Inteligencia Artificial 17(54), 1-16
doi: 10.4114/ia.v17i54.1011

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Conformación de equipos de proyectos de software aplicando algoritmos metaheurísticos de trayectoria multiobjetivo

MSc. Ana Lilian Infante Abreu[1], Dra. Margarita André Ampuero[2], Dr. Alejandro Rosete Suárez[3], Lalchandra Rampersaud[4]
[123]Facultad de Ingeniería Informática, Instituto Superior Politécnico "José Antonio Echeverría", 114 #11901 e/ Ciclovía y Rotonda, Marianao, La Habana, Cuba
[4]Ministry of Education, 26 Brickdam Stabroek, Georgetown, Guyana
[1]ainfante@ceis.cujae.edu.cu, [2]mayi@ceis.cujae.edu.cu, [3]rosete@ceis.cujae.edu.cu, [4]lalchandrar@gmail.com

Abstract The inadequate formation of software project teams is a problem that affects the software industry worldwide. The team formation process becomes very complex when taking into account several factors such as, assigning team roles to people with the right competences, consideration of incompatibilities between the members and workload, among others. This situation becomes more complex in medium and large organizations due to the large number of combinations of possible assignments. This stage becomes virtually impossible to address efficiently without the aid of mathematical models. These models represent the problem to be solved as objectively as possible.
This paper takes as its antecedent a model that includes both individual and team factors which proposes to maximize the competence of workers, minimize incompatibilities between team members and balance the workload. An expanded version of the model includes the minimization of cost associated to the development software remotely. The model used corresponds to a multiobjective combinatorial optimization problem. To solve this problem multiobjective variants of metaheuristic algorithms such as Tabu Search, Simulated Annealing and Hill Climbing were used.
The experimental study conducted has led to the identification of various algorithms which produced the best results for this problem. The algorithms identified are: Multi-case Multiobjective Simulated Annealing, and multiobjective variants of Hill Climbing Algorithms such as Multiobjective Stochastic Hill Climbing, Multiobjective Hill Climbing by Greater Distance and Multiobjective Hill Climbing with Restart.

Resumen La inadecuada conformación de equipos de proyecto de software es un problema que afecta a la industria de software a nivel mundial. Este proceso resulta complejo, teniendo en cuenta que debe considerar varios factores, como son, asignar a los roles del equipo las personas con las competencias apropiadas, considerar las incompatibilidades entre los miembros y la carga de trabajo, entre otros. Esta situación se torna más compleja en organizaciones medianas y grandes, debido a la gran cantidad de combinaciones de asignaciones posibles, por lo que esta etapa es prácticamente imposible de abordar de manera eficiente, sin la ayuda de modelos matemáticos que representen el problema a resolver lo más objetivamente posible.
Este trabajo toma como antecedente un modelo que incluye tanto factores individuales como factores de equipo y plantea: maximizar las competencias de los trabajadores, minimizar las incompatibilidades entre los miembros del equipo y balancear la carga de trabajo. Incluye además, en una versión ampliada del modelo, minimizar el costo de desarrollar software a distancia. El modelo citado responde a un problema de optimización combinatorio multiobjetivo, por lo que para su solución se utilizaron algunas variantes multiobjetivo de los algoritmos metaheurísticos: Búsqueda Tabú, Recocido Simulado y Escalador de Colinas.
El estudio experimental realizado ha llevado a identificar que las variantes multiobjetivo del Escalador de Colinas: Escalador de Colinas Estocático Multiobjetivo, Escalador de Colinas Multiobjetivo por mayor distancia y

ISSN: 1988-3064(on-line)

Escalador de Colinas Multiobjetivo con Reinicio, así como el algoritmo Recocido Simulado Multiobjetivo Multicaso son los que mejores resultados obtienen en este problema.

Keywords: single-solution based metaheuristics, assignment problem, optimization problem, multi-objective optimization problem.
Palabras claves: metaheurísticas de trayectoria, problema de asignación, problema de optimización, problema de optimización multiobjetivo.

1 Introducción

Resultan numerosos los problemas de la vida cotidiana en los que se tienen que satisfacer varios objetivos, en muchas ocasiones en conflicto. Estos son conocidos como problemas multiobjetivo [1].

El reto de conformar equipos capaces de desarrollar proyectos de software exitosos constituye un problema multiobjetivo, en tanto se requiere tomar en cuenta varios criterios como son, asignar a los roles del equipo las personas con las competencias apropiadas, considerar las incompatibilidades entre los miembros y la carga de trabajo, entre otros factores [2].

El proceso de formación de equipos se torna complejo en medianas y grandes empresas, debido a la gran cantidad de combinaciones de asignaciones posibles, en dimensiones relativamente significativas de roles a cubrir y empleados disponibles. Esto hace que esta etapa sea prácticamente imposible de abordar de manera eficiente, sin la ayuda de sistemas informatizados de soporte a la decisión que se basen en modelos matemáticos y que representen el problema a resolver lo más objetivamente posible.

En la bibliografía consultada se citan varios modelos que abordan la asignación de personal a proyectos de software [3-7]. Entre estos modelos, el propuesto en [2, 4, 8] es el que se considera más completo debido a que no solo considera factores que favorecen la asignación individual a los roles establecidos en un equipo de desarrollo de software sino que toma en cuenta factores que contribuyen a la formación del equipo como un todo.

El modelo propuesto en [2, 4, 8] responde a un problema de optimización combinatorio multiobjetivo, en tanto, la asignación de personas a equipos de proyecto de software consiste en asignar n trabajadores (según se requiere) a m roles necesarios para llevar a cabo un proyecto, considerando tres objetivos: maximizar las competencias de los trabajadores en el rol asignado, minimizar las incompatibilidades entre los miembros del equipo de desarrollo y balancear la carga de trabajo. Además, se propone una versión que incluye un cuarto objetivo asociado a minimizar el costo de desarrollar software a distancia para el caso de organizaciones que así lo requieran. Asimismo, el modelo incluye doce tipos de restricciones que limitan la asignación de personas a roles establecidos como incompatibles, regulan la cantidad máxima de roles a desempeñar por un individuo y la carga de trabajo máxima que puede tener un individuo, entre otros.

Dado la factibilidad de utilizar algoritmos metaheurísticos en la solución de problemas combinatorios multiobjetivo [1], para el modelo propuesto en [2, 4, 8] se utilizaron algoritmos de trayectoria como la Búsqueda Tabú [9], el Recocido Simulado [10], el Escalador de Colinas [11] de Mejor Ascenso con Reinicio, e híbridos del Procedimiento de búsqueda aleatorio, adaptativo y avaricioso (GRASP, por sus siglas en inglés) [12]: GRASP con Búsqueda Tabú, GRASP con Recocido Simulado y GRASP con Escalador de Colinas de Mejor Ascenso, obteniéndose buenas soluciones (soluciones factibles) en tiempos aceptables (no mayor de 3 minutos). Sin embargo, solo se utilizó el método de solución de factores ponderados [1], el que construye una única función objetivo a partir de la suma de las funciones objetivos iniciales ponderadas según un valor de peso. Este método supone que el responsable de conformar el equipo debe asignar pesos a los objetivos al inicio del proceso, lo que en muchas ocasiones resulta difícil para este. Por otra parte, para diferentes combinaciones de pesos pueden obtenerse iguales soluciones.

En este trabajo se propone el uso del método de solución multiobjetivo puro, que propone tratar el problema como multiobjetivo y evaluar las funciones objetivos por separado, haciendo uso del concepto de óptimo de Pareto.

En [4] se realizaron pruebas experimentales para validar la aplicabilidad del modelo propuesto en escenarios de organizaciones medianas (de 51 a 250 trabajadores) y grandes (250-999 trabajadores), utilizando en todos los casos algoritmos de trayectoria. Sin embargo, aún para el método de factores ponderados utilizado, no se evaluó la aplicación del modelo para organizaciones significativamente grandes (1000 o más trabajadores).

Por otra parte, en la bibliografía consultada se citan varios trabajos que utilizan algoritmos poblacionales multiobjetivo en la solución de problemas de asignación [1, 13-15], específicamente algoritmos genéticos [15-20]. De igual forma, existen diversas fuentes que refieren la capacidad de los algoritmos de trayectoria basados en óptimo de Pareto para resolver problemas de optimización multiobjetivo [21-25] y específicamente algunos tipos

de problemas de asignación [26-32]. Sin embargo, en estos trabajos no se evalúa el desempeño de diferentes algoritmos multiobjetivo aplicados a problemas de asignación.

Tomando en consideración lo planteado en el teorema No Free Lunch [33], de que no existe un método que sea absolutamente mejor que otro cuando se comparan en todas las funciones posibles, aún fijando una función a optimizar para un tipo de problema, un método puede ser más eficiente que otro dependiendo de la dimensionalidad de las instancias a resolver y del recurso a optimizar (calidad de la solución o tiempo de ejecución).

Los algoritmos de trayectoria tienen menor complejidad en los cruzamientos y necesitan menos memoria que los poblacionales, considerándose por tanto algoritmos más sencillos. Es por esto que se decide comenzar el estudio del comportamiento de los algoritmos de trayectoria para darle solución al modelo planteado.

Este trabajo tiene como objetivo presentar los resultados obtenidos al aplicar algoritmos de trayectoria multiobjetivo al modelo de conformación de equipos de proyectos de software para diferentes tamaños del problema (organizaciones medianas, grandes y especialmente grandes).

2 Modelo de conformación de equipos de proyectos de software

El problema de conformación de equipos de proyecto de software plantea la asignación de n personas a m roles, donde una persona puede desempeñar más de un rol y un rol puede ser desempeñado por más de una persona. El modelo propuesto en [2, 4, 34] plantea:

Sean:

m: cantidad de roles necesarios para desarrollar un proyecto.

n: cantidad de empleados disponibles.

MaxR: Cantidad máxima de roles que un empleado puede desempeñar en un proyecto dado.

R: cantidad de conjuntos de roles incompatibles contemplados

IR_r: Conjunto de roles incompatibles, r = 1,…,R.

C_j: Conjunto de competencias requeridas para desempeñar el rol j.

NE_j: Cantidad de empleados requeridos para desempeñar el rol j; j = 1,…,m.

c_{ij}: Capacidad del empleado i para desempeñar el rol j; i = 1..n, j = 1..m.

nc_{ic}: Nivel del empleado i en la competencia c.

s_{hi} : Incompatibilidad entre los empleados h e i; h, i = 1..n. (El valor de este coeficiente es 1 si las incompatibilidades son recíprocas entre los empleados h e i y 0 en caso contrario)

g_{ij} : Carga de trabajo del empleado i en el rol j según los proyectos a los que está ya asignado; i = 1..n; j = 1,…,m

b_j : Carga de trabajo que implica asumir el rol j en el proyecto de análisis; j = 1..m

MaxCT: Máxima carga de trabajo para un empleado.

l_{ij} : Costo del empleado i según la lejanía que tenga del proyecto y el rol j que va a desempeñar; i = 1..n, j = 1..m

mnc_{cj}: Nivel mínimo requerido de la competencia c para desempeñar el rol j, j = 1,…,m.

ME: Promedio de carga de trabajo incluyendo la carga de trabajo de los proyectos actuales y la carga de trabajo que genera desempeñar el rol en el proyecto bajo análisis.

$$ME = \frac{\sum_{j=1}^{m}[(\sum_{i=1}^{n} g_{ij}) + b_j]}{n}$$

y las variables:

x_{ij} = 1 si el empleado i es asignado al rol j y 0 en caso contrario, i = 1,…,n; j=1,…,m

u_i = 1 si el empleado i es asignado al menos a un rol y 0 en caso contrario; i = 1,…,n

Las funciones objetivo a optimizar son:

- Maximizar las competencias de los trabajadores en el rol o los roles asignados.
  $\max \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$
  Para determinar la competencia neta del empleado i para desempeñar el rol j (c_{ij}), se consideran los niveles de competencia (nc_{ic}) que posee el trabajador en las competencias requeridas para desempeñar el rol j (C_j), así como los pesos que tienen cada competencia c en cada rol j, ponderados según su importancia.
- Minimizar las incompatibilidades entre los miembros de un equipo de proyecto.
  $\min \sum_{h=1}^{n-1} \sum_{i>h}^{n} s_{hi} u_h u_i$
- Balancear la carga del personal del equipo.
  $min \sum_{i=1}^{n} \left[\left(\sum_{j=1}^{m} g_{ij} + \sum_{j=1}^{m} b_j x_{ij} \right) - ME \right]^2$

Cada término de la suma en i representa la diferencia entre la carga de trabajo total de un individuo (incluyendo los proyectos en los que ya está asignado y el proyecto bajo análisis), y el promedio de carga de trabajo de todos los miembros del equipo (considerando los proyectos en los que están asignados y el proyecto bajo análisis).

- Minimizar el costo de trabajar a distancia. Este objetivo solo es aplicable a organizaciones que enfrentan esta variante de desarrollo.
  $\min \sum_{i=1}^{n} \sum_{j=1}^{m} l_{ij} x_{ij}$
  El costo de que un empleado trabaje en el proyecto a distancia tiene en cuenta la ubicación del empleado con respecto a la ubicación del proyecto y el impacto que tiene desarrollar un rol a distancia.

El modelo toma en cuenta los siguientes tipos de restricciones:

- Los roles deben ser cubiertos en función de la cantidad necesaria de personas a desempeñarlo.
  $\sum_{i=1}^{n} x_{ij} = NE_j$, donde $NE_j \in$ N
- Una persona no puede desempeñar al mismo tiempo roles que se consideren incompatibles entre sí.
  $\sum_{j \epsilon Jr} x_{ij} \leq 1$, siendo r = 1,…,R
- Restringir el número máximo de roles que puede desempeñar cualquier trabajador en el proyecto a una cantidad fijada por el usuario (MaxR).
  $\sum_{j=1}^{m} x_{ij} \leq MaxR$, $MaxR \in$ N
- Para que una persona desempeñe un rol debe cumplir los requisitos mínimos de nivel de competencia para desempeñar dicho rol.
  $u_i(mnc_{cj} - nc_{ic}) \leq 0, \forall c \in C_j, \forall i = 1, \dots, n \quad \forall j = 1, \dots, m$
- La carga de trabajo total asignada a un empleado no debe ser mayor que un valor máximo.
  $g_i + \sum_{j=1}^{m} b_j x_{ij} \leq MaxCT$, i = 1,..,n,
- Garantizar que la variable u_i tome valor 1 ó 0.
  $(1 - u_i) \sum_{j=1}^{m} x_{ij} = 0$,
  $(1 - u_i) + \sum_{j=1}^{m} x_{ij} > 0$, i = 1,…,n

Otro conjunto de restricciones refleja la relación que existe entre los roles que Belbin[1] define que deben estar presentes en un equipo de trabajo, los tipos psicológicos de Myers Briggs[2] y los roles a desempeñar en un equipo de proyecto de software [2, 4, 34]. Siendo D = (d_{ij}) una matriz de valores 0 ó 1, tomando el valor 1 si el individuo i tiene el rol de Belbin j como preferido de acuerdo a su personalidad, $1 \leq i \leq n$; $1 \leq j \leq 9$ y B = (b_{ij}) una matriz de valores 0 ó 1, tomando el valor 1 si el individuo i tiene preferencia por la primera letra de cada dimensión del MBTI (E/I, S/N, T/F, J/P) y 0 por la segunda, $1 \leq i \leq n$; $1 \leq j \leq 4$:

- Un conjunto de restricciones garantizan que en el equipo de desarrollo se representen las tres categorías de roles propuestas por Belbin (roles de acción, roles mentales y roles sociales).
  $\sum_{k=1}^{3}(\sum_{i=1}^{n} d_{ik} u_i) > 0$, $\sum_{k=4}^{6}(\sum_{i=1}^{n} d_{ik} u_i) > 0$, $\sum_{k=7}^{9}(\sum_{i=1}^{n} d_{ik} u_i) > 0$
- En el equipo de trabajo la preferencia por desempeñar roles de acción debe sobrepasar la preferencia por desempeñar los roles mentales.
  $$\sum_{k=1}^{3}\left(\sum_{i=1}^{n} d_{ik} u_i\right) > \sum_{k=4}^{6}\left(\sum_{i=1}^{n} d_{ik} u_i\right)$$
- En el equipo de trabajo la preferencia por desempeñar roles mentales deben sobrepasar la preferencia por desempeñar los roles sociales.
  $$\sum_{k=4}^{6}\left(\sum_{i=1}^{n} d_{ik} u_i\right) > \sum_{k=7}^{9}\left(\sum_{i=1}^{n} d_{ik} u_i\right)$$
- La persona que desarrolla el rol de Jefe de Proyecto debe tener como preferido los roles de Belbin: Impulsor o Coordinador.
  $x_{i1} \leq d_{i1} + d_{i7}$, asumiendo que en una matriz D la preferencia o no por desempeñar los roles Impulsor y Coordinador se registran en las columnas 1 y 7 respectivamente.
- En el equipo al menos una persona debe tener como preferido el rol mental Cerebro.

[1]Meredith Belbin es el creador del test que lleva su apellido y que identifica la preferencia de las personas por desempeñar algunos de los nueve roles que define están presentes en un equipo. La metodología establece que en un equipo debe haber presencia de las tres categorías de roles (mentales, sociales y de acción) donde deben predominar los roles de acción y no debe existir una alta presencia de roles sociales ni mentales.

[2]Test que mide cuatro dimensiones diferentes de las preferencias humanas: Extroversión (E)-Introversión (I), Intuición (N)-Sentidos (S), Emoción (F)-Pensamiento (T), y Juicio (J)-Percepción (P). A partir de los valores de cada dimensión se identifica el tipo psicológico de la persona entre los 16 tipos posibles.

$\sum_{i=1}^{n} d_{i4} u_i \geq 1$, asumiendo que en la columna 4 de la matriz D se registra la preferencia o no por el rol Cerebro.

- La persona que desarrolla el rol Jefe de Proyecto debe ser extrovertida y planificada (subtipo E_ _J) según el test de Myers-Briggs.
  $x_{i1} \leq \frac{(b_{i1}+b_{i4})}{2}$, asumiendo que en una matriz B, la dimensión E/I (Extrovertido/Introvertido) se registra en la columna 1 y la J/P (Juicio/Percepción) en la columna 4.

3 Algoritmos metaheurísticos de trayectoria multiobjetivo

Para dar solución a un problema de optimización multiobjetivo es necesario definir las técnicas y algoritmos a utilizar. Existen numerosas técnicas de solución para problemas multiobjetivo, un conjunto de las cuáles se pueden encontrar en [1]. Algunas de las técnicas más simples proponen transformar el problema multiobjetivo en un problema escalar, ponderando las funciones objetivo, como es el caso del método combinación lineal de pesos (o método de factores ponderados). Otras proponen, tratar el problema como monobjetivo seleccionando solo una de las funciones objetivos y tratando las restantes como restricciones o considerar varios objetivos simultáneamente.

Este trabajo se centrará en esta última técnica conocida como multiobjetivo puro, la que propone obtener el conjunto de soluciones óptimas de Pareto [1].

A continuación se definen algunos conceptos necesarios, asumiendo un problema de minimización [1]:

- Óptimo de Pareto: Una solución x ϵ Ω es llamada óptimo de Pareto con respecto a Ω si y solo si, no existe x´ ϵ Ω para el cual $v = F(x') = \{f_1(x'), \ldots, f_2(x')\}$ domina a $u = F(x) = \{f_1(x), \ldots, f_2(x)\}$. Es decir x´ es un óptimo de Pareto si no existe ningún vector factible x que disminuya algún criterio sin causar un aumento simultáneo en al menos uno de los otros criterios.
- Dominancia de Pareto: Un vector u = (u1,…,uk) se dice que domina a otro vector v = (v1,…,vk) si y solo si u es parcialmente menor que v (denotado por $u \preccurlyeq v$). En otras palabras, x domina a y si x es mejor que y en al menos una de las funciones objetivos y no es peor en ninguna de las restantes.
- Conjunto óptimo de Pareto: Para un problema de optimización multiobjetivo dado, F(x), el conjunto óptimo de Pareto, P*, es definido como $P^* = x \in \Omega \mid \nexists x' \epsilon \Omega\, F^*(x') \preccurlyeq F(x)$. El conjunto óptimo de Pareto está compuesto por aquellos elementos x que pertenecen a Ω, tal que no existe ningún x' perteneciente a Ω para el cuál x' domina a x.
- Frente de Pareto: Para un problema de optimización multiobjetivo dado, F(x), y el conjunto óptimo de Pareto, P*, el frente de Pareto PF* es definido como $PF^* = \{u = F(x) | x \in P^*\}$.

La técnica multiobjetivo puro resulta conveniente en tanto no necesita que se asigne pesos o prioridades a los objetivos al inicio del proceso, a diferencia de otras técnicas como el método combinación lineal de pesos o el método lexicográfico. Estas últimas requieren que se definan pesos o prioridades al inicio del proceso en muchas ocasiones sin fundamentación para esto y obtienen como resultado de la búsqueda una única solución. La técnica multiobjetivo puro, al tratar los objetivos por separado refleja perfectamente la realidad multiobjetivo del problema y obtiene como resultado de la búsqueda un conjunto de soluciones.

En la bibliografía consultada se citan diferentes variantes de algoritmos metaheurísticos de trayectoria que proponen encontrar el conjunto de soluciones no dominadas. En las Tablas 1 y 2 se muestran las principales características de algunas de estas variantes. Los elementos considerados en las tablas incluyen:

1 – Algoritmo básico
2 - ¿Acepta soluciones peores?
3 – ¿Reinicia la búsqueda?
4- ¿Mantiene una lista de soluciones no dominadas?
5- ¿Necesita parámetros?
6- ¿Toma decisiones aleatorias?
7- ¿Ha sido aplicada a problemas de asignación?
8- Diferencia fundamental con el resto de los algoritmos

Todos los algoritmos estudiados utilizan una lista para almacenar las soluciones no dominadas encontradas durante la búsqueda.

En la selección de los algoritmos a implementar se tomaron en cuenta dos criterios:

- Incorporar las variantes utilizadas en [2] en la solución del modelo planteado, o sea, Escalador de Colinas, Búsqueda Tabú y Recocido Simulado, dado que su empleo en la solución del modelo mostró buenos resultados.
- Tomar en cuenta las variantes de algoritmos metaheurísticos de trayectoria multiobjetivo empleadas en la literatura para dar solución a problemas de asignación (ver Tabla 1 y 2).

Determinándose como algoritmos a implementar los siguientes:

- Variantes del Escalador de Colinas [23]: Escalador de Colinas Estocástico Multiobjetivo (ECEMO), Escalador de Colinas Multiobjetivo con Reinicio (ECEMO-R) y Escalador de Colinas Multiobjetivo por mayor distancia (ECEMO-Dist).
- Variantes de Recocido Simulado: Recocido Simulado Multiobjetivo de Ulungu y Teghem [32, 35] (UMOSA) y Recocido Simulado Multiobjetivo Multicaso (MC-MOSA) [36].
- Variantes de Búsqueda Tabú: Búsqueda Tabú Multiobjetivo (BT Multiobjetivo) [37].

4 Formulación del problema

La representación de una solución en el problema planteado se muestra en la Figura 1. Dado que el problema consiste en asignar trabajadores a los roles definidos en un equipo de proyecto determinado y que se puede necesitar que se asignen varios trabajadores a un mismo rol, se define como solución, una lista de elementos, donde cada elemento está compuesto por un rol y la lista de trabajadores asignados a ese rol.

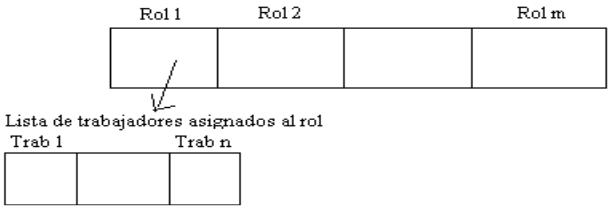

Figura 1. Representación de una solución

El espacio de soluciones está determinado por la instancia del problema que se analiza. Para una instancia de 60 trabajadores y 6 roles, suponiendo que un trabajador solo puede desempeñar un rol y que el rol Jefe de Proyecto se asigna al inicio y permanece fijo durante la asignación, el espacio de soluciones es 6 x 10^8 (según teorema de la r-permutación [38]).

En cuanto al tratamiento de las restricciones todos los algoritmos implementados consideran como soluciones no factibles aquellas que no cumplen con las restricciones del problema, y por tanto no las toma en cuenta durante la búsqueda de una solución. Es decir, en el proceso de búsqueda no se tienen en cuenta soluciones no factibles, por lo que si al generar un vecino de la solución actual resulta que es una solución no factible, esta no es tomada en cuenta y se busca otro vecino.

Los operadores utilizados son sustitución y permutación. El operador permutación escoge dos roles de la solución, selecciona un trabajador de cada rol y los intercambia, tal como se ilustra en la Figura 2 (izquierda). El operador sustitución, escoge uno de los roles de la solución y sustituye uno de los trabajadores que juegan ese rol por otro del espacio de soluciones tal como se muestra en la Figura 2 (derecha).

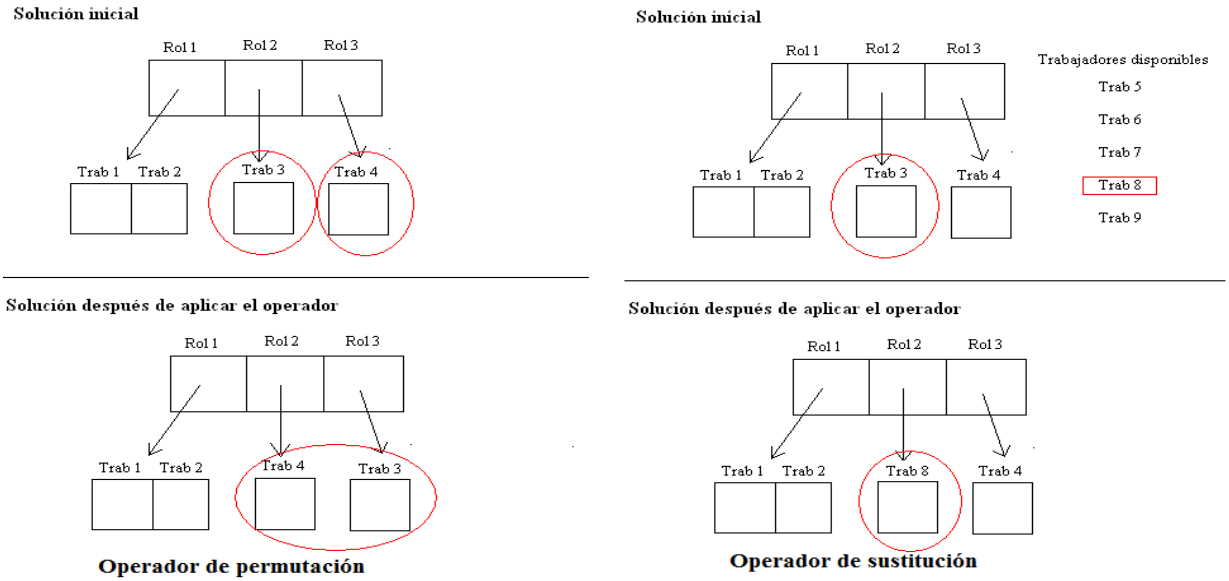

Figura 2. Operadores de permutación y sustitución

Tabla 1: Comparación entre algoritmos de trayectoria multiobjetivo

Variante multiobjetivo	Fuentes de referencia	1	2	3	4	5	6	7	8
EC Estocástico Multiobjetivo	Díaz 2001	**Escalador de Colinas**	No	No	Si	No	Si	No	Mantiene una lista de soluciones donde se almacenan las soluciones no dominadas encontradas durante la búsqueda.
EC Multiobjetivo con Reinicio	Díaz 2001		No	Si	Si	No	Si	No	El algoritmo reinicia la búsqueda si la solución candidata no es aceptada para sustituir a la actual luego de comprobar que se han generado todos los vecinos posibles de la solución actual.
EC Estocástico Multiobjetivo por mayor distancia	Díaz 2001		No	Si	Si	No	Si	No	El comportamiento es similar al anterior. Además calcula la distancia entre las soluciones de la lista de soluciones no dominadas y utiliza la solución de mayor distancia como nueva solución.
BT Multiobjetivo de Baykasoglu	Baykasoglu , Ozbaku et al. 2002	**Búsqueda Tabú**	Si	Si	Si	Si, los típicos del algoritmo	No	(Problema de asignación de trabajos a máquinas)	Utiliza una lista de soluciones que son no dominadas localmente en algún paso del algoritmo, la que permite diversificar la búsqueda.
BT Multiobjetivo de Ho	Ho, Yang et al. 2002		Si	No	Si	Si, los típicos del algoritmo	No	No	Utiliza funciones de escala para evaluar los objetivos. Cataloga las soluciones vecinas de la solución actual en función de un ranking que está dado por la cantidad de soluciones que dominan a la solución vecina.
BT Multiobjetivo de Balicki	Balicki 2007		Si	No	Si	Si, los típicos del algoritmo	No	No	Utiliza el procedimiento de ranking.
GRASP Multiobjetivo	Reynolds, Corne et al. 2009	**GRASP**	Si	Si	Si	Si, los típicos del algoritmo	Si	No	Mantiene una lista de soluciones donde se almacenan las soluciones no dominadas encontradas durante la búsqueda. Construye una lista de soluciones no dominadas y luego aplica un algoritmo de búsqueda local.
Recocido Simulado de Serafini	Serafini 1994	**Recocido Simulado**	Si, con determinada probabilidad	No	Si	Si, los típicos del algoritmo.	Si	No	Utiliza un vector de peso en el cálculo de la probabilidad de aceptación. Utiliza un método similar al lexicográfico.
UMOSA	Coello, Veldhuizen et al. 2007			Si	Si	Si, los pesos para el cálculo de la probabilidad de aceptación.	Si	(Problema de la mochila biobjetivo)	Utiliza una estrategia llamada criterio de escala donde la probabilidad de aceptación de una nueva solución depende de la distancia entre la solución actual y la nueva solución.

Tabla 2. Comparación entre algoritmos de trayectoria multiobjetivo (continuación)

Variante multiobjetivo	Fuentes de referencia	1	2	3	4	5	6	7	8
SMOSA	Suman and Kumar 2006	**Recocido Simulado**	Si, con determinada probabilidad	Si	Si	Si, la temperatura inicial de cada objetivo.	Si	No	La probabilidad de aceptación de una nueva solución tiene en cuenta múltiples temperaturas (una para cada objetivo). No utiliza vectores de peso en este cálculo.
WMOSA	Suman and Kumar 2006			Si	Si	Si, los típicos del algoritmo	Si	No	Utiliza un vector de peso en el cálculo de la probabilidad de aceptación que depende del número de restricciones violadas por el vector solución y el vector función objetivo.
PDMOSA	Suman and Kumar 2006			Si	Si	Si, los típicos del algoritmo	Si	No	No usa el valor de la función objetivo en el criterio de aceptación, sino un valor de aptitud que se obtiene a partir del número de soluciones que dominan a la solución actual en el conjunto optimal de Pareto.
Recocido Simulado Multiobjetivo con Búsqueda de Trayectoria Aleatoria	Baesler, Moraga et al. 2008			No	Si	Si, la temperatura inicial de cada objetivo y los pesos w1 y w2 que representan la importancia entre la memoria a largo y corto plazo.	Si	(Problema de programación de la producción de máquinas paralelas)	Selecciona en cada iteración un solo objetivo que será encargado de guiar la búsqueda en esa iteración del algoritmo. Incorpora memoria a corto y largo plazo, que permite seleccionar la dirección en la cual el algoritmo deberá concentrarse. Utiliza una temperatura para cada objetivo.
Recocido Simulado de Pareto basado en restricciones	Hamm, Beißert et al. 2009			No	Si	Si, los típicos del algoritmo	Si	(Problema de asignación de tareas a recursos)	La generación de una solución vecina utiliza la simulación basada en restricciones. Durante la simulación se modifican varios pasos teniendo en cuenta las restricciones definidas. La probabilidad de aceptar una nueva solución se basa en la dominancia entre la solución actual y candidata.
MC-MOSA	Haidine and Lehnert 2008			No	Si	Si, los típicos del algoritmo	Si	(Problema de la mochila)	Subdivide en 3 subcasos el caso en que la solución actual y la candidata son indiferentes entre sí para lograr una mejor exploración del espacio de búsqueda.
AMOSA	Bandyopadhyay, Saha et al. 2008			No	Si	Si, el tamaño máximo del archivo HL y el archivo SL	Si	No	El tamaño de la lista de soluciones no dominadas es limitado, y cuando se pasa de ese tamaño se aplican técnicas de agrupamiento para mantenerlo en el tamaño predefinido.
RS basado en Pareto	Burke 2009			No	Si	Si, los típicos del algoritmo	Si	(Problema de planificación de enfermeras)	Utiliza dos funciones para el cálculo de la probabilidad de aceptación: una función de evaluación de suma de pesos y una función de evaluación basada en la dominancia.

5 Métricas

Para comparar el rendimiento de los diferentes algoritmos multiobjetivo, teniendo en cuenta el hecho de que su resultado no es un único vector escalar sino una colección de vectores formados por un conjunto no dominado, existen varias métricas. Estas persiguen medir [39]:

- La distancia del frente de Pareto producido por el algoritmo diseñado con respecto al frente verdadero (suponiendo que el frente de Pareto verdadero es conocido).
- La distribución de soluciones obtenidas, de manera que se pueda tener una distribución de vectores lo más uniforme posible.
- La cantidad de elementos del conjunto de óptimos de Pareto generados.

Ninguna de las métricas que se analizan a continuación, captura en un solo valor numérico los tres elementos mencionados, ya que estos se refieren a aspectos de desempeño diferentes. Por ello, es recomendable usar diferentes métricas para evaluar los distintos aspectos de desempeño de un algoritmo [1].

Algunas de las métricas revisadas en la bibliografía se describen en [40]. Las empleadas en este trabajo incluyen:

- Tasa de error [41]: Indica el porciento de soluciones que no son miembros del frente de Pareto verdadero, siendo n el número de vectores en el frente de Pareto actual; $e_i = 0$ si el vector i es un miembro del frente de Pareto verdadero y $e_i = 1$ de lo contrario:

$$ER = \frac{\sum_{i=1}^{n} e_i}{n}$$

  El comportamiento ideal del algoritmo es que la tasa de error sea 0, puesto que indica que todos los vectores generados por el algoritmo pertenecen al frente de Pareto verdadero. Sin embargo, esta métrica requiere conocer los elementos del frente de Pareto verdadero, lo cual puede resultar difícil de obtener en problemas reales.

- Distancia generacional [41]: indica qué tan lejos están los elementos del frente de Pareto actual respecto al frente de Pareto verdadero, siendo n el número de vectores no dominados en el frente de Pareto actual y d_i la distancia euclidiana entre cada una de éstas y el miembro más cercano del frente de Pareto verdadero:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

  Si la distancia generacional es 0 indica que todos los elementos generados están en el frente de Pareto verdadero. Cualquier otro valor indica que tan lejos se está del frente de Pareto verdadero. Esta métrica, al igual que la anterior requiere conocer el Frente de Pareto verdadero.
- Dispersión [42]: Mide la distribución de los elementos en el frente de Pareto actual sobre la región no dominada. También conocido como conjunto eficiente espaciado (ESS, por sus siglas en inglés), donde j = 1,…,e, $\bar{d}$se refieren a la media de todas las d_i y e, es el número de elementos del conjunto de Pareto obtenidos hasta el momento:

$$ESS = \sqrt{\frac{1}{e-1}} \sum_{i=1}^{e} \left(\bar{d} - d\right)^2,\ d_i = \min_j \left\{ \left| f_1^i - f_1^j \right| + \left| f_2^i - f_2^j \right| \right\}$$

  Esta métrica mide la varianza de la distancia de cada miembro del conjunto de óptimos de Pareto (encontrados hasta el momento) con respecto a su vecino más cercano. Un valor de 0 de esta métrica indica que todos los miembros del frente de Pareto generados están equidistantes.
- Cobertura [39]: Propone una métrica que compara dos conjuntos de vectores no dominados, calculando la fracción de cada uno que es cubierta (o dominada) por el otro. Se define de la siguiente manera:

$$C(A,B) = \frac{|\{a \in A;\ \exists b \in B : b \succcurlyeq a\}|}{|A|}$$

  Donde A y B son dos conjuntos de vectores de solución. Si todos los puntos en A dominan o son iguales a todos los puntos en B, entonces C = 0. En caso contrario C = 1.
  En este caso se analizan aquellos algoritmos que dominan a más cantidad de conjuntos y son dominados por la menor cantidad de conjuntos.

6 Experimentos

6.1 Caso de estudio

Para validar el modelo y los algoritmos de solución no se dispone de datos reales y tampoco se conoce de bases de datos de pruebas a nivel internacional para este tipo de problema. Tomando en cuenta esta situación se decidió aplicar una encuesta a un grupo de especialistas de software, con el objetivo de elaborar un algoritmo para generar los datos requeridos para evaluar el modelo.

Los datos necesarios para aplicar el modelo y que fueron recogidos en la encuesta incluyó:

- Niveles de competencia de los trabajadores.
- Experiencia de los trabajadores en el desempeño de roles.
- Carga de trabajo de los trabajadores.
- Incompatibilidades entre los trabajadores.
- Características psicológicas a través de los test de Myers-Briggs y Belbin.

Las competencias definidas para el caso de estudio incluyó las competencias técnicas: Dominio de lenguaje de programación (Java, C++, Delphi, PHP, C#), Dominio de gestor de base de datos (MySQL, SQLServer, Oracle, Postgres) y Dominio de tecnologías (J2EE, .NET), basado en las tecnologías actuales utilizadas por estos especialista. Además, se definieron las competencias genéricas: Trabajo en equipo y cooperación, Capacidad de análisis, Capacidad de planificación y organización, Compromiso con la organización y Comunicación.

Los niveles de competencia se definieron en el rango del 1 al 4, con los siguientes significados: 1 – conoce, 2 – tiene habilidad, 3 – conoce y ejecuta bien, 4 – experto.

La experiencia en el desempeño de los roles tiene en cuenta las veces que el trabajador ha desempeñado el rol y la evaluación recibida en su desempeño.

La carga de trabajo toma valores: Baja (0.25), Media (0.5), Alta (0.75) y Muy alta (1) en dependencia de los roles que desempeña el trabajador actualmente y la carga que implica cada rol.

Las incompatibilidades toman en cuenta incompatibilidad recíproca entre los trabajadores y tomará los valores 0 ó 1, donde 0 indica que no existe incompatibilidad recíproca entre los trabajadores y 1 que si existe.

6.2 Diseño de experimentos

Se diseñaron tres escenarios de prueba, teniendo en cuenta escenarios de organizaciones medianas (51-250 trabajadores), grandes (250-999 trabajadores) y especialmente grandes (1000 o más trabajadores), según la definición dada en [43]:

- Escenario 1: 100 trabajadores y 6 roles a cubrir (Organizaciones medianas)
- Escenario 2: 500 trabajadores y 10 roles a cubrir (Organizaciones grandes)
- Escenario 3: 1500 trabajadores y 10 roles a cubrir (Organizaciones especialmente grandes)

El caso de prueba a realizar tuvo en cuenta las tres funciones objetivos del modelo por defecto: Maximizar competencias, Minimizar incompatibilidades y Balancear carga de trabajo, teniendo en cuenta las restricciones de que un trabajador solo puede desempeñar un rol y las competencias mínimas establecidas para cada rol. Además, se tomaron en cuenta las restricciones asociadas con la sinergia del equipo: considerar la presencia de todas las categorías de roles de Belbin, elegir la presencia de al menos una persona con el rol cerebro y que exista un balance entre las categorías de roles de Belbin (la presencia en el equipo de individuos con preferencia por los roles de acción debe ser mayor que la preferencia por los roles mentales y la preferencia por los roles mentales debe ser mayor que la preferencia por los roles sociales).

Las condiciones relacionadas con el ambiente experimental fueron las siguientes:

- Los algoritmos fueron implementados en Java bajo el ambiente de desarrollo Eclipse 3.2, compilado con el JDK 1.6.0.
- Los experimentos fueron realizados en un procesador Intel Pentium CPU P6100 con 2GHz y 2Gb de RAM.

Se realizaron 20 ejecuciones de cada uno de los algoritmos implementados para cada escenario de prueba. Los parámetros utilizados para la ejecución de los diferentes algoritmos se muestran en la Tabla 3.

Todos los algoritmos utilizados tomaron como condición de parada un número máximo de evaluaciones de las funciones objetivo, definida como 45000.

Tabla 3: Parámetros de los algoritmos

Algoritmo/Parámetro	Escenarios		
	1	**2**	**3**
Escalador de Colinas Estocástico Multiobjetivo (ECEMO)			
Cantidad de iteraciones: 45000			
Escalador de Colinas Estocástico Multiobjetivo con Reinicio (ECEMO-R)			
Cantidad de iteraciones:45000			
Escalador de Colinas Estocástico Multiobjetivo de Mayor Distancia (ECEMO-Dist)			
Cantidad de iteraciones: 45000			
Búsqueda Tabú Multiobjetivo (BT Multiobjetivo)			
Tamaño de la lista Tabú: 20			
Cantidad de iteraciones	91	10	3
Recocido Simulado Multiobjetivo de Ulungu (UMOSA)			
Temperatura inicial: 2 Temperatura final: 0			
Mecanismo de enfriamiento: Esquema de Cauchy(Tk = To/(1+k)), k es el número de la iteración			
Cantidad de iteraciones: 45000			
Recocido Simulado Multiobjetivo Multicaso (MC-MOSA)			
Cantidad de iteraciones: 45000			

6.3 Análisis de resultados de los experimentos

Para valorar la calidad de las soluciones obtenidas con los algoritmos se consideraron las métricas analizas en el apartado 5, tomándose como referencia el valor ideal de cada métrica. El valor ideal de las métricas tasa de error, dispersión, distancia generacional y cobertura es 0, lo que significa que mientras más cercano a 0 es el valor de la métrica mejor es el desempeño del algoritmo.

Por otra parte, aunque para el problema que se desea resolver el tiempo de respuesta no resulta un factor crítico, se considera necesario tomarlo en cuenta en el análisis de los resultados de los experimentos, ya que es útil que el responsable de conformar el equipo pueda evaluar diferentes propuestas de equipo antes de tomar su decisión, y estas deben obtenerse en tiempos razonables.

En ninguno de los escenarios propuestos fue posible determinar el frente de Pareto verdadero, por lo que para el cálculo de las métricas que lo requieren, se tomó como frente de Pareto verdadero aquellas soluciones no dominadas de las obtenidas por todos los algoritmos ejecutados en cada escenario respectivamente. Para cada escenario se compararon las soluciones no dominadas obtenidas por cada uno de los algoritmos y se obtuvieron las soluciones no dominadas entre ellas, tomándolas como soluciones del frente de Pareto verdadero.

La cantidad de soluciones no dominadas en los frentes de Pareto verdadero tomados como referencia en cada escenario se muestran en el Tabla 4.

Tabla 4: Cantidad de soluciones en el frente de Pareto verdadero tomado como referencia

	Escenarios		
	1	**2**	**3**
Cantidad de soluciones en el frente de Pareto verdadero	4	2071	48

La cantidad de soluciones no dominadas encontradas por cada algoritmo en los diferentes escenarios se muestra en la Tabla 5.

Tabla 5: Cantidad de soluciones no dominadas obtenidas con cada algoritmo en los diferentes escenarios de prueba

Algoritmos	Escenarios		
	1	2	3
ECEMO	3	69	15
ECEMO-R	3	49	12
ECEMO-Dist	3	1004	25
BT Multiobjetivo	3	44	4
UMOSA	3	9	16
MC-MOSA	3	176	10

6.3.1 Escenario 1

Los valores de la métrica tasa de error en los diferentes algoritmos no son cercanos al valor ideal (el valor ideal es 0). Comparándolos entre ellos el algoritmo que mejores resultados obtiene es BT Multiobjetivo, donde el 74% de las soluciones son parte del frente de Pareto verdadero. Los algoritmos ECEMO, ECEMO-R, ECEMO-Dist y MC-MOSA obtienen resultados similares entre ellos, pero no son considerados buenos y UMOSA tiene muy malos resultados, ya que ninguna de las soluciones obtenidas son parte del frente de Pareto verdadero (ver Tabla 6).

La métrica distancia generacional se comporta de manera similar en todos los algoritmos, destacándose con el mejor resultado el algoritmo BT Multiobjetivo, resultado esperado teniendo en cuenta que fue el algoritmo con mayor porciento de soluciones en el frente de Pareto verdadero, por lo que la distancia a este es menor que en el resto de los algoritmos.

La dispersión es similar en todos los algoritmos, evidenciando que las soluciones obtenidas por cada algoritmo son equidistantes entre ellas.

En la métrica cobertura mostrada en la Tabla 7, se evidencia los buenos resultados de los algoritmos ECEMO, ECEMO-R, BT Multiobjetivo y MC-MOSA, cuyas soluciones no son dominadas por las obtenidas en los otros algoritmos.

Los tiempo obtenidos en los diferentes algoritmos resultan aceptable atendiendo al tipo de problema, en ningún caso supera los 2 minutos.

Tabla 6: Valores promedio y posicionamiento de los algoritmos en función de las diferentes métricas para el Escenario 1 (1-mejor, 6-peor)

Escenario 1. 100 trabajadores y 6 roles								
Algoritmos	Tasa de error	Orden	Distancia generacional	Orden	Dispersión	Orden	Tiempo (minutos)	Orden
ECEMO	0,8417	5	0,0204	5	0,0010	5	0,7836	3
ECEMO-R	0,8333	4	0,0167	4	0,0006	3	1,5308	5
ECEMO-Dist	0,8250	3	0,0208	6	0,0009	4	1,6772	6
BT Multiobjetivo	0,3583	1	0,0046	1	0,0002	1	0,1314	1
UMOSA	1,0000	6	0,0162	3	0,0010	5	0,7647	2
MC-MOSA	0,7917	2	0,0140	2	0,0004	2	0,7878	4

Tabla 7: Métrica cobertura para Escenario 1

Algoritmos	ECEMO	ECEMO-R	ECEMO-Dist	BT Multiobjetivo	UMOSA	MC-MOSA
ECEMO	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
ECEMO-R	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
ECEMO-Dist	0,3333	0,3333	0,0000	0,3333	0,3333	0,3333
BT Multiobjetivo	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
UMOSA	1,0000	1,0000	0,6667	1,0000	0,0000	1,0000
MC-MOSA	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000

A partir de este análisis se puede concluir que UMOSA es considerado el peor algoritmo teniendo en cuenta las métricas tasa de error (ninguna de las soluciones obtenidas con UMOSA son parte del "frente de Pareto

verdadero") y cobertura (en casi todas los algoritmos generan soluciones que dominan en su totalidad a las soluciones obtenidas con UMOSA).

Para este escenario el algoritmo que mejores resultados obtiene en cuanto a todas las métricas es BT Multiobjetivo.

6.3.2 Escenario 2

Los valores obtenidos por los algoritmos ECEMO, ECEMO-R, ECEMO-Dist y MC-MOSA en la métrica tasa de error no muestra diferencias significativas entre ellos, siendo ECEMO-R el que mejores resultados obtiene (el 94% de las soluciones son parte del frente de Pareto verdadero). Por otra parte, los valores obtenidos de distancia generacional y dispersión no muestran diferencias significativas entre ninguno de los algoritmos y los valores son muy cercanos al valor ideal en todos los casos. (ver Tabla 8)

Los tiempo obtenidos en todos los casos resultan aceptable atendiendo al tipo de problema, en ningún caso supera los 3 minutos.

La métrica cobertura muestra resultados insatisfactorios para los algoritmos BT Multiobjetivo y UMOSA. El algoritmo que mejor se comporta es ECEMO-Dist cuyas soluciones no son dominadas por las obtenidas en ninguno de los otros algoritmos. El resto de los algoritmos (ECEMO, ECEMO-R, MC-MOSA) se comportan similares entre ellos, tal como se muestra en la Tabla 9.

Tabla 8: Valores promedio y posicionamiento de los algoritmos en función de las diferentes métricas para el Escenario 2 (1-mejor, 6-peor)

Escenario 2. 500 trabajadores y 10 roles								
Algoritmos	**Tasa de error**	**Orden**	**Distancia generacional**	**Orden**	**Dispersión**	**Orden**	**Tiempo (minutos)**	**Orden**
ECEMO	0,1055	3	0,0011	4	0,0003	3	0,9865	2
ECEMO-R	0,0687	1	0,0007	1	0,0002	2	1,6757	5
ECEMO-Dist	0,1481	4	0,0009	3	0,0003	3	2,3021	6
BT Multiobjetivo	0,6997	5	0,0033	5	0,0001	1	0,3898	1
UMOSA	1,0000	6	0,0171	6	0,0009	4	1,0259	3
MC-MOSA	0,0755	2	0,0008	2	0,0002	2	1,3059	4

A partir de estos resultados se puede decir que en este escenario los algoritmos ECEMO, ECEMO-R, ECEMO-Dist y MC-MOSA obtienen buenos resultados en las métricas analizadas y los algoritmos con peores resultados en las métricas tasa de error, distancia generacional y cobertura son UMOSA y BT Multiobjetivo.

Tabla 9: Métrica cobertura para Escenario 2

Algoritmos	**ECEMO**	**ECEMO-R**	**ECEMO-Dist**	**BT Multiobjetivo**	**UMOSA**	**MC-MOSA**
ECEMO	0,0000	0,3776	0,3787	0,0000	0,0000	0,3776
ECEMO-R	0,0000	0,0000	0,0014	0,0000	0,0000	0,0000
ECEMO-Dist	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
BT Multiobjetivo	0,9792	0,9792	0,9792	0,0000	0,0000	0,9792
UMOSA	1,0000	1,0000	1,0000	1,0000	0,0000	1,0000
MC-MOSA	0,0034	0,0034	0,0034	0,0000	0,0000	0,0000

6.3.3 Escenario 3

La métrica tasa de error tiene valores similares para los algoritmos ECEMO, ECEMO-R, ECEMO-Dist y MC-MOSA, siendo este último el que mejores resultados obtiene (el 82% de las soluciones son parte del frente de Pareto verdadero).

La métrica distancia generacional y dispersión no tiene diferencias significativas entre los algoritmos ECEMO, ECEMO-R, ECEMO-Dist y MC-MOSA. Los tiempo obtenidos en todos los algoritmos resultan aceptable atendiendo al tipo de problema, en ningún caso supera los 5 minutos.

La métrica cobertura obtiene resultados similares y aceptables en los algoritmos ECEMO, ECEMO-R, ECEMO-Dist y MC-MOSA. Haciendo una valoración de los resultados de las métricas que obtiene valores malos

(tasa de error y cobertura) se puede observar en las Tablas 10 y 11 que los algoritmos con peores resultados en dichas métricas fueron UMOSA y BT Multiobjetivo.

Tabla 10: Valores promedio y posicionamiento de los algoritmos en función de las diferentes métricas para el Escenario 3 (1-mejor, 6-peor)

Escenario 3. 1500 trabajadores y 10 roles								
Algoritmos	**Tasa de error**	**Orden**	**Distancia generacional**	**Orden**	**Dispersión**	**Orden**	**Tiempo (minutos)**	**Orden**
ECEMO	0,2584	4	0,0035	4	0,0002	2	1,5163	2
ECEMO-R	0,2396	3	0,0027	1	0,0002	2	4,4244	6
ECEMO-Dist	0,2375	2	0,0030	2	0,0001	1	4,2428	5
BT Multiobjetivo	1,0000	5	0,0329	6	0,0001	1	1,1150	1
UMOSA	1,0000	6	0,0291	5	0,0007	3	1,7244	4
MC-MOSA	0,1849	1	0,0031	3	0,0002	2	1,7037	3

Tabla 11: Métrica cobertura Escenario 3

Algoritmos	**ECEMO**	**ECEMO-R**	**ECEMO-Dist**	**BT Multiobjetivo**	**UMOSA**	**MC-MOSA**
ECEMO	0,0000	0,0930	0,1395	0,0000	0,0000	0,1628
ECEMO-R	0,1026	0,0000	0,1538	0,0000	0,0000	0,2051
ECEMO-Dist	0,2182	0,2000	0,0000	0,0000	0,0000	0,2545
BT Multiobjetivo	1,0000	1,0000	1,0000	0,0000	0,7500	1,0000
UMOSA	1,0000	1,0000	1,0000	0,3125	0,0000	1,0000
MC-MOSA	0,0541	0,0811	0,0270	0,0000	0,0000	0,0000

A partir de los resultados obtenidos en los diferentes escenarios se puede concluir que:

- En los experimentos analizados se destacan los buenos resultados obtenidos con los algoritmos ECEMO, ECEMO-R, ECEMO-Dist y MC-MOSA en los diferentes escenarios de prueba.
- El algoritmo BT Multiobjetivo obtiene buenas soluciones en el primer escenario de prueba (Escenario 1 – (100 trabajadores y 6 roles)) siendo uno de los peores algoritmos en los dos últimos escenarios de prueba (Escenario 2 – (500 trabajadores y 10 roles) y Escenario 3 – (1500 trabajadores y 10 roles)).
- El algoritmo con peores resultados en las métricas analizadas fue UMOSA.

7 Conclusiones y trabajos futuros

Los algoritmos de trayectoria multiobjetivo han sido utilizados anteriormente en la solución de una gran variedad de problemas, entre los que se encuentran los problemas de asignación. En el caso de este trabajo que se presenta de su aplicación al problema de conformación de equipos de proyectos de software, se obtuvieron buenas soluciones en tiempos aceptables (que no excedan los 5 minutos) para diferentes tamaños del problema, que incluyeron escenarios de organizaciones medianas, grandes y especialmente grandes.

Se destacan los buenos resultados de algoritmos más simples como son las variantes del Escalador de Colinas, aun para escenarios de organizaciones grandes y especialmente grandes.

Se propone como trabajos futuros aplicar variantes poblacionales multiobjetivo al problema de conformación de equipos de proyectos de software para comparar los resultados que se obtienen y validar los algoritmos que se presentan con datos reales de una organización de software.

Referencias bibliográficas

1. Coello, C.A.C., D.A.V. Veldhuizen, and G.B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Second ed. Genetic and Evolutionary Computation Series, ed. D.E. Goldberg and J.R. Koza. 2007, New York: Springer. 800 DOI: 10.1007/978-0-387-36797-2
2. André, M., M.G. Baldoquín, and S.T. Acuña, *Formal model for assigning human resources to teams in software projects.* Information and Software Technology, 2011. **53**(3): p. 16 DOI: 10.1016/j.infsof.2010.11.011.

3. Acuña, S.T. and N. Juristo, *Chapter 5: Software Process Modeling: Socio-Technical Perspectives, in Software Process Modeling*, in *The Kluwer International Series in Software Engineering*. 2005, Springer. p. 111-139.
4. André, M. (2009). *Un modelo para la asignación de recursos humanos a equipos de proyectos de software.* Unpublished Tesis de doctorado, ISPJAE, Ciudad de La Habana, Cuba.
5. Barreto, A.S. (2003). *Apoio à Decisão Gerencial na Alocação de Recursos Humanos em Projetos de Software.* Universidade Federal do Rio de Janeiro, Rio de Janeiro.
6. DeCarvalho, L.R. (2003). *Planejamento da alocação de recursos humanos em Ambientes de desenvolvimento de software orientados à Organização.* Unpublished Tese para a obtenção do grau de mestre em ciências em engenharia de sistemas e computação, Universidade Federal do Rio de Janeiro, COPPE, Rio de Janeiro.
7. Ngo-The, A. and G. Ruhe, *A Systematic Approach for Solving the Wicked Problem of Software Release Planning.* Soft Computing, 2008. **12**(1): p. 95-108 DOI: 10.1007/s00500-007-0219-2.
8. André, M., M.G. Baldoquín, and S.T. Acuña, *Identification of patterns for the formation of software development projects teams.* International Journal of Human Capital and Information Technology Professionals (IJHCITP), 2010. **1**(3): p. 11 DOI: 10.4018/jhcitp.
9. Glover, F., *Future Paths for Integer Programming and Links To Artificial Intelligence.* Computers & Operations Research, 1986. **13**(5): p. 17 DOI: 10.1016/0305-0548(86)90048-1.
10. Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi, *Optimization by Simulated Annealing.* Science, 1983. **220**: p. 10 DOI: 10.1126/science.220.4598.671.
11. Rich, E. and K. Knight, *Inteligencia Artificial*. 1994: McGraw-Hill.
12. Feo, T.A. and M.G.C. Resende, *Greedy randomized adaptive search procedures.* Journal of Global Optimization, 1995. **6**(24): p. 109 DOI: 10.1007/BF01096763.
13. Ahuja, R.K., J.B. Orlin, and A. Tiwari, *A Greedy Genetic Algorithm for the Quadratic Assignment Problem.* Computers and Operations Research 2000. **27**: p. 22 DOI: 10.1016/S0305-0548(99)00067-2
14. Ramkumar, A.S. and S.G. Ponnambalam. *Hybrid Ant colony System for solving Quadratic Assignment Formulation of Machine Layout Problems* in *Cybernetics and Intelligent Systems*. 2006. IEEE ConferenceDOI: 10.1109/ICCIS.2006.252286.
15. Sahu, A. and R. Tapadar, *Solving the Assignment problem using Genetic Algorithm and Simulated Annealing.* IAENG International Journal of Applied Mathematics, 2007. **36**(1): p. 4 DOI: 10.1.1.98.8085.
16. Fleurent, C. and J.A. Ferland, *Genetic Hybrids for the Quadratic Assignment Problem.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1993: p. 15 DOI: 10.1.1.47.7802.
17. Chu, P.C. and J.E. Beasley, *A Genetic Algorithm for the Generalized Assignment Problem.* Computers & Operations Research, 1997. **24**: p. 17-23 DOI: 10.1287/ijoc.1030.0036.
18. Vazquez, M. and L.D. Whitley, *A hybrid Genetic Algorithm for the Quadratic Assignment Problem*, in *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference* 2000. p. 8.DOI: 10.1.1.152.8794.
19. Drezner, Z., *A New Heuristic for the Quadratic Assignment Problem.* Journal of Applied Mathematics and Decision Sciences, 2002. **6**(3): p. 11 DOI: 10.1155/S1173912602000093.
20. Mishmast, H. and S. Gelareh, *A Survey of Meta-Heuristic Solution Methods for the Quadratic Assignment Problem.* Applied Mathematical Sciences, 2007. **1**(46): p. 20.
21. Donoso, Y., P. Albor, and A. Benavides, *Optimización con múltiples objetivos utilizando Tabú Search.* Ingeniería & Desarrollo, 2005(18): p. 16.
22. Czyzak, P. and A. Jaszkiewicz, *Pareto Simulated Annealing - a metaheuristic technique for multiple objective combinatorial optimization.* Journal of Multi-Criteria Decision Analysis, 1998. **7**: p. 22 DOI: 10.1002/(SICI)1099-1360(199801)7:1<34::AID-MCDA161>3.0.CO;2-6.
23. Díaz, R. (2001). *Estudio de la capacidad del algoritmo Escalador de Colinas Estocástico para enfrentar problemas multiobjetivo.* Unpublished Master en Informática Aplicada, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana.
24. Jaffrès, K., J.M. Gorce, and C. Comaniciu, *A Multiobjective Tabu Framework for the Optimization and Evaluation of Wireless Systems*, in *Local Search Techniques: Focus on Tabu Search* 2008: Vienna, Austria. p. 278.
25. Reynolds, A.P., D.W. Corne, and B.d.l. Iglesia. *A Multiobjective GRASP for Rule Selection*. in *GECCO'09*. 2009. Montreal Quebec, Canadá: ACMDOI: 10.1145/1569901.1569990.
26. Baykasoglu, A., L. Ozbaku, and T. Dereli. *Multiple dispatching rule based heuristic for multiobjective scheduling of job using Tabu Search* in *5th International Conference on Managing Innovations in Manufacturing*. 2002. Milwaukee, Wisconsin, USADOI: 10.1.1.75.795.

27. Kulcsár, G., F. Erdélyi, and O. Hormyák, *Multi-objective optimization and heuristic approaches for solving scheduling problems.* IFAC, 2007.
28. Burke, E.K., J. Li, and R. Qu, *Pareto-Based Optimization for Multi-objective Nurse Scheduling.* Computer Science Technical Report No. NOTTCS-TR-2007-5, 2007 DOI: 10.1.1.62.7271.
29. Chang, W. and C. Chyu. *Comparison between SA-based and EA-based Metaheuristics for Solving a Biobjective Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times*. in *9th Asia Pacific Industrial Engineering & Management Systems Conference*. 2008.
30. Hamm, M., U. Beißert, and M. König, *Simulation-based optimization of construction schedules by using pareto simulated annealing*, in *18th International Conference on the Application of Computer Science and Mathematics in Architecture and Civil Engineering*, K.G.a.C. Könke, Editor. 2009: Weimar, Germany.
31. Hapke, M., A. Jaszkiewicz, and R. Slowinski, *Pareto simulated annealing for fuzzy multiobjective combinatorial optimization.* Journal of Heuristics, 2000. **6**(3): p. 16 DOI: 10.1023/A:1009678314795.
32. Ulungu, E.L. and J. Teghem, *Multi-objective combinatorial optimization problems: A survey.* Journal of Multi-Criteria Decision Analysis, 1994. **3**(2): p. 22 DOI: 10.1002/mcda.4020030204.
33. Wolpert, D.H. and W.G. Macready, *No free lunch theorems for optimization.* IEEE Transactions on Evolutionary Computation, 1997. **1**(1): p. 67–82 DOI: 10.1109/4235.585893
34. André, M., et al. *A formalized model for the assignment of human resources to software projects*. in *XIV Latin Ibero-American Congress on Operations Research CLAIO 2008*. 2008. Colombia.
35. Suman, B. and P. Kumar, *A survey of simulated annealing as a tool for single and multiobjective optimization.* Journal of the Operational Research Society, 2006. **57**(10): p. 18 DOI: 10.1057/palgrave.jors.2602068.
36. Haidine, A. and R. Lehnert, *Multi-Case Multi-Objective Simulated Annealing (MC-MOSA): New Approach to Adapt Simulated Annealing to Multi-objective Optimization.* International Journal of Information Technology, 2008. **4**(3): p. 9.
37. Balicki, J., *Tabu Programming for Multiobjective Optimization Problems.* International Journal of Computer Science and Network Security, 2007. **7**(10): p. 8 DOI: 10.1287/ijoc.7.4.426.
38. Johnsonbaugh, R., *Matemáticas Discretas*. 2004, La Habana: Félix Varela.
39. Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* Swiss Federal Institute of Technology Zurich DOI: 10.1.1.39.9023.
40. Knowles, J. and D. Corne, *On Metrics for Comparing Non-Dominated Sets*, in *IEEE Congress on Evolutionary Computation - CEC*. 2001. p. 711-716.DOI: 10.1.1.18.5759.
41. Veldhuizen, D.V., Multiobjective evolutionary algorithm: Classifications, Analyses, and New Innovations, in Department of Electrical and Computer Engineering. 1999, Air Force Institute of Technology Wright Patterson AFB: Ohio.
42. Schott, J.R., Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization, in Department of Aeronautics and Astronautics. 1995.
43. Brandt, C. and R. Lindberg (2006). *Competitive IS/IT strategy – A qualitative study about how IS/IT strategy can influence business strategy in small service enterprises.* Unpublished Master's thesis in Business Informatics, Jönköping University.