La programación multiobjetivo es bastante diferente de la programación de un solo objetivo en los siguientes dos aspectos:

(1) Generalmente existen conflictos directos entre diferentes objetivos bajo consideración. Esto significa que, para obtener mejoras en un objetivo, algunos otros objetivos pueden tener que empeorar. Por lo tanto, si las funciones objetivo no son

positivamente correlacionadas entre sí, no existirá una solución única que sea óptima con respecto a cada uno de los objetivos.

(2) Los usuarios (gerentes de producción) requieren que el algoritmo de optimización genere un conjunto de soluciones satisfactorias (en lugar de una solución como en el caso de un solo objetivo) que sean suficientes en cantidad y que se distribuyan uniformemente. Luego, el responsable de la toma de decisiones puede elegir la política de programación más adecuada de este conjunto en función del escenario particular que enfrenta (por ejemplo, si la puntualidad en la entrega es el cuello de botella actual, puede ser necesario seleccionar los horarios que están sesgados a favor). de las funciones objetivas relacionadas con la tardanza).

Con este fin, primero proponemos nuevas relaciones de Pareto para clasificar las soluciones inciertas generadas tanto en casos mono-objetivo como multiobjetivo. En segundo lugar, sugerimos una extensión de dos conocidos algoritmos evolutivos de Pareto-base, a saber, SPEA2 y NSGAII.

Dominio de Pareto,

Optimización de Pareto,

Conjunto óptimo de Pareto

Frente de Pareto.

Dichos algoritmos se basan en tres componentes principales, a saber, asignación de actividad física, preservación de la diversidad y elitismo.

hibridizando metaheurisicas y programación matemática

debe interpretarse a la luz de lo que realmente significa el análisis de complejidad: por un lado, el hecho de que un problema pertenezca a la clase de N P-problemas difíciles implica que no hay conocimiento de un algoritmo capaz de resolver el problema por sí mismo a la optimalidad en tiempo polinomial con respecto a su tamaño de entrada (y muchos creen que nunca lo habrá); por otro lado, vale la pena recordar que el análisis de complejidad proporciona un escenario de peor caso, es decir, indica que, en el peor de los casos, con el crecimiento del tamaño de entrada, el algoritmo requerirá más y más tiempo / pasos para proporcionar una respuesta definitiva.

Este es exactamente el objetivo de un diseñador metaheurístico (analista), a saber, diseñar un algoritmo para el cual, aunque no se pueda ofrecer ninguna garantía sobre el peor de los casos, un cierto grado de confianza sobre el rendimiento del algoritmo "la mayoría de los tiempos "todavía se puede afirmar.

el verdadero desafío del experto en metaheurística no es solo medir objetivamente el algoritmo en términos de calidad de la solución y tiempo computacional sobre problemas para los cuales no se conoce una solución óptima (lo que, en sí mismo, puede ser una tarea difícil debido a la falta de cualquier punto de referencia), pero también para utilizar métodos y técnicas cuantitativos sólidos para afirmar la robustez del algoritmo en un amplio espectro de tipos de instancia.

* por un lado, es altamente deseable diseñar metaheurísticas de "propósito general", que no requieren conocimientos específicos del problema y se pueden aplicar fácilmente a un amplio espectro de clases de problemas.
* Por otro lado, se ha observado que las metaheurísticas de propósito general son superadas por los algoritmos híbridos, que generalmente son algoritmos que combinan técnicas de programación matemática con ideas basadas en metaheurística. Los algoritmos de esta clase están hechos a medida y específicamente diseñados para explotar las propiedades matemáticas del problema en cuestión.
* En general, un algoritmo hecho a la medida puede usarse solo para una clase específica de problemas y, a menudo, las ideas subyacentes no pueden extenderse fácilmente para operar en una clase diferente de problemas

Los algoritmos metaheuristicos en general parecen proporcionar un rendimiento variable dependiendo de la sensibilidad (habilidades, experiencia, ingenio, etc.) del diseñador en el ajuste fino algorítmico.

La variabilidad en los resultados de una metaheurística presenta al menos dos inconvenientes principales: (i) Por un lado, surge el problema de la reproducibilidad de los resultados, debido a, habilidades de implementación, uso especial de estructuras de datos y capacidad en la configuración de parámetros, etc. (ii) Por otro lado, se prevé un problema de maximización del rendimiento, el ajuste fino de los parámetros algorítmicos no solo es específico del problema, sino incluso específico de la instancia

Notas sobre aprendizaje.

El aprendizaje libre de sesgo es inútil (Mitchell (1980), Schaffer (1994) and Wolpert (1996))