases:

- La fase primària o d'establiment de paràmetres
- La fase secundària o de comparació de resultats

En ambdues fases es duran a terme una sèrie d'experiments amb dades de test generades de forma específica per a realitzar la comparativa dels algorismes. Aquestes dades s'han creat de forma aleatòria amb l'ajuda d'un programa *java* implementat especialment per a dur a terme aquesta funció. En el punt 4.3.3 s'especificaran les seves característiques.

4.3.1 Establiment de paràmetres

En aquesta primera fase es realitzaran les proves de configuració per avaluar individualment i amb un mateix conjunt de test els algorismes, amb l'objectiu de minimitzar el biaix en l'anàlisi comparatiu que es podria associar amb un establiment incorrecte de les configuracions inicials. El conjunt de test escollit és el P3, com es mostra en la taula 4 (pàg. 41), ja que introdueix un grau de variabilitat amb el que és factible realitzar una valoració de l'efectivitat dels canvis introduïts[1]. Els paràmetres que es consideraran seran:

- Coeficient de persistència de feromones ρ
- Límit d'exploració q
- Ràtio $\alpha:\beta$
- Nombre de formigues k

Si bé s'estudiarà un rang de valors per aquests paràmetres basat en el que recomana Dorigo [62], també s'avaluarà el nivell de correctesa en el seu establiment en funció de la mitjana de resultats obtinguda i en el nombre d'iteracions necessàries per a convergir a una solució. Tanmateix, es valorarà el comportament d'estancament[2] en aquesta convergència per evitar casos en què la troballa d'una solució inhabiliti la cerca d'altres possibilitats. Com a valors base d'avaluació s'escolliran els que es mostren a continuació en la taula 3.

[1] Ja que en aquest treball s'estudia el problema ACSP d'optimització amb restriccions en base TSP, cal considerar que un agent tindrà una limitació considerable en la tria dels nodes destinació en la iteració i-èssima. Si s'escollís un problema amb pocs vols es limitaria l'efectivitat en l'anàlisi dels paràmetres a causa d'una convergència prematura.

[2] Es considera estancament quan es produeix una convergència ràpida i generalitzada de tots els agents a una mateixa solució, evitant que l'algorisme pugui proporcionar altres solucions viables.

p	q	$\alpha:\beta$	k
0.01	0.7	1:1	50
0.05	0.9	1:2	100
		1:5	

Taula. 3. Valors dels paràmetres dels algorismes per a avaluació

Per a realitzar una valoració estadística correcta que pal·liï la introducció d'errors causats per l'aleatorietat en l'execució dels algorismes es realitzaran $n = 35$ proves per a cada combinació de paràmetres amb un problema test aleatori tipus P3. D'aquesta forma, la distribució estadística que prendran els resultats serà aproximadament una distribució normal[1].

Problema	*Aeroports*	*Vols*	*Dies*
P1	2	30	2
P2	2	100	3
P3	2	200	5
P4	3	30	2
P5	3	100	3
P6	3	250	5
P7	5	50	2
P8	5	150	3
P9	5	300	5
P10	10	100	2
P11	10	300	3
P12	10	500	5

Taula. 4. Especificació dels problemes de test

4.3.2 Comparació de resultats

L'objectiu d'aquesta segona fase és també l'objectiu base d'aquest projecte, que és l'avaluació i comparació del rendiment dels dos algorismes presentats envers la resolució d'una sèrie de problemes establerts. Per a dur a terme aquest anàlisi es consideraran tres criteris:

- El temps requerit per a què els agents formiga convergeixin en una solució (temps d'execució o CPU).

[1] El teorema del límit central afirma que amb un conjunt de mostres prou gran $(n > 30)$, la distribució estadística de la mitjana mostral serà aproximadament una distribució normal, de forma independent a la distribució original de la variable d'estudi [110].

- El nombre d'iteracions dutes a terme per l'algorisme en aquesta convergència. En aquest cas s'estableixen dues condicions de fi d'execució:

    - El nombre màxim d'iteracions permeses: Paràmetre que s'estableix a l'inici de les execucions. Es fixa en 1000 en base a les observacions de Guang-Feng Deng i Woo-Tsong Lin en el seu estudi sobre el comportament d'algorismes ACO en el problema ACSP [1].

    - El nombre màxim d'iteracions en què es mantingui una mateixa solució: Paràmetre que controla l'estancament de l'algorisme, acabant l'execució del mateix si no es troba una solució millor que la que ja s'ha trobat. Inicialment es fixa a 100[1].

- El resultat final obtingut en la minimització de la funció de cost mostrada en l'apartat 4.1.

Com en la fase d'establiment de paràmetres, es realitzaran $n = 35$ execucions per a cadascun dels conjunts problema de test mostrats a la taula 4 per a cadascun dels dos algorismes. Amb aquests resultats es mesurarà l'MBF (*mean best fit*), calculant les mitjanes obtingudes.

Malgrat aquest fet, per a determinar correctament les diferències entre aquestes mitjanes mostrals de les diferents variables de l'experiment s'utilitzarà, en ambdós casos, una distribució *t-Student* amb un nivell de significació $\propto = 0.05$.

El t test per a dues mostres independents es basa en l'estadístic:

$$t = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{(n-1)\hat{S}_1^2 + (m-1)\hat{S}_2^2}{n+m-2}} * \sqrt{\frac{1}{n} + \frac{1}{m}}}$$

On:

- $\bar{X}$ i $\bar{Y}$ són les mitjanes de cada variable a analitzar.
- $m = n = 35$ són el nombre d'execucions per a cadascun dels problemes.
- $\hat{S}_1^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$ i $\hat{S}_2^2 = \frac{1}{m-1}\sum_{i=1}^{m}(Y_i - \bar{Y})^2$ són les quasi-variances mostrals.

[1] La utilització del paràmetre q ofereix una probabilitat $(1-q)$ per a què un agent formiga es centri en l'exploració o en l'explotació, el que possibilita sortir de situacions de convergència ràpida. Per aquest motiu s'ofereix un marge de 100 iteracions del conjunt, abans d'aturar l'execució de l'algorisme.

Els valors que s'utilitzaran per a les restriccions del problema definides en el punt 4.2 estan basats en l'estudi dut a terme per Guang-Feng Deng i Woo-Tsong Lin en [1] i seran els que s'especifiquen en la taula 5:

Paràmetre restricció	Valor	Informació
C_{t_basic}	300	Pagament bàsic per servei
C_{t_extra}	1.5	Cost per minut extra de vol
$T_{t\_minim\_vol}$	5*60	Mínim temps de vol cobert pel pagament bàsic
C_{t_hotel}	30	Cost hotel
$C_{t\_desc\_extra}$	0.1	Cost per minut de descans extra
T_{desc_minim}	30	Temps de descans mínim entre vols
C_{perdua}	0.2	Cost per temps perdut
$N_{\max\_vol}$	8	Màxim nombre de vols en un servei
$T_{\max\_desc}$	50	Temps màxim de descans entre vols abans d'anar a hotel
$T_{\max\_servei}$	14*60	Temps màxim d'un servei
$T_{max\_vol\_servei}$	11*60	Temps màxim de vol en un servei
T_{desc_m}	24*60	Temps de descans mínim després de treballar temps màxim

Taula. 5. Valors dels paràmetres de restricció en el problema

4.3.3 Conjunts de test – Entorn de proves

Pel que fa als conjunts problema de test, s'ha decidit generar-los de forma aleatòria per a poder introduir variacions personalitzades en el nombre de trajectes, aeroports i dies dels vols, amb l'objectiu de parametritzar de manera més específica les limitacions i àmbits de treball dels algorismes que es presenten en el projecte.

Tanmateix, aquesta forma de treball amb conjunts de test propis comporta la introducció d'un nivell d'incertesa relacionat amb la connectivitat del graf de vols que generen els algorismes[1]. Ja que no existeixen solucions òptimes trobades, s'ha decidit establir un mecanisme específic per evitar condicions de no acabament i penalitzar les solucions no desitjades:

[1] És possible que no existeixi una solució bona per un conjunt problema determinat i per tant, el graf de vols sigui inconnex.

- Els agents formiga construeixen les seves solucions de forma unidireccional. Un agent no pot tornar enrere després de visitar un node.
- Si en un determinat instant un agent no pot anar a cap node que compleixi amb les restriccions però la llista de nodes no visitats no és buida, generarà un *pairing* nou i seleccionarà el vol que surti més aviat d'entre aquesta llista. El realitzar aquesta acció penalitzarà amb un cost de 5000 la solució trobada.
- Si la ciutat de sortida del primer vol d'un *pairing* no és igual a la ciutat destinació de l'últim vol del mateix *pairing* també es penalitzarà la solució amb el mateix cost que el punt anterior.

Per a la generació d'aquests conjunts de dades s'ha utilitzat un programa *java* creat específicament per a aquest projecte, que utilitza cinc paràmetres (nombre d'aeroports, nombre de vols, dies, duració mínima trajecte i duració màxima trajecte) per a generar els resultats seguint les restriccions següents:

- Un vol no pot començar i acabar en la mateixa ciutat.
- La duració d'un vol entre la ciutat X i la ciutat Y serà la mateixa que entre la ciutat Y i la ciutat X.
- El temps d'arribada d'un vol serà igual al temps de sortida més el temps de duració del trajecte. Si aquest temps supera el rang de temps per un dia, el vol arribarà el dia següent en el temps restant del dia anterior.

Com es pot comprovar en la taula 4, els problemes es divideixen en quatre grups en funció del nombre d'aeroports i inclouen configuracions de 2, 3 i 5 dies i un nombre de vols que va dels 30 fins als 500. En la taula 6 es pot apreciar una mostra de les dades del problema P1.

AEROPORT SORTIDA	DIA SORTIDA	HORA SORTIDA	AEROPORT DESTINACIÓ	DIA ARRIBADA	HORA ARRIBADA
AIR0	0	11:44	AIR1	0	14:29
AIR0	0	14:55	AIR1	0	17:40
AIR1	0	15:29	AIR0	0	18:14
AIR1	1	20:23	AIR0	1	23:08
AIR1	1	14:25	AIR0	1	17:10
AIR0	1	0:19	AIR1	1	3:04
AIR0	0	9:00	AIR1	0	11:45
AIR1	0	10:14	AIR0	0	12:59
AIR1	1	12:35	AIR0	1	15:20
AIR1	1	2:06	AIR0	1	4:51

Taula. 6. Mostra dades problema P1

5. Algorisme Max-Min Ant System

5.1 Introducció

L'algorisme Max-Min Ant System va ser presentat per primera vegada per Thomas Stützle i Holger H. Hoos a [2] com a millora de l'algorisme Ant System envers problemes on la complexitat evolucionava de forma exponencial amb conjunts d'instàncies de test molt grans, com en el cas del Traveling Salesman Problem. Els investigadors varen demostrar que es podia obtenir un augment del rendiment de l'algorisme a través d'una explotació més fefaent de les millors solucions trobades durant la cerca.

Tot i que el fet d'utilitzar aquest mecanisme de cerca àvida es demostra potencialment perjudicial en l'Ant System respecte al problema d'una convergència prematura cap a solucions no optimitzades, l'algorisme MMAS evita aquesta dificultat a través de tres aspectes clau:

- Sistema elitista: Per explotar només les millors solucions en cada iteració només es permet que l'agent formiga que ha trobat la millor solució en la iteració sigui el que actualitzi els camins de feromones.
- Rang: Per evitar l'estancament de la cerca s'estableix un rang de possibles valors de feromones que està limitat en l'interval $[\tau_{min}, \tau_{max}]$.
- Valors inicials: Per a introduir una cerca deliberada en tot l'espai possible de solucions s'estableix inicialment un valor τ_{max} a tots els camins de feromones.

5.2 Funcionament

L'algorisme Max Min Ant System té la mateixa estructura de funcionament que l'algorisme base Ant System, com es pot veure en la figura fig.11., amb variacions en l'actualització de les feromones, l'establiment dels límits en els rastres i en la inicialització d'aquests rastres, com es detalla a continuació.

Algorisme Ant System
Inicialitza valors de feromones **Mentre no es compleixi la condició d'acabament fes** **Per tots els agents formiga $a \in k$ fes** **$S_a =$ Construeix solució** **Fi Per** **Aplica actualització de feromones** **Fi mentre** **Retorna millor solució**

Fig. 11. Algorisme Ant System

En cada punt en què un agent ha de realitzar una transició de l'estat actual a un estat nou, es considera la següent regla de transició d'estats per a calcular les probabilitats de seleccionar cadascun dels nous nodes disponibles. Aquesta regla depèn del paràmetre q i del paràmetre aleatori q_1 generat per cada formiga de forma independent:

$$\begin{cases} p_{ij} = \arg_{max} \left\{ [\tau_{ij}]^{\alpha} [\mu_{ij}]^{\beta} \right\} q \geq q_1 \\ p_{ij} = \dfrac{[\tau_{ij}]^{\alpha} [\mu_{ij}]^{\beta}}{\sum_1^{mida\_r} [\tau_{ij}]^{\alpha} [\mu_{ij}]^{\beta}} \; si \; q < q_1 \end{cases}$$

En el primer cas es prioritza l'explotació, mentre que en el segon s'aplica la regla heurística d'exploració. El valor heurístic μ s'inicialitza a 0 per aquells parells de vols que no compleixin les restriccions i a $\frac{1}{\sqrt{Temps\_entre\_vols}}$ altrament.

5.2.1 Actualització de les feromones

Ja que només una formiga per iteració actualitza els rastres de feromones, existeixen dues possibilitats per a dur a terme aquesta actualització: utilitzar la millor solució de la iteració o la millor solució fins al moment. En el present treball s'utilitza exclusivament la primera de les opcions. L'equació d'actualització és la següent:

$$\tau_{ij} \leftarrow (1 - p)\tau_{ij} + \Delta\tau_{ij}$$

On:

- $(1 - p)\tau_{ij}$ representa l'evaporació de les feromones en l'arc ij
- $\Delta\tau_{ij}$ representa la quantitat de feromones dipositada en l'arc ij per la formiga que ha trobat la millor solució. Si la formiga no ha passat per aquest arc $\Delta\tau_{ij} = 0$.

La quantitat de feromones $\Delta\tau_{ij}$ es calcula com:

$$\Delta\tau_{ij} = \frac{Q}{cost(millorSolució)}$$

On:

- Q és un factor de control de feromones per evitar que hi hagi una convergència a punts locals òptims de forma prematura. Inicialment s'estableix a 100, seguint els paràmetres de [1].

5.2.2 Establiment del límit de rastres

La particularitat de l'algorisme és la incorporació de límits explícits en les quantitats de feromones que es poden dipositar en un arc concret durant l'execució de cada iteració:

$$\tau_{min} \leq \tau_{ij} \leq \tau_{max}$$

Al finalitzar cadascuna de les iteracions, els valors dels rastres per cada arc han de quedar compresos en aquest interval $[\tau_{min}, \tau_{max}]$, pel que es realitzen els ajustaments següents:

$$\tau_{ij} = \tau_{max} \quad si \ \tau_{ij} > \tau_{max}$$
$$\tau_{ij} = \tau_{min} \quad si \ \tau_{ij} < \tau_{min}$$

Teòricament el valor τ_{max} es calcularia a partir del valor del cost òptim per al problema que s'estudia:

$$\tau_{max}^{teòric} = \frac{1}{\rho(cost(millorsolucióproblema))}$$

Ja que a la pràctica no es disposa del valor $cost(millorsolucióproblema)$, el càlcul de τ_{max} es duu a terme amb el cost de la millor solució trobada en la iteració actual.

Per a calcular el valor de τ_{min}, Stützle i Hoos [2] parteixen de tres assumpcions:

- La millor solució es troba just abans que s'arribi a un punt d'estancament per convergència.
- La principal influència en la construcció de la solució està determinada per la diferència relativa entre els límits màxim i mínim dels camins de feromones.
- Les opcions entre les que pot elegir un agent formiga i la probabilitat que esculli l'opció correcta en cada pas són constants.

Amb aquestes premisses plantegen l'equació:

$$\tau_{min} = \frac{\tau_{\max}(1 - \sqrt[n]{p_{best}})}{(avg - 1)\sqrt[n]{p_{best}}}$$

On:

- p_{best} és la probabilitat d'escollir la millor opció en cada punt de decisió.
- avg és la mitjana d'eleccions disponibles en cada punt de decisió i n el nombre d'eleccions totals, on $avg = n/2$.

En el nostre cas el nombre d'eleccions totals n ocasiona que $\sqrt[n]{p_{best}}$ convergeixi a 1, associant τ_{min} a 0 en totes les iteracions. Per evitar aquesta problemàtica, s'associarà un valor establert a τ_{min} dependent de l'equació $\frac{\tau_{\max}}{(avg-1)}$, on el valor avg serà calculat en cada iteració i representarà el nombre de *pairings* creats pel millor agent formiga.

5.2.2 Inicialització dels rastres de feromones

Tots els rastres de feromones s'inicialitzen amb un valor preestablert que representa el valor $\tau_{\max}$ en l'instant $t = 0$. Ja que el càlcul de $\tau_{\max}$ abans de la primera iteració no es pot dur a terme per què es desconeix el valor de la solució per al problema que cal tractar, els rastres s'inicialitzen amb un valor $Q = \tau_{\max} = 100$, com s'especifica a [1].

5.3 Resultats fase establiment de paràmetres

En la taula 7 es mostren els resultats obtinguts després de realitzar la fase d'establiment de paràmetres per a l'algorisme Max-Min Ant System. Es destaquen els resultats positius (millors minimitzacions de la funció de cost i nombre d'iteracions mitjanes) que mostren valors per sota de la mitjana total de cada columna i els resultats negatius (pitjors minimitzacions de la funció de cost) que mostren valors per sobre de la mitjana total de les columnes respectives.

Test	p	q	α: β	k	*Millor Resultat en agent*	*Millor mitjana en iteració*	*Pitjor mitjana en iteració*	*Millor mitjana en agent*	*Pitjor mitjana en agent*	*Mitjana iteracions*
1	0.01	0.7	1:1	50	131527,4	132786,254	212894,388	155770,018	229051,084	525,489
2	0.01	0.7	1:1	100	135936,9	135936,900	208287,432	151775,339	152352,079	461,411
3	0.01	0.7	1:2	50	107017,6	125161,520	202097,306	156688,829	169971,196	311,913
4	0.01	0.7	1:2	100	117170,9	128742,041	199989,691	151622,172	210362,474	510,138
5	0.01	0.7	1:5	50	109991,2	127177,602	180739,352	150844,159	180851,463	326,958
6	0.01	0.7	1:5	100	110183,0	115551,194	174937,903	145312,497	184524,562	222,969
7	0.01	0.9	1:1	50	152445,8	152865,440	212356,296	153138,716	252060,185	205,171
8	0.01	0.9	1:1	100	112088,7	132924,139	208844,372	154935,302	228693,234	501,766
9	0.01	0.9	1:2	50	122462,3	127677,364	198037,046	154423,814	211525,367	404,965
10	0.01	0.9	1:2	100	116656,4	120447,774	197234,010	151377,226	210406,250	464,768
11	0.01	0.9	1:5	50	112637,6	123785,524	181113,958	147069,145	192839,070	548,563
12	0.01	0.9	1:5	100	112136,9	119800,604	172484,462	141712,831	192455,908	472,508
13	0.05	0.7	1:1	50	127009,9	146505,880	211201,140	154465,440	226132,339	394,789
14	0.05	0.7	1:1	100	134522,4	134522,400	209143,867	142891,602	143527,306	308,151
15	0.05	0.7	1:2	50	113083,1	121474,094	203441,142	145430,894	200013,695	343,884
16	0.05	0.7	1:2	100	116888,3	122023,260	189956,582	136296,486	147529,642	249,996
17	0.05	0.7	1:5	50	118076,7	127062,484	179040,296	133679,873	141889,421	273,007

18	0.05	0.7	1:5	100	113705,5	119449,852	173470,030	137857,595	184761,888	429,077
19	0.05	0.9	1:1	50	132799,8	133378,080	212157,144	143914,775	227083,920	268,549
20	0.05	0.9	1:1	100	116355,6	124610,674	207339,582	148118,267	228457,501	381,763
21	0.05	0.9	1:2	50	111544,6	119547,728	204720,344	142590,328	196357,777	330,987
22	0.05	0.9	1:2	100	111616,7	119376,611	197519,724	139172,277	197009,491	470,317
23	0.05	0.9	1:5	50	112114,3	124995,850	170924,860	140239,365	185226,771	326,850
24	0.05	0.9	1:5	100	111938,7	126927,261	165943,937	137301,021	174339,398	298,286

Taula. 7. Resultats execució algorisme MMAS amb problema test P3 per a diferents valors de paràmetres $N_{execucions} = 24 * 35 = 840$.

Per a poder adquirir una visió més extensa del comportament de l'algorisme amb l'establiment dels diferents paràmetres s'han inclòs tant els millors resultats com els pitjors, en funció de cada agent, de cada iteració i de les mitjanes dels anteriors.

El primer que es pot comprovar amb els valors obtinguts és que no sembla poder apreciar-se una diferència significativa entre les configuracions amb valors per sota la mitjana de cada columna. Malgrat això, la representació de les dades de la fig.12. mostra tres conjunts de resultats en què les columnes més representatives per als resultats (millor resultat i millor mitjana iteració) mantenen una evolució paral·lela i similar.

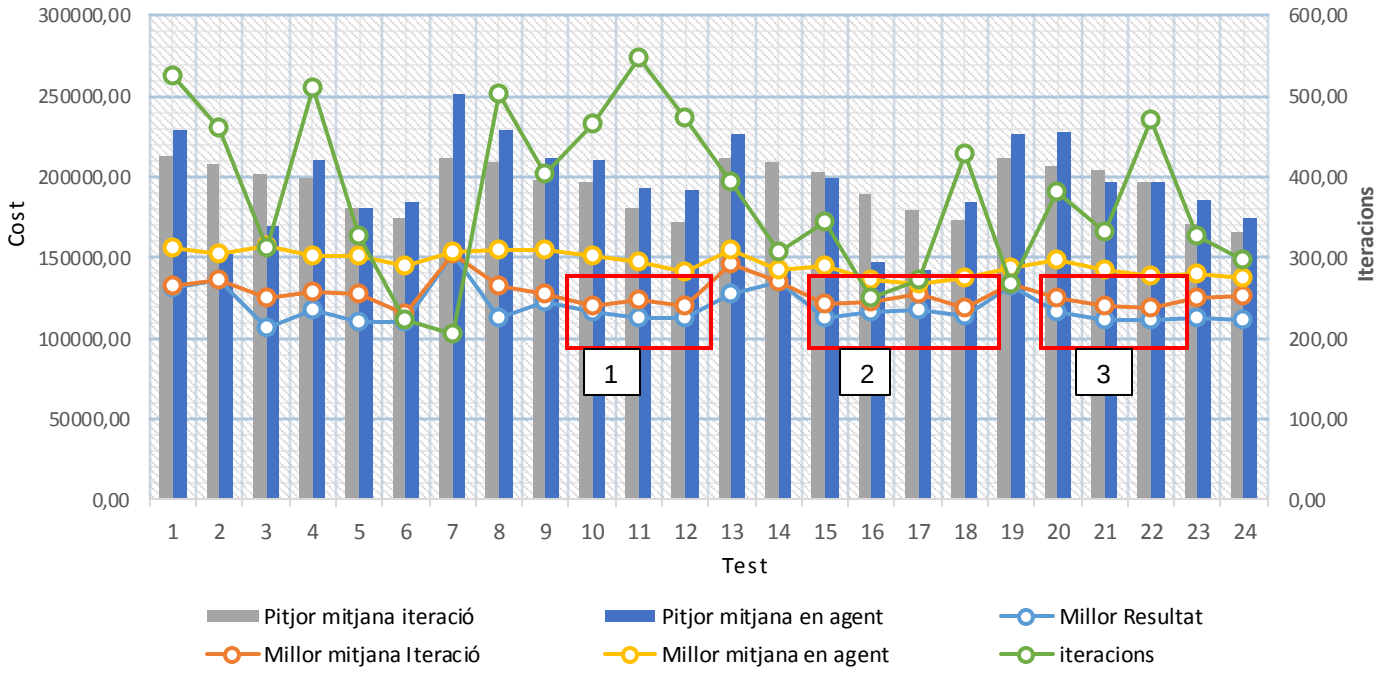


Fig. 12. Evolució dels valors de la funció de cost en l'execució del problema P3 en funció de la configuració de paràmetres $p, q, \alpha : \beta$ i k. Algorisme MMAS.

Els conjunts agrupen els tests $\{10,11,12\}$, $\{15,16,17,18\}$ i $\{20,21,22\}$, respectivament. Les configuracions de paràmetres i la representativitat dels

valors d'aquells més utilitzats en la totalitat de conjunts es pot observar més detalladament a la taula 7.

Pel que fa a l'anàlisi de la convergència prematura de resultats i tenint en compte el nombre màxim d'iteracions en què es permet mantenir una mateixa solució (establert a 100 en el punt 4.3.2), es pot veure com en els casos 15 i 16 s'arriba a una solució amb un nombre d'iteracions bastant per sota de la mitjana, pel que es considera que la configuració de paràmetres en aquest cas, no evita l'estancament en la cerca de noves solucions.

D'aquesta forma, considerant el nombre mitjà d'iteracions, la representativitat dels valors dels paràmetres en els grups escollits i els casos de test en què el millor resultat es manté en un valor proper a la mitjana del millor resultat per iteració, s'escollirà la configuració de paràmetres del test 22 per a dur a terme la fase de comparació de resultats.

Test	p	q	α:β	k
10	0.01	0.9	1:2	100
11	0.01	0.9	1:5	50
12	0.01	0.9	1:5	100
15	0.05	0.7	1:2	50
16	0.05	0.7	1:2	100
17	0.05	0.7	1:5	50
18	0.05	0.7	1:5	100
20	0.05	0.9	1:1	100
21	0.05	0.9	1:2	50
22	0.05	0.9	1:2	100
Representativitat Valor Comú	7/10	6/10	5/10	6/10

Taula. 8. Millors configuracions paràmetres per l'algorisme MMAS.

6. Algorisme Forgetful Ant System

6.1 Introducció

Com s'ha pogut veure en els punts anteriors, els algorismes de colònia de formigues tradicionals cerquen solucions a problemes d'optimització complexos a través de la conjunció entre els resultats parcials obtinguts pels agents formiga en cada iteració. Aquesta metodologia meta-heurística permet realitzar aproximacions bones per a casuístiques amb un gran nombre de variables.

Malgrat això, existeix una problemàtica associada amb els rastres de feromones que utilitzen els agents per comunicar-se de forma asíncrona: la convergència a solucions úniques. En el models més utilitzats de colònia de formigues els

dipòsits de feromones són actualitzats únicament pels agents que troben la millor solució en cada iteració [62]. Tot i que existeixen diferents mecanismes per a pal·liar aquest efecte (com per exemple l'utilitzat per l'algorisme MMAS), és la configuració dels paràmetres inicials dels algorismes el que més influencia l'estancament.

En aquest projecte es presenta l'algorisme Forgetful Ant System[1] com a variant d'estudi de l'algorisme Elitist Ant System per a la resolució del problema ACSP en forma TSP. Aquest nou algorisme està dissenyat per evitar els problemes d'estancament comentats anteriorment a través d'un mecanisme conjunt d'oblit de nodes i cerca local estesa. En el següent punt es detallarà el seu funcionament.

6.2 Funcionament

L'algorisme funciona amb un plantejament similar al que mostra la figura fig.10. del punt 4.2.1., malgrat això, les seves particularitats es mostren alhora d'addicionar nous nodes. En la figura fig.13. es mostren els passos que segueix un agent formiga en aquests casos. La regla de transició d'estats es manté igual que en el cas de l'algorisme MMAS (punt 5.2).

ForgetFul Ant System: construcció solució

```
Inici
// n: conjunt de vols pendents. Inicialment igual al conjunt total de vols m
// r: conjunt de vols que compleixen les restriccions
// s: conjunt de vols que compleixen les restriccions sense considerar el 1r vol del pairing actual

crea nou pairing
selecciona vol de m
afegeix vol al pairing
actualitza n
Mentre no s'assoleixi condició d'acabament fes
        r = selecciona vols que compleixin restriccions en funció del vol actual
        s = idem anterior sense considerar el primer vol del pairing actual si |Pairing|>1
        si (r = ∅) fes
                si (s = ∅) fes
                        crea nou pairing
                        selecciona vol de n aleatòriament
                        afegeix vol al pairing
                        actualitza n
                sinó fes
                        selecciona vol en funció de la transició d'estats
                        compara cost amb vol actual amb cost amb vol inicial anterior
                        si (costActual < costAnterior) fes
                                elimina vol inicial del pairing
                                afegeix vol actual al pairing
                                actualitza n
                        fi si
```

[1] Forgetful Ant System (FAS)

```
            fi si
        sinó
            selecciona vol en funció de la transició d'estats
            afegeix vol actual al pairing
            actualitza n
fi mentre
Acaba
```

Fig. 13. Estructura del procés de construcció de solucions per als agents en l'algorisme FAS.

El conjunt de restriccions imposades en el problema ACSP plantejat, juntament amb l'aleatorietat dels conjunts de test utilitzats ocasiona moltes situacions en què un agent no pot continuar construint un camí. En la metodologia proposada fins ara, un agent formiga era capaç de saltar a nous nodes generant *pairings* nous amb una penalització establerta. Tot i que aquest fet permetia mantenir els millors *pairings* trobats fins al moment, també impossibilitava la cerca de noves solucions amb costos més baixos.

Per introduir un procés de cerca estesa, els agents en l'algorisme FAS no generen un *pairing* nou en cada camí sense sortida sinó que abans utilitzen un mecanisme de cerca basat en la comparació entre el cost del *pairing* actual i un possible *pairing* nou seleccionat a partir del conjunt de vols que respecta les restriccions sense tenir en compte el primer vol del *pairing* actual. En la figura fig.14. es pot veure aquest comportament.

La idea és pal·liar dos efectes negatius al mateix temps: l'estancament causat pel seguiment dels rastres de feromones i les limitacions en la creació de conjunts de vols que respectin les restriccions. Aquesta mecànica de treball està dissenyada específicament pel problema ACSP en base TSP i només entra en acció en cerques amb punts morts. En última instància, en cas de no poder continuar, també s'apliquen les mateixes penalitzacions que en el cas de l'algorisme MMAS.

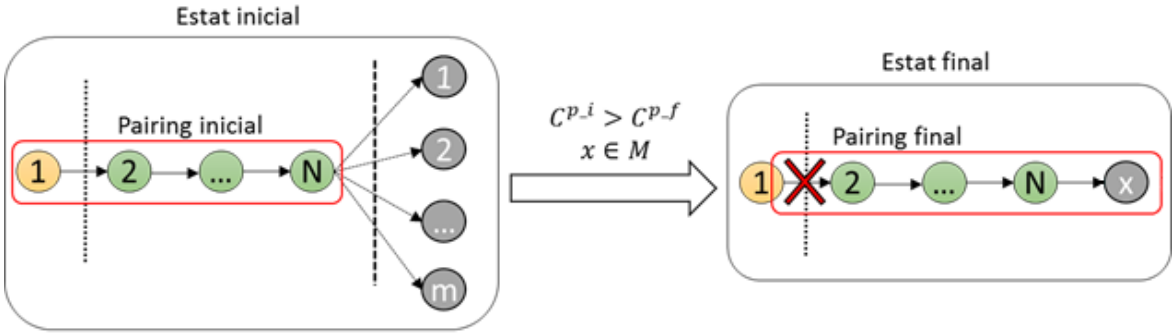


Fig. 14. Procés d'oblit de nodes en casos d'estancament en l'algorisme FAS. C^{p_i} representa el cost total del *pairing* inicial mentre que C^{p_f} representa el cost total del *pairing* temporal final. M és el conjunt de nodes que s'adapten a les restriccions del *pairing* inicial sense comptar el vol 1, i m és el nombre de vols en aquest conjunt.

Pel que fa a l'actualització dels dipòsits de feromones, l'algorisme funciona de forma similar a la variació de l'Elitist Ant System anomenada Ant System Local Best Tour [111]: en cada iteració només un nombre x d'agents és habilitat per a incrementar les quantitats de feromones en els camins que formen part dels seus recorreguts. Més concretament, en aquest nou algorisme només actualitzaran les feromones els agents que obtinguin un cost inferior a la mitjana de costos obtinguts per tots els agents en la mateixa iteració. Aquesta mesura es pot descriure dins l'equació d'actualització dels camins de feromones com:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

On:

- $\Delta\tau_{ij}^{k}$ és la quantitat de feromones que diposita la formiga elitista k en els arcs que ha visitat.
- m és el nombre de formigues elitistes en cada iteració si per cada formiga $k \in m$, $C^k < (\frac{\sum_{i=1}^{n} C^n}{n})$.
- $(1-\rho)$ representa l'evaporació de les feromones.

Pel que fa a la inicialització dels rastres de feromones, aquests es mantenen de forma inicial amb el mateix valor que en el cas de l'algorisme MMAS.

6.3 Resultats fase establiment de paràmetres

En la taula 9 es mostren els resultats obtinguts després de realitzar la fase d'establiment de paràmetres per a l'algorisme Forgetful Ant System. Es destaquen els millors i pitjors resultats per a les mateixes variables que en el cas de l'algorisme MMAS.

Test	p	q	α: β	k	*Millor Resultat en agent*	*Millor mitjana en iteració*	*Pitjor mitjana en iteració*	*Millor mitjana en agent*	*Pitjor mitjana en agent*	*Mitjana iteracions*
1	0.01	0.7	1:1	50	131608,03	131608,03	182985,66	190240,93	214832,95	356,80
2	0.01	0.7	1:1	100	122280,59	132849,41	172503,77	182440,33	170548,26	269,51
3	0.01	0.7	1:2	50	124514,40	134367,93	178010,52	181086,17	188462,93	181,53
4	0.01	0.7	1:2	100	121413,98	133208,43	171809,58	169599,21	182217,75	269,53
5	0.01	0.7	1:5	50	113214,93	123420,53	171722,57	165122,51	182587,26	197,35
6	0.01	0.7	1:5	100	108082,88	112516,76	171999,37	155474,87	187186,37	141,76
7	0.01	0.9	1:1	50	146654,68	163938,10	171045,42	194259,77	175949,19	142,49
8	0.01	0.9	1:1	100	130678,87	151215,26	179792,57	187434,66	183504,60	350,92
9	0.01	0.9	1:2	50	130762,65	135508,49	186192,49	185911,69	184718,94	315,69

10	0.01	0.9	1:2	100	113982,01	120262,24	187757,83	173615,60	175295,34	416,44
11	0.01	0.9	1:5	50	121378,83	146256,84	178348,45	178333,27	182511,45	321,91
12	0.01	0.9	1:5	100	113343,34	126833,51	174080,99	169715,84	190031,59	205,45
13	0.05	0.7	1:1	50	116616,89	134560,91	173252,72	173683,24	186127,59	334,90
14	0.05	0.7	1:1	100	113211,62	128088,86	181708,12	163526,63	189494,09	630,59
15	0.05	0.7	1:2	50	121714,48	128511,49	185143,60	171605,51	185903,08	315,08
16	0.05	0.7	1:2	100	117028,40	121227,44	172839,36	162636,74	178174,61	316,64
17	0.05	0.7	1:5	50	107653,80	122456,89	180046,42	163039,69	187839,68	232,30
18	0.05	0.7	1:5	100	108582,09	114413,34	164788,15	158195,37	172080,67	286,30
19	0.05	0.9	1:1	50	126118,16	153916,06	188124,02	182790,63	181508,17	686,94
20	0.05	0.9	1:1	100	108170,41	113357,30	176130,43	170730,46	188848,23	627,90
21	0.05	0.9	1:2	50	108777,32	143498,90	189526,05	172089,94	190351,59	573,78
22	0.05	0.9	1:2	100	117489,59	126340,26	174771,29	170680,97	171962,95	309,04
23	0.05	0.9	1:5	50	104980,20	129517,63	173001,51	169810,15	176560,36	226,04
24	0.05	0.9	1:5	100	121257,99	125604,96	172923,75	170527,57	177871,37	594,05

Taula. 9. Resultats execució algorisme FAS amb problema test P3 per a diferents valors de paràmetres $N_{execucions} = 24 * 35 = 840$.

Per a realitzar aquesta fase, s'ha partit dels mateixos valors per als paràmetres $p, q, \alpha{:}\beta$ i k que en el cas de l'algorisme MMAS. Malgrat aquest fet, tal i com es pot veure en la figura fig.15., els resultats han estat diferenciats, mostrant una alta fluctuació. De la observació d'aquests valors i la seva tendència es poden extrapolar les següents afirmacions generals:

- Quan el valor del coeficient de persistència de feromones p s'inicialitza a 0.01 (casos $1-12$), el nombre d'iteracions es manté en un interval $(200-416)$. Aquest comportament (amb l'excepció de les mostres $1, 8-11$) demostra un estancament massa ràpid que pot permetre obviar possibles millors solucions.
- Les mostres 1, 6, 15, 16, 18 i 20 mostren una convergència entre la millor mitjana per iteració i el millor resultat obtingut per un agent, el que indicaria un assoliment correcte dels objectius de configuració. En la taula 10 es poden veure les característiques concretes d'aquesta configuració.

Test	***p***	***q***	***α:β***	***k***
1	0.01	0.7	1:1	50
6	0.01	0.7	1:5	100
15	0.05	0.7	1:2	50
16	0.05	0.7	1:2	100
18	0.05	0.7	1:5	100
20	0.05	0.9	1:1	100

Representativitat Valor Comú	4/6	5/6	--	4/6

Taula. 10. Configuracions de convergència entre millors resultats per l'algorisme FAS.

- Les mostres de test 14, 19, 20, 21 i 24 són les que mostren valors mitjans d'iteracions més elevats. En aquests casos són significatius els valors de $p = 0.05$ $(5/5)$, i $q = 0.9$ $(4/5)$.

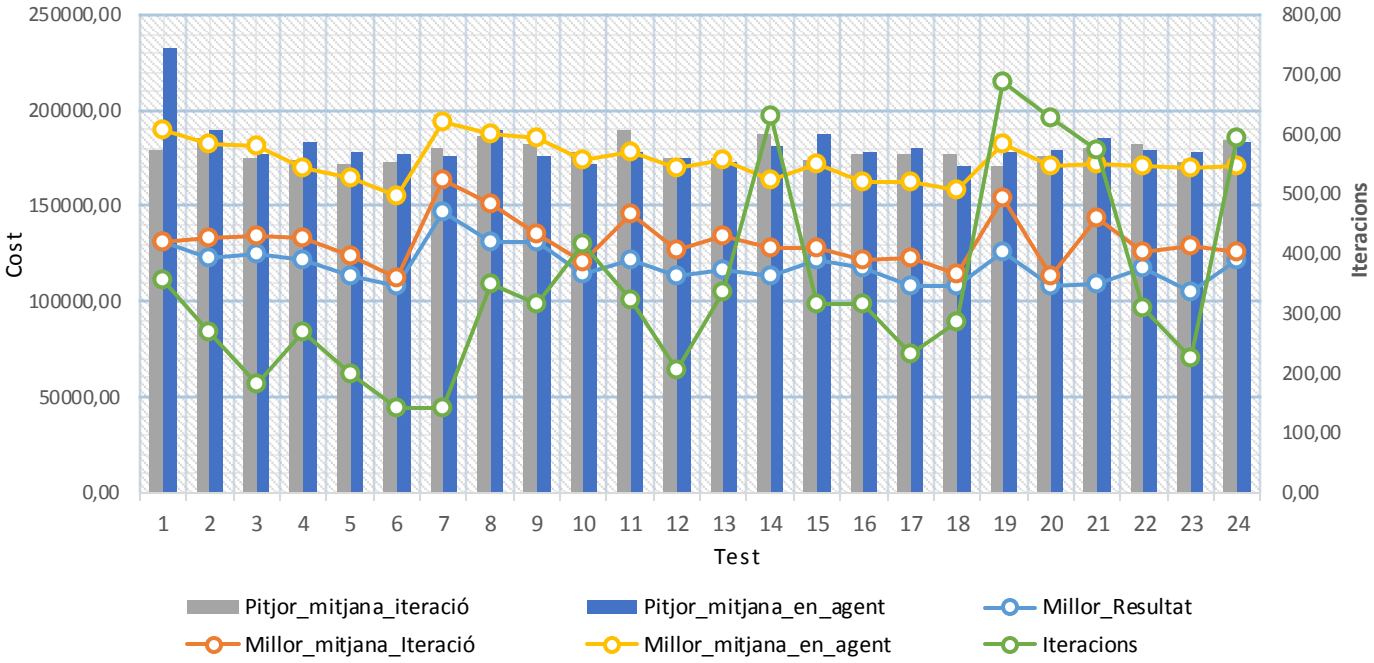


Fig. 15. Evolució dels valors de la funció de cost en l'execució del problema P3 en funció de la configuració de paràmetres $p, q, \alpha{:}\beta$ i k. Algorisme FAS.

Considerant els resultats obtinguts en què no es demostra una influència significativa de la ràtio $\alpha{:}\beta$ i no sembla existir una pauta clara de comportament que impliqui una relació directa entre una configuració específica i uns millors resultats, s'optarà per utilitzar els valors del cas test 18 ($p = 0.05, q = 0.7, \alpha{:}\beta = 1{:}5$ i $k = 100$). La motivació principal és la convergència demostrada pel millor resultat en agent i el millor resultat en iteració i la representativitat de cadascuna de les variables observada en la taula 10.

Cal mencionar la no elecció del cas 16 en comptes del 18 per una qüestió purament aleatòria derivada de la influència no demostrada de la ràtio $\alpha{:}\beta$. Els paràmetres del cas 16 són els mateixos que els del cas 18 amb excepció d'aquesta ràtio.

7. Results

In this section, we are going to compare and evaluate the performance of both Max-Min Ant System and Forgetful Ant System algorithms on the 12 different problems presented on section 4.3.3. Every test problem has been randomly generated with different parameters of airports, flights and days, in order to perform a wide analysis including different degrees of complexity.

As the ACSP problem is solved in a TSP like approach [1], flights can be seen as vertices $v \in V$ in a directed, asymmetric and weighted graph $G = (V, E)$, with limited connected components restricted by problem constrains (4.2). Edges $e \in E$ in every path construction exist only between nodes whose connectivity in a given moment depends on its compliance with the problem specifications.

To counterbalance result errors related with algorithms implementation and execution environment, both algorithms were coded following the same patterns[1] and were executed in the same platform[2]. Each algorithm was independently run 35 times on each problem to obtain normally distributed results, following the central limit theorem [110]. In addition, a termination criteria was used to determine whether the algorithms had exceeded the maximum number of iterations allowed (2000 iterations) or no better solutions had been observed (200 iterations).

Comparisons were carried out through three variables: best obtained cost in path construction, required iterations to reach a solution and time needed to obtain the solution. To analyse the results, an F-test was first performed at the 0.05 α-level to determine whether the null hypothesis of equal variances for every pair of same type-variables (one for each algorithm) was true or false. From this point and depending on the previous results, a Student t-test at the 0.05 α-level of equal or unequal variances was executed to resolve the existence of significant differences among medians of same type-variables pairs for both algorithms.

Table 11 shows a comparison of the results obtained for the minimization of the cost function, and Table 12 exhibits the average number of iterations and CPU time required to resolve the ACSP problems.

[1] Same programming language (Java™) and methodology, main functions and helper functions except specific algorithm mechanisms such pheromone update and get constrained flights for pairing construction.

[2] Intel Core i5 vPro with 8GB RAM. SO Windows 7 Professional. Java v1.8.0_65.

Problem Name	Number of flights	Number of days spanned	Number of Airports	FAS		MMAS	
				Best Cost	Average Cost	Best Cost	Average Cost
P1	30	2	2	91.051,57	91.289,11	**86.698,74**	87.673,86
P2	100	3	2	**147.746,15**	153.621,26	157.828,35	161.183,08
P3	200	5	2	134.680,84	139.741,96	**130.957,20**	136.840,11
P4	30	2	3	61.449,59	65.803,48	**51.780,10**	56.479,55
P5	100	3	3	117.827,08	128.583,20	**101.390,53**	104.624,69
P6	250	5	3	222.828,29	237.900,89	**222.610,95**	233.352,45
P7	50	2	5	**98.993,57**	106.486,10	100.652,50	106.420,02
P8	150	3	5	**231.261,68**	245.778,38	244.367,30	257.588,18
P9	300	5	5	**420.943,53**	438.964,00	424.962,28	431.809,78
P10	100	2	10	273.755,45	281.266,41	**255.671,63**	264.332,38
P11	300	3	10	480.611,39	496.745,53	**473.387,23**	487.548,56
P12	500	5	10	809.801,94	830.552,34	**795.338,56**	805.602,12
Number of best results				3		8	

Table 11. Comparison of results for the cost function

Problem Name	Number of flights	Number of days spanned	Number of Airports	FAS		MMAS	
				Iterations	CPU Time (s)	Iterations	CPU Time (s)
P1	30	2	2	237,17	0,99	**160,20**	**0,45**
P2	100	3	2	322,43	16,64	**223,83**	**7,41**
P3	200	5	2	**197,29**	**22,85**	209,11	23,91
P4	30	2	3	**232,63**	0,72	232,86	**0,51**
P5	100	3	3	277,71	12,50	**214,03**	**6,58**
P6	250	5	3	253,46	69,47	**205,34**	**34,45**
P7	50	2	5	389,91	2,25	**379,40**	**2,19**
P8	150	3	5	**258,69**	**20,17**	439,09	24,83
P9	300	5	5	**391,66**	108,67	483,91	**102,04**
P10	100	2	10	**243,23**	**6,90**	386,40	8,77
P11	300	3	10	**263,80**	**61,38**	421,20	76,87
P12	500	5	10	**382,83**	245,63	455,74	**218,86**
Number of best results				7	4	5	8

Table 12. Average iterations and CPU Time.

Problems with observed significant differences for every pair of type-variables can be found in Table 13. As can be seen, there are 10, 9 and 8 significant differences for cost, iteration and CPU time respectively. Results show the following:

- Cost function minimization:
  - Best results for MMAS: {P1, P4, P5, P10, P11, P12}
  - Best results for FAS: {P2,P7,P8,P9}
- Iterations:
  - Best results for MMAS: {P1,P2,P5,P6}
  - Best results for FAS: {P8,P9,P10,P11,P12}
- CPU time:
  - Best results for MMAS: {P1,P2,P4,P5,P6}
  - Best results for FAS: {P8,P10,P11}

Problem name		Cost	Iterations	CPU Time
P1	t-statistic	18,83122865	9,384591744	19,47176957
	P(T<=t) two tailed	2,4556E-27	1,94248E-12	4,14716E-28
	Result	S. differences	S. differences	S. differences
P2	t-statistic	-11,58654574	4,321056499	-11,58654574
	P(T<=t) two tailed	9,44547E-18	6,42163E-05	9,44547E-18
	Result	S. differences	S. differences	S. differences
P3	t-statistic	1,768233267	-0,970621252	-0,730830352
	P(T<=t) two tailed	0,081507269	0,335176827	0,467393527
	Result	**No S. Diff.**	**No S. Diff.**	**No S. Diff.**
P4	t-statistic	17,07476474	-0,014373637	5,540254352
	P(T<=t) two tailed	7,30294E-24	0,988598376	7,38427E-07
	Result	S. differences	**No S. Diff**.	S. differences
P5	t-statistic	12,61159681	3,515212189	8,258944222
	P(T<=t) two tailed	1,80213E-19	0,00078693	4,32698E-11
	Result	S. differences	S. differences	S. differences
P6	t-statistic	0,134512651	3,359314555	13,1144574
	P(T<=t) two tailed	0,893394554	0,001557627	2,08848E-19
	Result	**No S. Diff.**	S. differences	S. differences
P7	t-statistic	-2,092099551	0,368379681	0,334593983
	P(T<=t) two tailed	0,040530929	0,713734498	0,738961247
	Result	S. differences	**No S. Diff.**	**No S. Diff.**
P8	t-statistic	-8,725582862	-6,043489787	-2,697359283
	P(T<=t) two tailed	1,0577E-12	4,9621E-07	0,010097677
	Result	S. differences	S. differences	S. differences
P9	t-statistic	-2,452985567	-3,199635887	0,993696177
	P(T<=t) two tailed	0,016737317	0,002285335	0,323893719
	Result	S. differences	S. differences	**No S. Diff.**
P10	t-statistic	15,78155291	-6,964439721	-3,935031652
	P(T<=t) two tailed	2,00191E-24	2,39811E-08	0,000306602
	Result	S. differences	S. differences	S. differences
P11	t-statistic	3,63409571	-6,017773126	-2,998989913
	P(T<=t) two tailed	0,000586349	2,53002E-07	0,004149327
	Result	S. differences	S. differences	S. differences

P12	t-statistic	5,006416153	-2,108157975	1,380889725
	P(T<=t) two tailed	5,01662E-06	0,038705275	0,17227039
	Result	S. differences	S. differences	**No S. Diff.**

Table 13. Student's t-test results. Results show cases with significative differences (S.differences) and with no significative differences (No S. Diff.).

In some of the cases, the MMAS algorithm obtains better results but the use of temporal extended search through forgetting nodes in FAS has been shown to be a good candidate mechanism to avoid stagnation in spite of using an elitist type strategy for pheromone trail update process [112, pag. 14]. Best results for FAS in cost function minimization also denote a promising success ratio of 40%, which provides evidences of how the forgetfulness allows mitigating the effects of stagnation and derived poor performances.

CPU time in most of the FAS results indicate an expected longer time to find good solutions, derived from the new search in dead ends[1] function, whose time complexity is always higher than the constrained-search in the MMAS algorithm. FAS must perform new searches whenever it builds a temporary pairing that fits the constraints of the problem.

Iteration results for both algorithms show that half of the best results for FAS (P8 and P9) obtain also best results for number of iterations, and only two (P1 and P5) out of six problems for the MMAS algorithm show the same behavior. As can be seen in Fig. 16., MMAS iterations trend to follow a fast growing linear model that depends on the problem's complexity, in contrast to FAS, whose values remain in a bounded 200 iterations zone.

Fig. 16. Graphical evolution for FAS and MMAS iteration values for executed problems.

[1] FAS algorithm forgetful approach performs a temporal new search when no more flights can be added to a created pairing, trying to find less costly solutions.

8. Conclusions

8.1 General conclusions

En el present treball s'ha realitzat una exposició generalista dels conceptes més significatius en el camp de la intel·ligència artificial, i s'han introduït de forma més específica els mecanismes d'intel·ligència d'eixam existents actualment per a resoldre problemes d'optimització en combinatòria complexos. Han quedat patents les diferents opcions i les diverses línies de treball presents en aquest àmbit, juntament amb les seves característiques particulars.

El procés de cerca, documentació i redacció ha permès no només veure la importància creixent d'aquesta àrea basada en el comportament de sistemes naturals en el context actual de fort desenvolupament tecnològic, sinó també avaluar la dificultat de descobrir i establir nous mecanismes per a resoldre problemes d'optimització complexos.

En aquest sentit, cal fer una menció especial al desenvolupament del nou algorisme de colònia de formigues, amb el que tot i haver assolit l'objectiu de crear una nova variant dels algorismes ACO, s'ha posat de relleu la impossibilitat de poder mantenir una fidelitat completa en reflex de tots els treballs duts a terme en el mateix àmbit. Malgrat el gran avantatge que suposa tenir accés a la xarxa i a bases de dades d'articles d'investigació per a dur a terme projectes de recerca, s'ha trobat a faltar una metodologia específica per a obtenir les dades fonamentals i realitzar un triatge actuant amb criteri objectiu.

Pel que fa a la planificació i execució de les tasques plantejades inicialment, cal comentar la dificultat que ha suposat seguir els diferents terminis d'entrega preestablerts com a conseqüència, principalment, de la disponibilitat de temps per a poder dedicar al propi treball i de la complexitat d'obtenir i comparar la diversa informació cercada. A pesar d'aquest fet, finalment s'han pogut mantenir les línies generals de treball tot i no haver pogut ampliar la comparativa del nou algorisme amb altres algorismes ACO relacionats.

8.2 Conclusions comparing FAS and MMAS algorithms

This project presented the Forgetful Ant System (FAS) algorithm as a variant of the Elitist Ant System (EAS) algorithm for solving the ACSP problem in a TSP approach. Conclusions about the new called forgetful mechanism in FAS compared to the MMAS algorithm approach (bounded pheromone trails) in relation to fast convergence to local optima points cases, are described below.

Values obtained comparing MMAS and FAS performance through the ACSP problem show promising results using the new forget node mechanism as an

alternative or complementation to avoiding stagnation approaches used in many different ACO algorithms:

- **ACS**: Dorigo and Gambardella introduced a global state transition rule in the Ant Colony System ACS algorithm [69] for balancing between exploration and exploitation.
- **MMAS**: Stützle and Hoos [2], used a proportional mechanism so called 'trail smoothing mechanism' to minimize the relative differences between the trail weights in order to favor the global exploration of new paths.
- **COAC**: Hu, Zhang and Li [3] proposed an 'adaptive regional radius' method to reduce the probability of being caught in a local optima, improving the global search capability.
- **HyperCube Framework:** Dorigo, Blum and Roli [71], showed a new way of implementing ACO algorithms based on treating vectors of pheromone values as a $|C|$-dimensional vectors $\vec{\tau}$ moving in a $|C|$-dimensional hyper-space defined by the range of values assumable for pheromone trail parameters. This approach allowed normalizing the amounts of pheromone values avoiding stagnation through a controlled restarting of the ACO algorithms in a biased way.

Regarding the importance of pheromone trails in ACO algorithms as oriented exploration mechanism (exploitation) in contrast to global non-guided search mechanisms, the new FAS algorithm served as a model to compare two general different approaches (both with successful results):

- Avoiding local optima by changing the relative importance of the pheromone trails.
- Using 'restart' mechanisms to extend local search spaces or to introduce new search spaces in dead-end and stagnated cases.

In this work, has been demonstrated that the new mechanism has allowed avoiding expected premature convergence behaviour observed by Dorigo and Di Caro in the Elitist Ant System algorithm [112, pag. 14]. Despite of that, these early results only represent a preliminary study showing the potential for the introduction of changes in the search processes carried out by ant agents individually. From this point, it would be necessary to conduct a broader investigation centred on mathematical analysis and probabilistic variations to use this mechanism together with other pheromone updating mechanisms that have been proved to be more efficient, such the one used in the MMAS case [68].

Other possible future research in this field might be related to the analysis of the ACO algorithms configurations influence depending on the type of optimization problem to solve, and also with the creation of a general framework for developing new algorithms based on swarm behaviour in the same manner that HyperCube Framework does with ACO algorithms.

9. Glossari

10. Bibliografia

[1] Guang-Feng Deng and Woo-Tsong Lin. Ant Colony Optimization-based algorithm for airline crew scheduling problem. Expert Systems with Applications 38 (2011) 5787-5793.

[2] T. Stützle and H.H. Hoos, *MAX MIN Ant System*, Future Generation Computer Systems, volume 16, pages 889-914. Elsevier, 2000.

[3] Hu, X. and Zhang, J. and and Li, Y. Orthogonal methods based ant colony search for solving continuous optimization problems. *JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY*. Springer, 2008.

[4] Aloysius George and B. R. Rajakumar, *Fuzzy Aided Ant Colony Optimization Algorithm to Solve Optimization Problem*, Intelligent Informatics, Advances in Intelligent Systems and Computing, volume 182, pages 207-215.

[5] Abbas Afshar, Fariborz and Massoumi, Amin Afshar and Miquel A. Mariño. *State of the Art Review of Ant Colony Optimization Applications in Water Resource Management*, Water Resources Management, September 2015, Volume 29, Issue 11, pp 3891-3904

[6] McCarthy, John and Minsky, Marvin and Rochester, Nathan and Shannon, Claude. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. 1955

[7] Engelbrecht, Andries P. Computational Intelligence. Wiley, 2007.

[8] Felix T.S. Chan and Manoj Kumar Tiwari. Preface: Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, Felix T.S.Chan and Manoj KumarTiwari (Ed.) InTech, 2007.

[9] Xin-She Yang. Swarm-Based Metaheuristic Algorithms and No-Free-Lunch Theorems, Theory and New Applications of Swarm Intelligence, Dr. Rafael Parpinelli (Ed.). InTech, 2012.

[10] Mehmet Sevkli and Aise Zulal Sevkli. A Stochastically Perturbed Particle Swarm Optimization for Identical Parallel Machine Scheduling Problems, Bio-Inspired Computational Algorithms and Their Applications, Dr. Shangce Gao (Ed.). InTech, 2012.

[11] Bonabeau, Eric and Dorigo, Marco and Theraulaz, Guy. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, 1999.

[12] Hazem, Ahmed and Glasgow, Janice. Swarm Intelligence: Concepts, Models and Applications. Technical Report, 2012.

[13] Engelbrecht, Andries. Fundamentals of Computational Swarm Intelligence. Wiley & Sons, 2005.

[14] Zhang, H. T. and Cheng, Z. and Fan, M. C. and Wu, Y. Collective behavior coordination with predictive mechanisms: Understanding Complex Systems. IEEE Circuits and Systems Magazine, 2016.

[15] Blum, Christian and Merkle, Daniel (Eds.) Swarm intelligence: Introduction and Applications. Springer-Verlag Berlin Heidelberg, 2008.

[16] Dorigo, Marco and Birattari, Mauro and Garnier, Simon and Hamann, Heiko and Montes de Oca, Marco and Solnon, Christine and Stützle, Thomas (Ed.) Swarm Intelligence. 9th International Conference, ANTS 2014, Brussels, Belgium, September 10-12 2014. Proceedings. Springer, 2014.

[17] Tan, Ying and Shi, Yuhui and Mo, Hongwei (Eds.) Advances in Swarm Intelligence. 4th International Conference, ICSI 2013 Harbin, China, June 2013. Proceedings. Springer, 2013.

[18] Tao, Yong-Qin and Cui, Du-Wu and Miao, Xiang-Lin and Chen, Hao. An Improved Swarm Intelligence Algorithm for Solving TSP Problem. Springer Berlin Heidelberg, 2007.

[19] Cheng, Shi and Shi, Yuhui and Qin, Quande and Bai, Ruibin. Swarm Intelligence in Big Data Analytics. Springer Berlin Heidelberg, 2013.

[20] Winklerová, Zdenka. Maturity of the Particle Swarm as a Metric for Measuring the Collective Intelligence of the Swarm. Springer Berlin Heidelberg, 2013.

[21] Chu, Shu-Chuan and Huang, Hsiang-Cheh and Roddick, John F. and Pan, Jeng-Shyang. Overview of Algorithms for Swarm Intelligence. Springer Berlin Heidelberg, 2011.

[22] Merkle, Daniel and Middendorf, Martin. Swarm Intelligence. Springer US, 2005.

[23] Abbasi, Sadrollah and Manteghi, Sajad and Heidarzadegan, Ali and Nemati, Yasser and Parvin, Hamid. A Robust Clustering via Swarm Intelligence. Springer International Publishing, 2015.

[24] Liu, Yanmin and Niu, Ben and Chan, Felix T.S. and Liu, Rui and Changling, Sui. Application of Disturbance of DNA Fragments in Swarm Intelligence Algorithm. Springer International Publishing, 2015.

[25] Rahman, Imran and Vasant, Pandian and Singh, BalbirSinghMahinder and Abdullah-Al-Wadud, M. Hybrid Swarm Intelligence-Based Optimization for Charging Plug-in Hybrid Electric Vehicle. Springer International Publishing, 2015.

[26] Nilsson, Nils J. The Quest for Artificial Intelligence: A History Of Ideas And Achievements. Cambridge University Press, 2010.

[27] Turing, Alan. Computing Machinery and Intelligence. Mind, New Series, Vol. 59, No. 236 (Oct., 1950), pp. 433-460.

[28] Russell, Stuart and Norvig, Peter. Artificial Intelligence: A Modern Approach 3rd Edition. Prentice Hall, 2009.

[29] Yang, Xin-She and Karamanoglu, Mehmet. Swarm Intelligence and Bio-Inspired Computation: An Overview. Elsevier Inc. 2013.

[30] Ducatelle, Frederick and Di Caro, Gianni A. and Gambardella, Luca M. Principles and applications of swarm intelligence for adaptive routing in telecommunications networks. Springer International Publishing, 2010.

[31] Das, Swagatam and Abraham, Ajith and Konar, Amit. Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives. Springer International Publishing, 2008.

[32] Duch, Wlodzislaw. What is Computational Intelligence and what could it become?, Duch, Wlodzislaw and Mandziuk, J. (eds) Challenges for Computational Intelligence. Studies in Computational Intelligence (J. Kacprzyk Series Editor), vol. 63. Springer, Heidelberg, 2007.

[33] Shaffer, D.R. Psicología del desarrollo: Infancia y Adolescencia. Internacional Thompson, 2000.

[34] Chang, Weng-Long and Huang, Shu-Chien and Lin, KawuuWeicheng and Ho, Michael (Shan-Hui). Fast parallel DNA-based algorithms for molecular computation: discrete logarithm. Springer US, 2011.

[35] Pandey, SubhashChandra and Nandi, GoraChand. Convergence of knowledge, nature and computations: a review. Springer Berlin Heidelberg, 2014.

[36] Dodig-Crnkovic, Gordana. Significance of Models of Computation, from Turing Model to Natural Computation. Springer Netherlands, 2011.

[37] Khalid, H. and Obaidat, M. S. Application of Neural Networks to Cache Replacement. Springer-Verlag, 1999.

[38] Rothwell, John. Control of Human Voluntary Movement: Cerebral cortex. Springer Netherlands, 1994.

[39] De Jong, Kenneth A. Evolutionary Computation: A Unified Approach. Massachussets Institute of Technology, 2006.

[40] Weise, Thomas. Global Optimization Algorithms: Theory and Application. GNU Licensed, 2009.

[41] Mayr, Ernst. The Growth of Biological Though. Harvard University Press, 1982.

[42] Darwin, Charles. On the Origin of Species. John Murray (Ed.) (1859) http://www.gutenberg.org/files/1228/1228-h/1228-h.htm, October 2015.

[43] Eiben, A.E. And Smith, J.E. Introduction to evolutionary computing. Springer Verlag, 2003

[44] Feoktistov, Vitaliy. Differential Evolution: In Search of Solutions. Springer US, 2006.

[45] Sivanandam, S.N. and Deepa, S.N. Introduction to Genetic Algorithms. Springer Berlin Heidelberg, 2008.

[46] Worzel, Bill and Riolo, Rick. Genetic Programming Theory and Practice (Vol 6.). Springer US. 2003.

[47] Morrison, Ronald W. Designing Evolutionary Algorithms for Dynamic Environments. Springer Berling Heidelberg, 2004.

[48] Beni, G., and J. Wang. Swarm Intelligence (Proceedgins Seventh Annual Meeting of the Robotics Society of Japan). RSJ Press, Tokyo, 1989.

[49] Aickelin, Uwe and Dasgupta, Dipankar. Artificial Immune Systems. Springer US, 2005.

[50] Wikipedia – Immune System, October 2015.

[51] N. K. Jerne. Towards a network theory of the immune system. Ann Immunol, Paris, 1974.

[52] Castro, L. N. de and Timmis, J. I. Artificial immune systems as a novel soft computing paradigm. Springer-Verlag, 2003.

[53] Aickelin, Uwe and Dasgupta, Dipankar. Artificial Immune Systems. Springer US, 2005.

[54] Herrmann, ChristophS. Fuzzy Logic as interfacing technique in hybrid AI-systems. Springer Berlin Heidelberg, 1997.

[55] Mencar, Corrado. Interpretability of Fuzzy Systems. Springer International Publishing, 2013.

[56] Wikipedia – Fuzzy set, October 2015.

[57] Wikipedia – Fuzzy logic, October 2015.

[58] Giardina, Irene. Collective behaviour in animal groups: theoretical models and empirical studies. HFSP Publishing, 2008.

[59] Sumpter, David J.T.. Collective Animal Behaviour. Princeton University Press, 2010.

[60] Vela-Pérez and M., Fontelos and M.A. and Garnier, S. From individual to collective dynamics in Argentine ants (Linepithema humile). Elsevier Inc, 2015.

[61] Camazine, Scott and Deneubourg, Jean-Louis and Franks, Nigel R. and Sneyd, James and Theraulaz, Guy and Bonabeau, Eric. Self-Organization in Biological Systems. Princeton University Press, 2003.

[62] Dorigo, Marco and Stützle, Thomas. Ant Colony Optimization. Massachussets Institute of Technology, 2004.

[63] Ebling, M. and Di Loreto, M. and Presley, M. and Wieland, F. An Ant Foraging Model Implemented on the Time Warp Operating System, Proceedings of the SCS Multiconference on Distributed Simulation, 1989.

[64] Herbers, Joan M. and Choiniere, Eric. Foraging behavior and colony structure in ants. Elsevier, 1996.

[65] Dorigo, M. Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, 1992.

[66] Dorigo, M. and Stutzle, Thomas. The Ant Colony Optimization Metaheuristic: Algorithms, Applications and Advances. Springer US, 2003.

[67] Dorigo, M. and Maniezzo, V. and Colorni, A. The ant system: Optimization by a colony of cooperating agents. IEEE Systems, Man, and Cybernetics Society, 1996.

[68] Dorigo, M. and Blum, Christian. Ant colony optimization theory: A survey. Elsevier, 2005.

[69] Dorigo, M. and Gambardella, L.M. Ant colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, Vol.1, No.1, 1997.

[70] Bullnheimer, Bernd and Hartl, Richard F. and StrauB, Christine. A New Rank Based Version of the Ant System: A computational Study. Central European Journal for Operations Research and Economics, 1997.

[71] Blum, C. and Roli, Andrea and Dorigo, M. The hyper-cube framework for ant colony optimization. IEEE Transactions on Systems, Man, and Cybernetics, Vol.34, Issue 2, 2004.

[72] Socha, Krysztof and Dorigo, M. Ant colony Optimization for continuous domains. Elsevier, 2006.

[73] Bezerra, L. and Goldbarg, E. and Goldbarg, M. and Buriol, L. Analyzing the impact of MOACO components: An algorithmic study on the multi-objective shortest path problem. Elsevier, 2012.

[74] Angus, D. and Woodward, C. Multiple objective ant colony optimization. Springer Science & Business Media, 2008.

[75] García-Martínez, C. and Cordón, O. and Herrera, F. A taxonomy and en empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. Elsevier, 2007.

[76] Wikipedia – Pareto efficiency, November 2015.

[77] Hamed Zamani Sabzi and Delbert Humberson and Shalamu Abudu and James Phillip King. Optimization of adaptive fuzzy logic controller using novel combined evolutionary algorithms, and its application in Diez Lagos flood controlling system, Southern New Mexico. Elsevier, 2015

[78] Dávila Patrícia Ferreira Cruz and Renato Dourado Maia and Leandro Augusto da Silva and Leandro Nunes de Castro. BeeRBF: A bee-inspired data clustering approach to design neural network classifiers. Elsevier, 2015.

[79] Mostafa Mahi and Ömer Kaan Baykan and Halife Kodaz. A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem. Elsevier, 2015.

[80] Mohamed M.S. Abdulkader and Yuvraj Gajpal and Tarek Y. ElMekkawy. Hybridized ant colony algorithm for the Multi Compartment Vehicle Routing Problem. Elsevier, 2015.

[81] Marzband, M. and Yousefnejad, E. and Sumper, A. and Domínguez-García, J.L. Real time experimental implementation of optimum energy management system in standanlone Microgrid by using multi-layer ant colony optimization. Elsevier, 2015.

[82] Chávez, J.J.S. and Escobar, J.W. and Echeverri, M.G. A multi-objective pareto ant colony algorithm for the multi-depot vehicle routing problem with backhauls. International Journal of Industrial Engineering Computation 7, 2016.

[83] Li, B.H. and Lu, M. and Shan, Y.G. and Zhang. H. Parallel ant colony optimization for the determination of a point heat source position in a 2-D domain. Elsevier, 2015.

[84] Wang, G. and Chu, H.E. and Zhang, Y. and Chen, H. and Hu, W. and Li, Y. and Peng, X. Multiple parameter control for ant colony optimization applied to feature selection problem. Springer London, 2015.

[85] Vijayalakshmi, P. and Francis, S.A.J. and Dinakaran, J.A. A robust energy efficient ant colony optimization routing algorithm for multi-hop ad hoc networks in MANETs. Springer US, 2015.

[86] Dong, Yi. And Zhao, S. and Ran, H.D. and Li, Y. and Zhu, Z. Routing and wavelength assignment in a satellite optical network based on ant colony optimization with the small window strategy. Optical Society of America, 2015.

[87] Wikipedia – Case-based reasoning, November 2015.

[88] Wikipedia – Cognition, November 2015.

[89] Wikipedia – Microbat, November 2015.

[90] Wikipedia – B cell, November 2015.

[91] Wikipedia – Stochastic Process, November 2015.

[92] Wikipedia – Traveling Salesman Problem, November 2015.

[93] Yen, J. and Birge, J. A stochastic programming approach to the airline crew scheduling problem. Technical report, University of Washington, 2000.

[94] Minoux, M. Column generation techniques in combinatorial optimization: a new application to crew pairing problems. In Proceedings XXIV AGIFORS Symposium, 1984.

[95] Makri, A. and Klabjan, D. Efficient column generation techniques for airline crew scheduling. Technical report, University of Illinois, 2001.

[96] Lettovský, L. and Johnson, E. and Nemhauser, G. Airline crew recovery. Transportation Science, 2000.

[97] Klabjan, D. and Johnson, E. and Nemhauser, G. and Gelman, E. and Ramaswamy, S. Solving large airline crew scheduling problems: Random pairing generation and strong branching. Computational Optimization and Applications, 2001.

[98] Gopalakrishnan, B.and Johnson, Ellis.L. Airline Crew Scheduling: State-of-the-Art. Kluwer Academic Publishers, 2005.

[99] Barnhart, Cynthia and Cohn, AmyM. and Johnson, EllisL. and Klabjan, Diego and Nemhauser, GeorgeL. and Vance, PamelaH. Handbook of Transportation Science: Airline Crew Scheduling – 517-560. Springer US, 2003.

[100] Crawford, Broderick and Castro, Carlos and Monfroy, Eric. A Constructive Hybrid Ant Algorithm for the Airline Crew Pairing Problem. Springer Berlin Heidelberg, 2006.

[101] Aguiar, Bruno and Torres, José and Castro, António J.M. Operational Problems Recovery in Airlines: A Specialized Methodologies Approach. Springer Berlin Heidelberg, 2011.

[102] Zhi-Gang Ren and Zu-Ren Feng and Liang-Jun Ke and Zhao-Jun Zhang. New ideas for applying ant colony optimization to the set covering problem. Elsevier, 2010.

[103] Pierce, Brian. Economic Performance of the Airline Industry. Mid-year 2015 forecast presentation. IATA, 2015.

[104] Kasirzadeh, A. and Saddoune, M. and Soumis, F. Airline Crew Scheduling: models, algorithms and data sets. Springer-Verlag Berlin Heidelberg, 2015.

[105] Özdemir, U. Methodology for crew-pairgin problem in airline crew scheduling. Marmara University, 2004.

[106] Kohl, N. and Karisch, S. E. Airline Crew Rostering: Problem types, modeling, and optimization. Kluwer Academic Publishers, 2004.

[107] Garey, M. R. and Johnson, D.S. Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman & Co, 1979.

[108] Wikipedia – Knapsack problem, November 2015.

[109] Ozdemir, H. T. and Mohan C.K. Flight graph based genetic algorithm for crew scheduling in airlines. Elsevier, 2001.

[110] Grinstead, C. M and Snell, J. L. Introduction to Probability. American Mathematical Society, 1997.

[111] Tony White, Simon Kaegi, and Terri Oda. Revisiting Elitism in Ant Colony Optimization. Springer-Verlag, 2003.

[112] Dorigo, Marco and Di Caro, Gianni. Ant Algorithms for Discrete Optimization. Artificial Life, MIT Press, 1999.

Apèndix A. Detalls de la implementació dels algorismes

A.1 Entorn i programari utilitzat

La implementació dels algorismes utilitzats en aquest projecte ha estat realitzada en llenguatge de programació Java™[1]. Concretament, s'ha utilitzat l'entorn de desenvolupament o JDK[2] versió 8u65 juntament amb l'IDE[3] Eclipse[4] versió Mars[5]. Els experiments han estat duts a terme en la següent plataforma:

- Intel Core i5 vPro with 8GB RAM. SO Windows 7 Professional.

El programari que es requereix per a dur a terme l'execució és:

- Java JRE[6] 8

A.2 Estructura dels algorismes

Cadascun dels algorismes ha estat modelitzat a partir de quatre classes diferents. A continuació s'exposen les seves característiques principals:

- **ACSP[7]**: És la classe que emmagatzema totes les variables associades a les restriccions imposades en el problema que cal minimitzar. També és la que conté la funció d'execució principal o *main*, amb la que s'inicien i es controlen els paràmetres d'execució, el nombre d'execucions, la presentació dels resultats, ... En els punts següents es detallarà el seu funcionament.
- **Ant[8]**: Representa un agent formiga. Conté metodologies per a dur a terme la construcció d'un conjunt de *pairings* en funció de les restriccions del problema i dels paràmetres d'execució. Es diferent en cadascun dels algorismes.
- **Flight[9]**: Conté la informació referent a un vol concret.
- **ForgetfulAntSystem[10] / minMaxAntSystem[11]**: Cadascun dels algorismes utilitzats en aquest projecte utilitza una de les dues classes especificades en funció de la seva metodologia pròpia de funcionament. Aquestes classes són les encarregades de crear l'entorn de colònia de formigues, recopilar els

[1] https://www.java.com/en/about/
[2] Java Standard Edition Development Kit (JDK™)
[3] Integrated Development Environment (IDE)
[4] http://help.eclipse.org
[5] Release 4.5.1. Build id: 20150924-1200
[6] Java Runtime Environment (JRE)
[7] Arxiu ACSP.java
[8] Arxiu ant.java
[9] Arxiu flight.java
[10] Arxiu forgetfulAntSystem.java
[11] Arxiu minMaxAntSystem.java

resultats de cadascun dels agents, aplicar els càlculs referents als costos obtinguts i representar la informació.

A continuació es poden apreciar els diagrames UML de les classes que interactuen en ambdós algorismes. La figura A.1 és la representació de l'algorisme MMAS mentre que la figura A.2 és la representació de l'algorisme FAS.

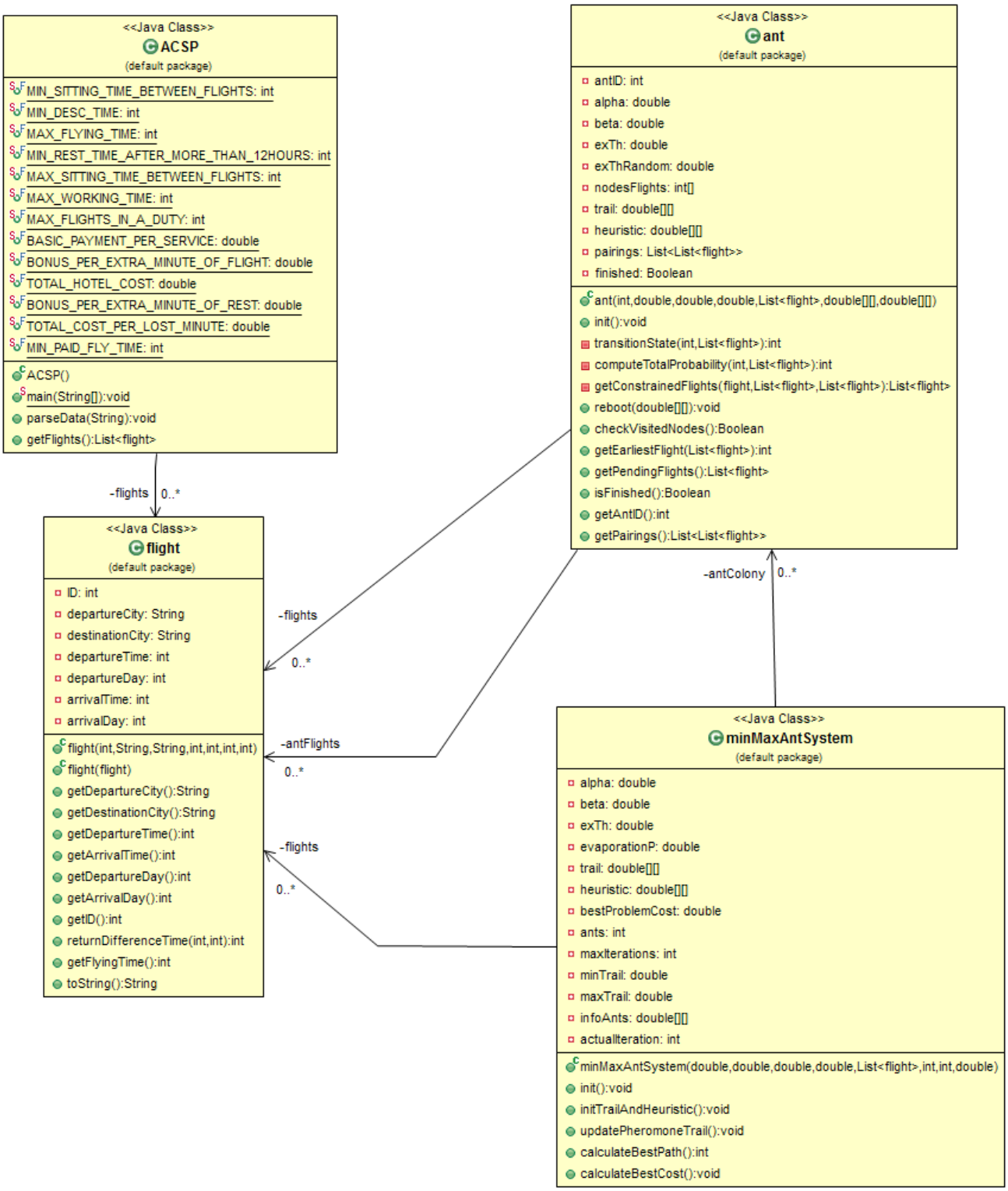


Fig. A.1. Representació UML de la implementació de l'algorisme MMAS.

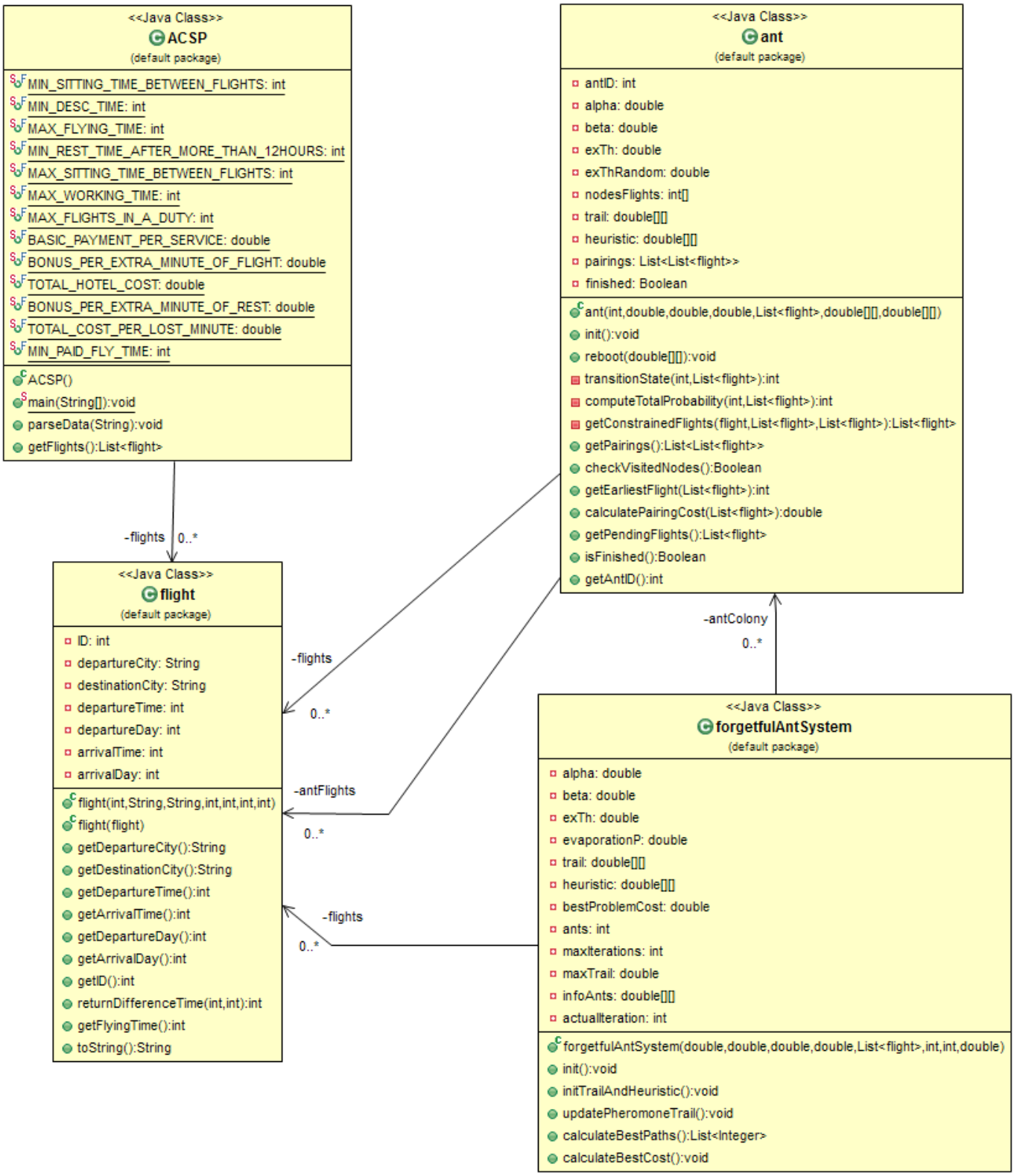


Fig. A.2. Representació UML de la implementació de l'algorisme FAS.

A.3 Obtenció de resultats

Com s'ha comentat en el punt anterior, existeix una classe anomenada ACSP per a cadascun dels algorismes. Aquesta és la que controla els paràmetres d'entrada per a dur a terme l'execució. A continuació se'n detallarà el funcionament de forma específica.

Els paràmetres que es poden modificar en dur a terme una execució són els següents:

- Alpha
- Beta
- Límit d'exploració q
- Ràtio d'evaporació
- Nombre de formigues
- Ruta absoluta per a l'arxiu amb les dades del problema

Existeixen dos modes d'execució en funció dels paràmetres passats a cadascun dels arxius java[1] representatius de cada algorisme. En aquest cas s'utilitzarà com a exemple el de l'algorisme FAS:

- **Execució predeterminada**: Només cal passar com a paràmetre la ruta absoluta de l'arxiu de test que es vol utilitzar com a entrada de dades del problema a resoldre.

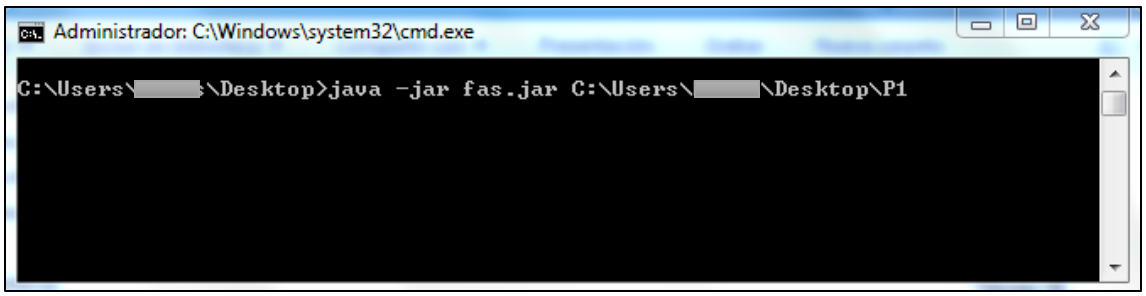


Fig. A.3. Exemple d'execució predeterminada de l'algorisme FAS

En aquest cas, l'algorisme utilitzarà els paràmetres considerats com a òptims, especificats en cadascuna de les seccions corresponents d'aquest projecte (secció 5.3 en l'algorisme MMAS i secció 6.3 en l'algorisme FAS).

[1] Arxius fas.jar i mmas.jar

- **Execució específica**: Cal especificar tots els paràmetres comentats anteriorment.

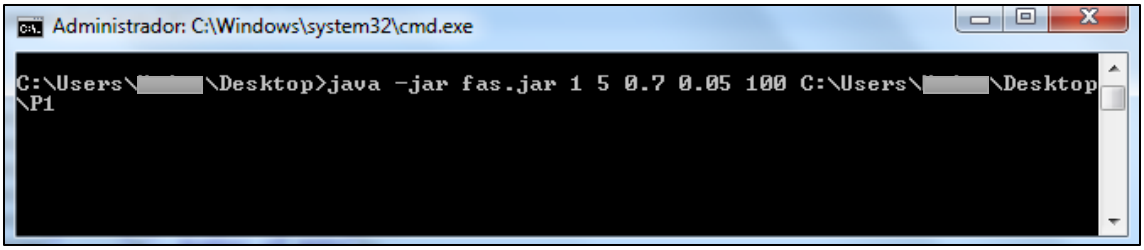


Fig. A.4. Exemple d'execució específica de l'algorisme FAS

Cal tenir en compte els paràmetres que no són modificables en cada execució, i que han estat concretats en punts anteriors del projecte. Aquests són:

- **Quantitat de feromones inicial**: Establert a un valor de 100.
- **Nombre d'iteracions màximes permeses**: Establert a un valor de 2000.
- **Nombre d'execucions per problema**: Establert a 35.
- **Criteri de finalització per estancament**: Establert a 200 iteracions.

Quan es crida el programa amb qualsevol dels dos modes, aquest duu a terme els següents passos:

- Comprovació de la validesa dels paràmetres.
- Creació d'una nova instància ACSP i *parseig* de les dades del problema a una llista de vols continguda en aquesta a través de la funció:

```
public void parseData(String path);
```

- Creació d'una nova instància forgetfulAntSystem o minMaxAntSystem en funció de l'algorisme triat, amb els paràmetres d'entrada o predeterminats especificats.
- Inicialització del comptador de temps.
- Inicialització de l'execució de l'algorisme de colònia de formigues.
- Presentació de resultats

Els resultats obtinguts s'escriuen per consola de forma predeterminada. Malgrat això, existeix la possibilitat de realitzar una redirecció a un arxiu de text per al seu posterior anàlisi. La comanda en aquest cas, hauria d'incloure un operador de redirecció juntament amb el nom de l'arxiu, tal i com es mostra a continuació:

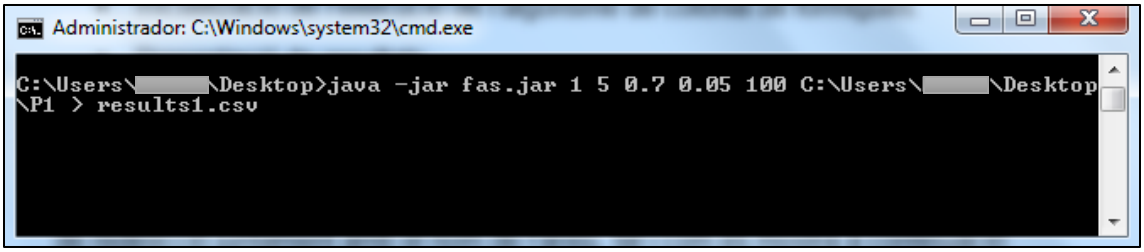


Fig. A.5. Exemple de redirecció a arxiu de text en l'execució específica de l'algorisme FAS

Els càlculs referents a l'F-test i a l'Student t-test per cada agrupació d'execucions de cada problema s'han dut a terme de forma separada utilitzant el programari Microsoft Excel 2013.

A.4 Implementació dels algorismes MMAS i FAS

La implementació dels dos algorismes MMAS i FAS s'ha realitzat a través de les dues classes homònimes mmas i fas. Tot i que és evident que ambdues difereixen en el seu comportament, es realitzarà una presentació conjunta, ja que ambdós algorismes comparteixen els mateixos patrons d'actuació.

L'objectiu d'aquest punt no és detallar explícitament el funcionament de cadascuna de les funcions, ja que aquesta documentació es troba inclosa en els arxius *javadoc*[1] del projecte, sinó oferir una visió general dels processos que segueixen els algorismes en la resolució d'un problema.

En la creació d'una instància de l'algorisme MMAS, per exemple, s'inicialitzen els atributs que es mostren en la figura A.6.

```
    private double alpha;              //Pheromone trail importance
    private double beta;               //Heuristic function importance
    private double exTh;               //Exploration threshold.
    private double evaporationP;       //Evaporation pheromone parameter
    private List<flight> flights;      //Flights in problem
    private double bestProblemCost;    //Best calculated cost
    private int ants;                  //Number of ants
    private int maxIterations;         //Max iterations per ant
    private double minTrail;           //Minimum amount of pheromones
    private double maxTrail;           //Maximum amount of pheromones
    private int actualIteration;       //Actual iteration
    private double[][] infoAnts;       //Array to store results
```

Fig. A.6. Atributs d'inicialització en el constructor de l'algorisme MMAS

La funció *init()* és la que controla el flux d'execució, i funciona seguint els passos que es descriuen a la figura A.7.

[1] Documentació de la API en format HTML.

```
public void init(){

        initTrailAndHeuristic();              //Initialize trail and heuristics matrix
        ....
        ....
        //Initialize ant colony
        ....
        ....

        //While we don't reach maxIterations or we don't reach iterationsBeforeStop, do:
        for (int j=0; j<this.maxIterations; j++){

            //Init all the ants.
            ...
            ...
            updatePheromoneTrail();  //Update pheromone trail with best ant results
            ...
            ...
            //Reboot ants <-> Update iteration
            ...
            ...
            //Check iterations before stop
        }
        //Print results
        calculateBestCost();
    }
```

Fig. A.7. Flux d'execució i funcionament de la funció init() en els algorismes.

En primer lloc s'inicialitzen les matrius que emmagatzemen els nivells de feromones i els valors heurístics per cadascun dels arcs a través de la funció ***initTrailAndHeuristic()***. Els valors dels camins de feromones entre cada parell de vols s'estableix inicialment a ***maxTrail*** (100). Pel que fa als valors heurístics, aquests s'estableixen a $1/\sqrt{Temps\_entre\_vols}$ també per cada parell de vols, amb excepció d'aquells vols en què no existeix connectivitat[1], on aquest valor s'inicialitza a 0.

A continuació s'estableix la colònia de formigues creant totes les instàncies formiga especificades en l'atribut ***ants*** i inicialitzant-les amb els paràmetres establerts en el constructor.

A partir d'aquest punt, s'inicia un cicle d'iteracions restringit per les regles d'acabament imposades en els algorismes: ***maxIterations*** (màxim nombre d'iteracions permeses) i ***iterationsBeforeStop*** (màxim nombre d'iteracions mantenint un resultat pitjor que el millor establert amb anterioritat). En aquest cicle es duen a terme tres accions principals:

- Inicialitzar els procés de cerca de cadascun dels agents formiga.
- Actualitzar els camins de feromones un cop aquests agents han acabat les construccions dels recorreguts individuals a través de la funció

[1] Per exemple si un primer vol (1) aterra en una ciutat d'on no surt un segon vol (2). En aquest cas, no hi haurà connectivitat en el sentit 1->2. És possible, però, que aquesta connectivitat sí que existeixi en el sentit invers 2->1.

updatePheromoneTrail(). Aquesta també s'utilitza per a emmagatzemar els valors obtinguts en la matriu ***infoAnts[ants][iterations]***.

- Actualitzar la matriu de feromones i reinicialitzar cadascun dels agents amb els nous valors.

Finalment, quan s'arriba al final del cicle d'iteracions, s'utilitza la funció ***calculateBestCost()*** per a comprovar els resultats obtinguts per cadascun dels agents en cadascuna de les iteracions. S'obté el millor cost, el millor i el pitjor cost mitjà per cada iteració, el millor i el pitjor cost mitjà per totes les iteracions de cada agent i el nombre d'iteracions comptabilitzades durant l'execució de l'algorisme.

El funcionament de la funció ***updatePheromoneTrail()*** difereix en cadascun dels dos algorismes implementats. En el cas de l'algorisme MMAS, primer es realitza un càlcul del millor recorregut amb la funció ***calculateBestPath()*** i a continuació s'actualitzen tots els camins de feromones a partir de les dades obtingudes. En l'algorisme FAS, en canvi, s'utilitza la funció ***calculateBestPaths()***, que retorna una llista de les formigues amb millors resultats (les que obtenen resultats inferiors a la mitjana). Amb aquesta informació s'accedeix al recorregut dut a terme per cadascun d'aquests agents, i s'actualitzen els camins de feromones utilitzant aquestes dades.

Les classes ***ant*** de cada algorisme difereixen principalment en els mecanismes que utilitzen en la construcció dels *pairings*. Mentre que en l'algorisme MMAS, la funció ***ant.init()*** segueix els passos mostrats en la figura 10 de la pàgina 46, en l'algorisme FAS, aquesta mateixa funció segueix els passos observables en la figura 13 de la pàgina 59.

El .javadoc inclòs amb el projecte ofereix una informació més específica del funcionament de cadascuna de les funcions en aquestes classes.

Apèndix B. Especificació dels problemes de test

B.1 Obtenció d'arxius de test

Per a obtenir els conjunts de test utilitzats en aquest projecte s'ha creat un executable java[1] anomenat ***dataGenerator.jar***. Aquest arxiu és capaç de generar dades de vols aleatoris en format .csv[2]. L'executable espera com a paràmetres el nombre d'aeroports, el nombre de vols a generar, el nombre de dies en què es realitzaran els vols i el nom de l'arxiu de sortida. En la figura B.1. es pot observar un exemple del seu funcionament.

[1] Arxiu dataGenerator.jar
[2] Comma Separated Value (CSV)

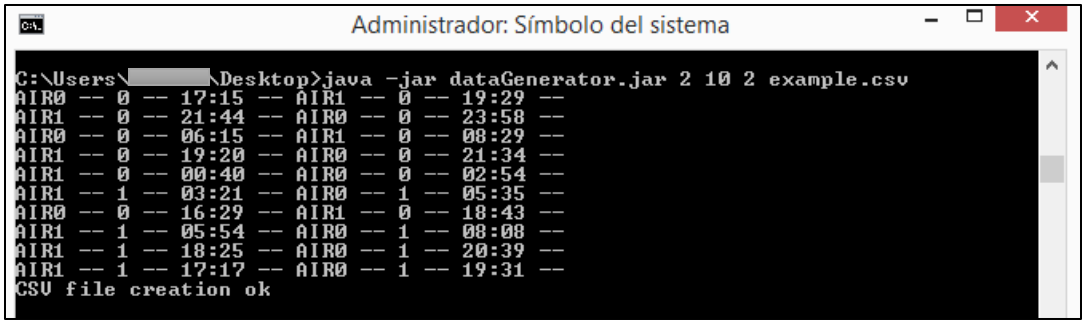


Fig. B.1. Exemple de funcionament del generador d'arxius de test ***DataGenerator.jar***

B.2 Problemes de test

En els següents apartats s'amplia breument la informació sobre els models de test utilitzats per a dur a terme la comparativa entre els algorismes MMAS i FAS.

B.2.1 Problema 1

Aeroports		**AIR0,AIR1**
Nombre i tipus de trajectes	AIR0->AIR1	11
	AIR1->AIR0	19
Total Vols (m)		30
Dies		2

Taula. B.2.1. Detall problema P1

B.2.2 Problema 2

Aeroports		**AIR0,AIR1**
Nombre i tipus de trajectes	AIR0->AIR1	60
	AIR1->AIR0	40
Total Vols (m)		100
Dies		3

Taula. B.2.2. Detall problema P2

B.2.3 Problema 3

Aeroports		**AIR0,AIR1**
Nombre i tipus de trajectes	AIR0->AIR1	100
	AIR1->AIR0	100
Total Vols (m)		200
Dies		5

Taula. B.2.3. Detall problema P3

B.2.4 Problema 4

Aeroports		**AIR0,AIR1,AIR2**
Nombre i tipus de trajectes	AIR0->AIR1	6
	AIR0->AIR2	7
	AIR1->AIR0	3

	AIR1->AIR2	4
	AIR2->AIR0	6
	AIR2->AIR1	4
Total Vols (m)		30
Dies		2

Taula. B.2.4. Detall problema P4

B.2.5 Problema 5

Aeroports		**AIR0,AIR1,AIR2**
Nombre i tipus de trajectes	AIR0->AIR1	15
	AIR0->AIR2	15
	AIR1->AIR0	20
	AIR1->AIR2	15
	AIR2->AIR0	18
	AIR2->AIR1	17
Total Vols (m)		100
Dies		3

Taula. B.2.5. Detall problema P5

B.2.6 Problema 6

Aeroports		**AIR0,AIR1,AIR2**
Nombre i tipus de trajectes	AIR0->AIR1	42
	AIR0->AIR2	46
	AIR1->AIR0	42
	AIR1->AIR2	38
	AIR2->AIR0	40
	AIR2->AIR1	42
Total Vols (m)		250
Dies		5

Taula. B.2.6. Detall problema P6

B.2.7 Problema 7

Aeroports				**AIR0,AIR1,AIR2,AIR3,AIR4**
Nombre i tipus de trajectes	AIR0->AIR1	1	AIR2->AIR3	0
	AIR0->AIR2	2	AIR2->AIR4	3
	AIR0->AIR3	1	AIR3->AIR0	4
	AIR0->AIR4	2	AIR3->AIR1	4
	AIR1->AIR0	3	AIR3->AIR2	3
	AIR1->AIR2	4	AIR3->AIR4	5
	AIR1->AIR3	3	AIR4->AIR0	1
	AIR1->AIR4	0	AIR4->AIR1	5
	AIR2->AIR0	2	AIR4->AIR2	2
	AIR2->AIR1	1	AIR4->AIR3	4

Total Vols (m)	50
Dies	2

Taula. B.2.7. Detall problema P7

B.2.8 Problema 8

Aeroports				**AIR0,AIR1,AIR2,AIR3,AIR4**
Nombre i tipus de trajectes	AIR0->AIR1	5	AIR2->AIR3	6
	AIR0->AIR2	9	AIR2->AIR4	7
	AIR0->AIR3	3	AIR3->AIR0	9
	AIR0->AIR4	7	AIR3->AIR1	13
	AIR1->AIR0	3	AIR3->AIR2	7
	AIR1->AIR2	7	AIR3->AIR4	6
	AIR1->AIR3	5	AIR4->AIR0	7
	AIR1->AIR4	10	AIR4->AIR1	9
	AIR2->AIR0	10	AIR4->AIR2	8
	AIR2->AIR1	9	AIR4->AIR3	10
Total Vols (m)				150
Dies				3

Taula. B.2.8. Detall problema P8

B.2.9 Problema 9

Aeroports				**AIR0,AIR1,AIR2,AIR3,AIR4**
Nombre i tipus de trajectes	AIR0->AIR1	26	AIR2->AIR3	16
	AIR0->AIR2	17	AIR2->AIR4	7
	AIR0->AIR3	13	AIR3->AIR0	17
	AIR0->AIR4	13	AIR3->AIR1	19
	AIR1->AIR0	15	AIR3->AIR2	16
	AIR1->AIR2	11	AIR3->AIR4	12
	AIR1->AIR3	24	AIR4->AIR0	17
	AIR1->AIR4	12	AIR4->AIR1	12
	AIR2->AIR0	7	AIR4->AIR2	23
	AIR2->AIR1	12	AIR4->AIR3	11
Total Vols (m)				300
Dies				5

Taula. B.2.9. Detall problema P9

B.2.10 Problema 10

Aeroports										**AIR0,AIR1,AIR2,AIR3,AIR4,AIR5,AIR6,AIR7,AIR8,AIR9**
Nombre i tipus de trajectes										
		AIR0	AIR1	AIR2	AIR3	AIR4	AIR5	AIR6	AIR7	AIR8	AIR9
	AIR0 ->	0	1	2	4	0	0	4	1	0	0
	AIR1 ->	0	0	3	1	0	0	1	1	0	1
	AIR2 ->	0	1	0	3	2	0	0	0	2	2
	AIR3 ->	2	1	1	0	2	0	0	4	0	1
	AIR4 ->	2	0	0	2	0	2	0	1	3	0

	AIR5 ->	1	1	3	0	2	0	1	2	1	1
	AIR6 ->	1	0	3	2	1	1	0	2	4	1
	AIR7 ->	1	1	1	0	0	1	1	0	0	1
	AIR8 ->	1	0	0	0	1	3	0	2	0	0
	AIR9 ->	0	1	2	2	0	2	2	1	0	0
Total Vols (m)											100
Dies											2

Taula. B.2.10. Detall problema P10

B.2.11 Problema 11

Aeroports		**AIR0,AIR1,AIR2,AIR3,AIR4,AIR5,AIR6,AIR7,AIR8,AIR9**									
Nombre i tipus de trajectes											
		AIR0	AIR1	AIR2	AIR3	AIR4	AIR5	AIR6	AIR7	AIR8	AIR9
	AIR0 ->	0	6	1	0	4	6	5	3	1	3
	AIR1 ->	5	0	3	2	2	2	5	1	7	2
	AIR2 ->	2	2	0	3	3	4	4	5	7	4
	AIR3 ->	2	1	3	0	3	4	2	3	4	3
	AIR4 ->	3	2	3	3	0	4	2	3	2	5
	AIR5 ->	3	5	4	1	1	0	7	3	1	2
	AIR6 ->	3	5	1	5	0	3	0	6	5	3
	AIR7 ->	5	5	5	4	5	1	4	0	7	5
	AIR8 ->	3	2	2	2	2	7	5	6	0	3
	AIR9 ->	3	1	5	0	3	2	4	4	3	0
Total Vols (m)											300
Dies											3

Taula. B.2.11. Detall problema P11

B.2.12 Problema 12

Aeroports		**AIR0,AIR1,AIR2,AIR3,AIR4,AIR5,AIR6,AIR7,AIR8,AIR9**									
Nombre i tipus de trajectes		**Destinació**									
		AIR0	AIR1	AIR2	AIR3	AIR4	AIR5	AIR6	AIR7	AIR8	AIR9
	AIR0 ->	0	7	1	5	4	6	8	6	6	6
	AIR1 ->	7	0	5	7	4	9	4	5	4	6
	AIR2 ->	7	4	0	3	4	3	6	4	6	1
	AIR3 ->	4	6	11	0	6	7	5	7	4	4
	AIR4 ->	5	7	7	9	0	7	4	8	5	4
	AIR5 ->	4	8	6	8	3	0	7	4	4	3
	AIR6 ->	5	5	3	11	8	10	0	4	5	3
	AIR7 ->	5	7	6	5	4	4	4	0	6	4
	AIR8 ->	4	9	3	9	10	6	7	7	0	4
	AIR9 ->	5	5	7	5	5	4	5	7	4	0
Total Vols (m)											500
Dies											5

Taula. B.2.12. Detall problema P12