Metaheurísticas basadas en Inteligencia Computacional Aplicadas a la Resolución de Problemas de Optimización Numérica con y sin Restricciones y Optimización Combinatoria

Victoria Aragón[†], Leticia Cagnina[†], Claudia Gatica[†], Susana Esquivel[†]

[†]Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Facultad de Ciencias Físicas, Matemáticas y Naturales
Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106
(5700) - San Luis - Argentina
Tel: (02652) 420823 / Fax: (02652) 430224
e-mail: {esquivel, vsaragon, lcagnina, crgatica}@unsl.edu.ar

Resumen

En esta presentación se describen en forma breve algunas de las direcciones de investigación que en la actualidad se están desarrollando dentro de la línea "Optimización Mono y Multiobjetivo" del Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC). Uno de los objetivos de esta línea, es el estudio y desarrollo de metaheurísticas aptas para resolver problemas de optimización numérica y combinatoria. En particular, el énfasis está puesto en las heurísticas de la inteligencia computacional basadas en los paradigmas de inteligencia colectiva y biológicos.

1. Introducción

En la actualidad la línea de investigación focaliza su trabajo sobre tres heurísticas: Sistemas Inmunes (SI), Particle Swarm Optimizers (PSO) y Algoritmos Genéticos (AGs) Centralizados y Paralelos, atacando los siguientes problemas: Optimización Numérica con restricciones (SI, PSO), Optimización Multiobjetivo (PSO) y Problemas de Scheduling de Máquinas Paralelas Idénticas (AGs). En la sección dos se describe el grado de avance de cada problema tratado y la sección tres presenta las conclusiones y futuras líneas de trabajo.

2. Grado de Avance

2.1. Sistemas Inmunes

Actualmente el estudio está centrado en el desarrollo de un nuevo modelo de sistema inmune inspirado en el sistema inmune de los vertebrados. El modelo propuesto está basado en los linfocitos T.

El modelo trabaja sobre 3 subpoblaciones: células T vírgenes (CV), células T efectoras (CE) y células T de memoria (CM). Las subpoblaciones CV y CE contienen células representadas por cadenas binarias en código gray, en cambio, CM contiene cadenas de valores reales para representar a las células de memoria. Esto está inspirado en el hecho que las células de memoria poseen una representación fenotípica diferente a las otras dos. Los pasos del algoritmo se indican a continuación:
1. Para un número prefijado de iteraciones:
2. Generar NV células vírgenes en CV.
3. Evaluar función objetivo para determinar afinidad.
4. Efectuar procesos de proliferación (clonación) y de diferenciación (mutación).
5. Formar CE y CM con las mejores células de CV (reemplazando a las peores si no están vacias).
6. Aplicar proceso de reacción a CE, con las mejores formar una nueva CE.
7. Aplicar proceso de reacción a CM, con las mejores formar una nueva CM.
8. Mover mejores células de CE a CM (acorde con un porcentaje predeterminado).
9. Si corresponde decrementar en 1 el tamaño de CV e incrementar en 1 el de CE y CM.
El último paso tiene como objeto permitir una exploración global en las primeras etapas de la búsqueda y una exploración más localizada sobre las células consideradas promisorias hacia el final del proceso.

El modelo anterior se desarrolló incialmente para optimizar funciones sin restricciones y el mismo fue validado con un conjunto de funciones de prueba tomado de la literatura especializada. Para poder aplicar el modelo propuesto a funciones con restricciones, fue necesario realizar pequeños cambios para distinguir las zonas del espacio de búsqueda que son factibles de las no factibles. En consecuencia la acción de determinar la afinidad de una célula ahora implica no sólo calcular su valor de función objetivo sino también verificar la cantidad de restricciones que satisface, el grado de violación de las restricciones y consecuentemente si la célula representa una solución factible o no. Por lo tanto, en la etapa de afinidad se incorporó el siguiente esquema de manejo de restricciones, dadas dos soluciones:
1. Si las dos soluciones son factibles se tiene en cuenta el valor de la función objetivo para determinar cuál es la mejor.
2. Si una solución es factible y la otra no lo es, entonces la mejor solución es la factible.
3. Si las dos son no factibles entonces se toma como mejor solución a aquélla que viole en menor grado las restricciones del problema.
También el operador de mutación fue modificado puesto que si se está aplicando a una solución factible debe modificarla levemente para permitir la exploración del vecindario de la solución factible mientras que si la solución es no factible deberá modificarse en mayor medida, para permitirle escapar del óptimo local en el cual puede estar atrapada dicha solución. En este último caso, la dureza del cambio estará afectada por el grado de violación de las restricciones. Al ser evaluado el modelo (incorporado ya el esquema de manejo de restricciones descripto), usando las funciones con restricciones [10], los resultados obtenidos no fueron del todo satisfactorios. Por esta razón, actualmente se está trabajando en una modificación del esquema de manejo de restricciones. A continuación se describen brevemente los aspectos más relevantes de dicha modificación.

Cada población sigue encargada se efecutar una tarea específica: CV aportar diversidad, CE explorar la frontera entre la zona factible y la no factible y por último CM explorar la vecindad de las mejores soluciones encontradas. La diferencia radica en la forma de realizar estas tareas. A continuación se presenta el pseudocódigo del nuevo modelo:
1. Para un número determinado de iteraciones.
2. Inicializar población CV.
3. Dividir CV.

4. Reacción CE.
5. Inserción CE en CM.
6. Reacción CM.
En el primer paso se inicializa completamente la población de células vírgenes (CV) pero a medida que avanza la evolución la cantidad de células disminuye puesto que se espera que hacia el final del proceso no será necesaria mucha diversidad porque se ha alcanzado la vecindad de la solución óptima. Luego, se divide CV en soluciones factibles y no factibles insertándolas en CE_fac y en CE_nfac respectivamente. El proceso de reacción sobre ambas CEs es de la siguiente forma: CE_fac muta con un tamaño de paso pequeño para explorar el vecindario de cada solución factible encontrada mientras que para aplicar la mutación en CE_nfac se incorpora información del problema. Esto es, se asocia a cada solución el grado de violación de cada restricción como también la suma de éstas. También se ha asociado qué variables de decisión influyen en cada restricción. Esta información se usa al momento de aplicar el operador de mutación a las soluciones no factibles, para ello se busca la restricción con mayor violación y se modifican sólo las variables involucradas en dicha restricción. Este proceso hasta el momento parece ser satisfactorio para encontrar la zona factible. Finalmente se inserta un porcentaje de las mejores soluciones de CE_fac y CE_nfac en CM, priorizando las factibles sobre las no factibles y se aplica nuevamente el operador de mutación con paso pequeño para poder explorar el vecindario de las soluciones almacenadas en CM.

Actualmente se está revaluando el algoritmo modificado con el conjunto de funciones de prueba standarizado.

2.2. Particle Swarm Optimizer

En [15] se presentó el algoritmo CPSO para optimización de funciones con restricciones. Los resultados fueron contrastados con los de Stochastic Ranking [9] y con los de un algoritmo PSO [13] y publicados en un evento internacional [1]. El algoritmo es una implementación básica de la heurística PSO al que se le agregó un mecanismo sencillo de manejo de restricciones. Las restricciones fueron consideradas como desigualdades, convirtiendo las igualdades a través de un factor ϵ de tolerancia. CPSO incluye un operador de mutación y una fórmula particular de actualización de partículas la cual evita la convergencia prematura hacia óptimos locales. Los detalles de implementación pueden consultarse en [1] como así también los resultados obtenidos para las 13 funciones de prueba elegidas. Actualmente se está trabajando en una versión mejorada de este algoritmo llamado CPSO-shake, debido al mecanismo de movimiento de partícula que hemos incorporado. El fin de este mecanismo es aumentar la diversidad de la población de forma tal de escapar de zonas de estancamiento. CPSO-shake está siendo evaluado con el benchmark extendido presentado en [5], el cual cuenta con 20 funciones de prueba de diferente tipo. Los resultados obtenidos están siendo contrastados con el algoritmo Stochastic Ranking (SR) [9] el cual sigue siendo uno de los más representativos en el área de optimización restringida, así como también con un nuevo algoritmo propuesto en el 2006 [8] que es una aproximación de una estrategia evolutiva (AESSR) que utiliza también Stochastic Ranking. La primera comparación se realiza de manera directa ya que tanto CPSO-shake como SR realizan la misma cantidad de evaluaciones: 350000. Para la segunda comparación se realiza una comparación indirecta ya que el número de evaluaciones reportadas en [8] es superior a la utilizada por CPSO-shake, o sea: 500000.
Estos resultados están siendo reportados a un congreso internacional para su evaluación, aunque en estas pruebas previas puede observarse la alta competividad de CPSO-shake cuando es comparado con SR y aún con AESSR el cuál utiliza más evaluaciones. Un resumen de los mejores valores obtenidos

por los algoritmos para las 20 funciones de prueba, puede ser visto a continuación. En la figura 1 se compara el algoritmo CPSO-shake con SR y con AESSR en la figura 2 para el total de funciones de prueba.

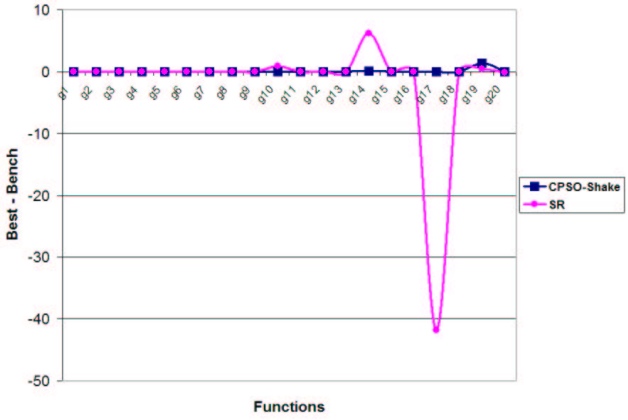

Figura 1: Comparación entre CPSO-shake y Stochastic Ranking

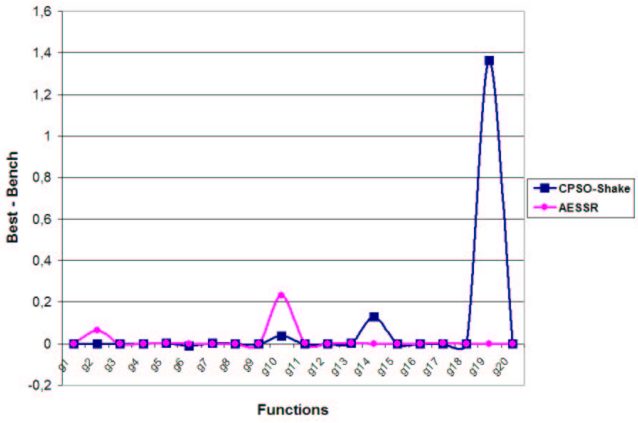

Figura 2: Comparación entre CPSO-shake y Approximate Evolution Strategy con Stochastic Ranking

Adicionalmente se está trabajando en la adaptación de la heurística PSO para problemas de optimización multicriterio, en primer instancia, para ambientes estáticos y, posteriormente, si los resultados obtenidos son buenos se extenderá para su aplicación en ambientes dinámicos.

2.3. Algoritmos Genéticos

Un interesante problema de optimización combinatoria es el problema de scheduling de máquinas paralelas idénticas, común en los sistemas de producción, donde es usual tratar de minimizar objetivos relacionados con el *due date*. El modelo básico se define: Las *n* tareas son procesadas sin interrupción sobre algunas de las *m* máquinas paralelas idénticas del sistema (P_m), y cada máquina puede procesar una única tarea a la vez. La tarea j (*j=1, 2, ..., n*) se hace disponible para su procesamiento en el tiempo cero, y ésta requiere un tiempo positivo y sin interrupciones para su procesamiento p_j, tiene un *due date* d_j en el cual la tarea debe estar idealmente terminada, y tiene un peso w_j.

Las funciones objetivo que nos interesa (relacionadas con *due date*) minimizar son: el *maximum tardiness* (T_{max}), *average tardiness* (T_{avg}), *weighted tardiness* (T_{wt}), *number of tardy jobs* (N_t), *weighted number of tardy jobs* (N_{wt}).

Para estos problemas de scheduling, en la literatura [7] [6] se definen diversas reglas de despacho y heurísticas que proveen soluciones buenas en tiempos razonablemente cortos, dependiendo de que el número de tareas no sea muy grande.

Se pueden clasificar los AGs como: *centralizados secuenciales* donde existe una única población y se ejecutan secuencialmente sobre un único procesador; *descentralizados paralelos*, los cuales a su vez pueden subdividirse en gránulo grueso o AG distribuido, en ellos la población se sudivide en islas y cada isla ejecuta independientemente existiendo intercambios de individuos, entre las islas con una frecuencia dada o gránulo grueso o AG celular , donde los individuos son ubicados en una grilla toroidal d-dimensional (con d=1,2 y 3 es usado en la práctica), donde un individuo es ubicado en una celda de la grilla y se comunica con sus vecinos [12]. En la actualidad se está trabajando con un AG simple, tratando de optimizar en primer instancia las funciones objetivo maximun tardiness y average tardiness definidas:

$$T_{max} = max_j(T_j)$$

$$T_{avg} = \frac{1}{n}\sum_{j=1}^{n} T_j$$

Para llevar adelante los experimentos se han seleccionado 20 instancias de prueba de 100 jobs que fueron utilizadas en trabajos anteriores [4] [3] [2] y tienen un valor óptimo (el mejor valor encontrado hasta el momento) conocido y que son los que se usarán como benchmarks para comparar nuestros resultados. Todos los algoritmos implementados y ha implementar usan la librería MALLBA [14]. En una primera etapa se trabaja con un modelo centralizado secuencial y con un modelo distribuido (de gránulo grueso). Los algoritmos se ejecutan sobre un sistema paralelo, *cluster*, con el que cuenta el LIDIC. Algunos resultados preeliminares ya muestran que un algoritmo genético simple, como el que se ha implementado, no logra buenos resultados para un problema duro como el que se está tratando (tanto en su versión secuencial como distribuida). Estos resultados pueden consultarse en [11]. Por tal razón los investigadores abocados a este problema están rediseñando el algoritmo para insertarle nuevas potencias, tales como insersión de conocimiento del problema, multirecombinación entre otras.

3. Conclusiones

En los problemas de optimización restringidos, el grupo que trabaja con sistemas inmunes se centrará en mejorar el esquema de manejo de restricciones y luego, el modelo será adaptado para trabajar sobre problemas de optimización en ambientes dinámicos y problemas de ingeniería del mundo real. El grupo que trabaja sobre el mismo tipo de problemas pero usando particle swarm optimizer dedicará, en primer instancia, su esfuerzo hacia el análisis de cómo lograr una mayor robustez del algoritmo, de manera de conseguir un comportamiento más homogéneo para las diferentes funciones de prueba y una menor variabilidad en ciertos estadísticos. En cuanto al problema de scheduling como ya ha sido expresado se está trabajando en una nueva versión del algoritmo. Una vez que éste esté puesto a punto tanto para el modelo secuencial como distribuido se encarará el modelo celular.

4. Reconocimientos

El LIDIC reconce el constante soporte brindado por la Universidad Nacional de San Luis y la ANPYCIT que financian sus actuales investigaciones.

Referencias

[1] L. C. Cagnina, Susana C. Esquivel, and C. A. Coello Coello. A particle swarm optimizer for constrained numerical optimization. In *9th International Conference - Parallel problem Solving from Nature - PPSN IX*, pages 910–919, Reykjavik, Island, 2006.

[2] Ferretti E. and Esquivel S. A comparison of simple and multirecombinated evolutionary algorithms with and without problem specific knowledge insertion, for parrallel machines scheduling. *International Transaction on Computer Science and Engineering*, 3(1):207–221, 2005.

[3] Ferretti E. and Esquivel S. An efficient approach of simple and multirecombinated genetic algorithms for parallel machine scheduling. In *IEEE Congress on Evolutionary Computation*, volume 2, pages 1340–1347, Scotland, UK, September 2005. IEEE Center.

[4] Ferretti E. and Esquivel S. Knowledge insertion: An efficient approach to simple genetic algorithms for unrestricted for parallel equal machines scheduling. In *GECCO'05*, pages 1587–1588, Washington DC, USA, 2005.

[5] http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-06/CEC06.html.

[6] T. Morton and D. Pentico. *Heuristic Scheduling Systems*. John Wiley and Sons, New York, 1993.

[7] M. Pinedo. *Scheduling: Theory, Algorithms and System*. Prentice Hall, 1995.

[8] T. P. Runarsson. Approximate evolution strategy using stochastic ranking. In *2006 IEEE World Congress on Computation Intelligence*, volume 3, pages 2760–2767, British Columbia, Canada, 2006.

[9] T. P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. In *IEEE Transactions on Evolutionary Computation*, volume 3, pages 284–294, 2000.

[10] Thomas P. Runarsson and Xin Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.

[11] Esquivel S. and Gatica C. A comparision between centralized and decentralized genetic algorithms for the identical parallel machines scheduling. In *XII Congreso Argentino de Ciencias de la Computación*.

[12] Enrique Torres Alba. *Parallel Metaheuristics, A New Class of Algorithms), publisher = Wiley-Interscience, year = 2005*.

[13] G. Toscano Pulido and C. A. Coello Coello. A constrained-handling mechanism for particle swarm optimization. In *Congress on Evolutionary Computation*, pages 1396–1403, Portland, Oregon, USA, 2004.

[14] España Universidad de Malaga. http://neo.lcc.uma.es/mallba/mallba.html.

[15] Aragón V., Cagnina L., and Esquivel S. Metaheurísticas basadas en inteligencia computacional aplicadas a la resolución de problemas de optimización restringidos. In *VIII Workshop de Investigadores en Ciencias de la Computación (WICC 2006)*, pages 195–201, Universidad Nacional de Morón, Buenos Aires, Argentina, 2006.