한국 CAD/CAM학회 논문집
제 16권 제 3호 2011년 6월 pp. 236-245

순차 및 병렬처리 환경에서 효율적인 다분야통합최적설계 문제 해결 방법

이세정*

An Efficient Solution Method to MDO Problems in Sequential and Parallel Computing Environments

Se Jung Lee*

ABSTRACT

Many researchers have recently studied multi-level formulation strategies to solve the MDO problems and they basically distributed the coupling compatibilities across all disciplines, while single-level formulations concentrate all the controls at the system-level. In addition, approximation techniques became remedies for computationally expensive analyses and simulations. This paper studies comparisons of the MDO methods with respect to computing performance considering both conventional sequential and modern distributed/parallel processing environments. The comparisons show Individual Disciplinary Feasible (IDF) formulation is the most efficient for sequential processing and IDF with approximation (IDFa) is the most efficient for parallel processing. Results incorporating two popular design examples show this finding. The author suggests design engineers should firstly choose IDF formulation to solve MDO problems because of its simplicity of implementation and not-bad performance. A single drawback of IDF is requiring more memory for local design variables and coupling variables. Adding cheap memories can save engineers valuable time and effort for complicated multi-level formulations and let them free out of no solution headache of Multi-Disciplinary Analysis (MDA) of the Multi-Disciplinary Feasible (MDF) formulation.

Key words : Approximation, Distributed Computing, Individual Disciplinary Feasible (IDF), Multi-disciplinary Design Optimization (MDO), Parallel Computing

1. 서 론

이 연구의 초기에는 다양한 MDO(Multi-disciplinary Design Optimization) 방법들을 이용하여 주어진 설계점에서 자동적으로 가장 효율적인 방법을 선택하여 그에 맞추어 설계문제를 재설정하고 다시 설계를 진행하는 자동적응 형 MDO 해법을 구상하였다. 이를 위하여 여러 가지 MDO 방법들의 장, 단점을 분석할 필요가 있는데, 문헌 연구뿐 아니라 여러 종류의 설계문제에 대하여 성능 비교를 하였다.

이러한 비교를 하던 중 IDFa(Individual Disciplinary Feasible with approximation)이 순차처리뿐 아니라 병렬처리에서도 유난히 효율적이라서 이 방법을 개선하여 문제를 푸는 것을 고려하게 되었으며, 그 결과물이 본 논문이다.

설계 엔지니어들은 MDO 문제를 풀 때 MDF (Multi-Disciplinary Feasible) 방식으로 접근하게 된다. 왜냐하면 MDF는 설계문제의 재설정 없이 자연스럽게 생각할 수 있는 방법이기 때문이다. 그러나 MDF는 MDA(Multi-Disciplinary Analysis)의 해법을 전제로 하는데, 그 해법에는 FPI(Fixed-Point Iteration)과 Newton 방법이 대표적이다. 이들 방법들로 해에 수렴하기 위해서 만족해야 하는 조건들이 있는데, 이는 문제를 풀기 전에 미리 알기가 매우 어려우므로 일반적으로 이 방법들을 적용하기에는 부리가 따른다. 특히 MDF의 경우, 연성 강도가 강해질수록 MDA를

*종신회원, 서울시립대학교 기계정보공학과
- 논문투고일: 2010. 09. 16
- 논문수정일: 2011. 05. 12
- 심사완료일: 2011. 05. 12

해결하는 것이 어려워지고[3], 이 경우, 부서해석(Disciplinary Analysis)의 연산 횟수가 기하급수적으로 증가하거나 아예 풀지 못하는 경우가 발생한다.

MDF에 대한 대안으로 설계문제를 여러 개의 소규모 문제로 분해(decomposition)하는 다양한 MDO 문제 해결 방법들이 개발되었는데, 그에 대한 비교연구도 많이 진행되었다[3-6]. 이 연구들은 다양한 평가 기준으로 각 방법들을 비교하였는데 저자는 그 중에서도 해를 잘 찾는지, 찾는다면 얼마나 빨리 찾는 지와 같은 효율성에 관한 객관적으로 정량화 할 수 있는 기준에만 초점을 맞추어 연구를 진행하였다.

이 논문에서 성능평가를 위하여 두 가지 예제문제를 선택하였는데, 그 선택 기준은 다음의 요구조건을 만족하는 것으로 정하였다.

- 공통설계변수와 지역설계변수가 공존
- 각 부서의 해석은 블랙박스로 입력과 출력만 정의
- 기존 부서 해석 코드는 수정하지 못함
- 부서 해석은 최적화에 비하여 계산시간이 확실 긴 경우
- 연성이 있는 두 개 이상의 부서가 존재
- 병렬 또는 분산 처리 환경에서도 성능 평가 가능

이 연구에서 모든 알고리즘의 구현은 Matlab R2008을 이용하였으며 펜티엄 듀얼코어 컴퓨터에서 실행하였다. 또한 성능평가 기준으로 해를 구하는 데 걸리는 계산시간을 직접 측정하지 않고, 부서해석 횟수를 측정하였다. 왜냐하면 현실적인 제품설계 문제의 경우 해석의 연산시간이 최적화나 근사화에 비하여 압도적으로 길기 때문이다. 한편 최적화 알고리즘에 필요한 목적함수, 제한 조건 식의 미분 값들은 유한 차분 법으로 구하는 것을 기본으로 하였다. 물론 부서해석 코드에서 미분 값을 제공하는 경우도 있기는 하지만 어떤 경우에도 적용할 수 있도록 보수적으로 처리하였다.

최근 각 방법론들이 근사화 이론을 활용하여 알고리즘을 개선하고 있는데, CO(collaborative Optimization), CSSO(Concurrent Subspace Optimization), BLISS (Bi-Level Integrated System Synthesis) 등은 병렬처리를 적극 활용할 수 있다는 점에서 매우 고무적인 일이다. 이 연구에서는 각 방법들을 순차처리와 병렬처리의 환경하에서 비교하였다. 이 논문에서는 MDO방법론에 근사화를 활용하는 경우, 원래 방법론의 이름과 구별하기 위해서 각 방법론의 이름 끝에 근사화(approximation)의 약자 a를 붙여서 BLISSa, IDFa 등의 형식으로 표기하였다.

2. 수학적 예제를 통한 성능 비교

첫 예제는 수학적으로 구성된 문제로 많은 문헌[6-9]에서 다루었던 문제로 이를 통한 목적은 각 MDO 방법론에 대한 성능을 측정하여 기존 문헌들과의 경향을 비교, 분석하고 저자의 코딩의 적합성을 확인하고자 함이다. 각 방법론들을 공정하게 비교하기 위하여 특정 방법론에 국한된 코드 개선은 하지 않았으며 기본 알고리즘에 충실하였다.

이 설계문제를 최적설계로 정식화하면 다음 식으로 표현할 수 있으며

$$\begin{aligned}
&\text{Find} \quad x_1, x_2, x_3 \\
&\text{Min} \quad f = x_2^2 + x_3 + y_1 + e^{-y_2} \\
&\text{subject to} \quad g_1 = 1 - y_1/3.16 \le 0 \\
&\qquad\qquad g_2 = y_2/24 - 1 \le 0 \\
&\text{where} \quad y_1 = x_1^2 + x_2 + x_3 - 0.2y_2 \\
&\qquad\qquad y_2 = \sqrt{y_1} + x_1 + x_3
\end{aligned} \tag{1}$$

여기서 y_1, y_2는 연성상태변수, x_1, x_2, x_3 는 설계변수이다. 두 개의 부서로 구성되어 있으며 부서 1, 2 해석의 출력은 각각 y_1, y_2이다. 각 부서 해석의 입, 출력변수들을 다음 그림에 표시하였다. 부서해석 1의 출력은 부서해석 2로, 부서해석 2의 출력은 부서해석 1로 다시 입력되어 연성이 존재한다.

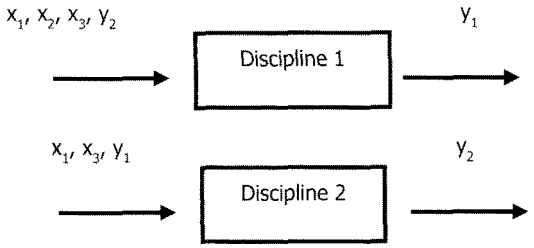


Fig. 1. Input/output variables of disciplinary analyses 1 and 2.

이 문제의 경우 비 연성 상태변수가 없고 부서해석기는 블랙박스로 취급하므로 해석은 입력이 정의되면 출력이 계산되는 것이라서 AAO(All At-Once) 나 IDF나 같은 형태가 된다. 또한 해석기가 블랙박스이므로 사용자가 코드를 연산 기(evaluator)로 변환할 수 없다고 가정하고 있으므로 IDF의 시스템 레벨의 적합

(Compatibility) 조건이 다음 식과 같아진다.

$$y_1 - \hat{y}_1 = y_2 - \hat{y}_2 = 0$$

where

$$\hat{y}_1 = x_1^2 + x_2 + x_3 - 0.2y_2$$
$$\hat{y}_2 = \sqrt{y_1} + x_1 + x_3 \quad (2)$$

여기서 입, 출력변수를 명확하게 구분하기 위해서 모자(hat)은 출력변수를 표시하며 모자가 없는 변수는 입력변수를 표시한다. 여기서 다루는 각 방법론의 구체적인 문제구성 방법 및 알고리즘은 기존 논문들에 있으므로 다시 기술하지 않았다. 이 문제의 초기 설계점은 $x = [1\ 5\ 2]$이고 최적 설계 점에서 목적 함수 값은 3.183이고, 최적화 수렴조건은 10E-4이다.

다중레벨 방법론의 경우 CSSO는 이 연구에서 고려하지 않았는데, 이 방법은 한 부서의 하층레벨의 문제구성에서 다른 부서의 목적함수나 제한조건에 대한 근사식을 사용하므로 한 부서의 설계자가 다른 부서의 설계문제를 이해하고 있어야 한다는 조건 때문에 제외하였다. 실제 산업체 설계부서의 경우 거의 불가능한 요구조건이다. CO의 경우 수렴성문제 때문에 개선된 알고리즘이 발표되었는데, 그 중 가장 최근 방법인 ECO[10,11]의 경우 다른 부서의 정보를 필요로 하므로 CSSO와 같은 이유로 본 논문에서 제외하였다. 이들 다중레벨 방법론들은 설계문제를 여러 개의 작은 문제로 분해할 때, 변수들의 문제간 불일치를 처리하는 방법이 다른 것인데, Tosserams[12]의 구분에 따르면 분산조화(distributed coordination) 방법이며 이에 반하여 IDF는 집중조화(centralized coordination) 개념으로 볼 수 있다.

다양한 방법들에 적용해 본 결과 최적화 과정에서 소요된 부서해석의 횟수를 Table 1과 같이 정리하였다. 기존 문헌[7]과 비교해 볼 때, 코딩 습관이나 개발 하드웨어 및 소프트웨어 환경이 다르므로 연산 횟수가 정확하게 일치하지는 않지만, 그 경향은 잘 일치하고 있으므로 코딩이나 컴퓨팅 환경의 하자는 없는 것으로 파악되었다. 이 비교연구에서 MDF의 경우 FPI와 Newton 방법을 모두 적용하였는데, FPI가 더 효율적이었다. 역시 IDF와 AAO는 거의 같은 결과를 보여 주었으며 CO가 가장 많은 연산을 하고서도 최적 해에 수렴하지 않았다.

결과 표에서는 순차처리의 경우 IDF가 가장 적은 연산 횟수로 해에 수렴하였다. 이에 근사화를 활용한 IDFa는 IDF보다도 더 적은 연산 횟수를 보였다. 병렬처리를 고려하면 BLISS와 IDFa가 우수한 결과를 주었다. IDFa는 완전히 새로운 문제 구성 방법은 아니지만 어떻게 기존 개념을 활용하는가에 따라 효과적인 문제 해결 방법이 될 수 있음을 시사하고 있으며 저자가 제안하는 자세한 알고리즘은 다음 절에 기술한다.

3. 근사화를 활용한 IDF 방법(IDFa)

앞 절의 결론은 IDF와 IDFa가 고려한 방법들 중 가장 효율적이었다는 것이다. 예제문제의 해결 과정에서 정의에 필요한 상태변수와 설계변수의 근사화만을 고려하므로, 시스템 레벨에서의 목적함수와 제한조건 식들은 모두 근사화된 식이었다. 이러한 근사화 식들과 IDF를 결합한 것이 IDFa이며, 본 예제문제에 대하여 가장 우수한 성능을 보였다. 여기서 특기할 사항은 IDF는 다중레벨이 아니라 단일 레벨이고 근사화하는 대상이 주로 상태변수이므로 레벨과 무관하다.

IDFa 알고리즘은 기본적으로 IDF구조에 연성상태변수들에 대한 근사화를 활용하는 방법이다. 저자는 이에 새로운 두 가지 개념을 도입하였는데, 첫 번째는

Table 1. Performance comparison of various MDO methods for the analytic problem (DA: Disciplinary Analysis, NF: Number of Function evaluations)

Method	Object function at optimum	NF of DA1	NF of DA2	Total NF
MDF with FPI	3.183	352	352	704
MDF with Newton	3.183	756	756	1512
IDF	3.183	54	54	108
AAO	3.183	60	60	120
CO	3.172	9456	12734	22190
BLISS	3.184	100	100	200
IDFa	3.183	45	30	75

매 시스템 사이클이 종료될 때, 시스템 해석, 즉 MDA를 하지 않고 근사화 품질을 확인하는 것이고, 두 번째는 설계 사이클이 진행해 갈 때 설계변수의 상, 하한 값들을 조정하는 알고리즘이다.

근사화 품질평가 방법은 매 사이클이 끝난 후, 시스템 설계변수 값을 이용하여 각 부서의 해석을 하면, 새로운 상태변수, $\hat{y}$가 계산이 된다. 이 값과 사이클 종료 시 상태변수에 해당하는 설계변수, y를 비교하면 근사화의 품질을 평가할 수 있다. 즉 다음 식을 만족하면 근사화 정도가 충분하다고 판단한다.

$$\|y - \hat{y}\| \le \varepsilon_1 \tag{3}$$

이렇게 하면 사이클 종료 시마다 해야 하는 MDA를 피할 수 있으므로 부시해석 횟수를 획기적으로 줄일 수 있다.

기존 BLISSa와 같은 근사최적화 방법들에서는 매 사이클마다 근사화의 품질을 개선하기 위하여 설계변수들의 상, 하한 값을 조정하여 영역을 감소시키는 신뢰구간(trust-region) 기법을 사용하였다. 이 기법은 MDA를 요구하는데, 이 문제의 해를 구하는 작업이 불가능하지는 않지만 매우 계산시간이 오래 걸릴 수 있으며 해를 못 구하는 경우도 발생한다. 해법이 FPI나 Newton류의 방법들인데 모두 한계가 있기 때문이다.

변수의 상, 하한 값을 조정하기 위하여, 사이클 종료 후, $\|y-\hat{y}\|$의 크기에 따라 각 설계변수의 영역, 즉 상한과 하한 값의 차이를 일정한 비율로 감소시키는 방법과 영역은 그대로 두고 샘플 수를 증가시키는 두 가지 방법을 시도하였다. 본 연구에서는 전자의 방법이 유효하였다. 후자의 방법은 직접 변수의 영역을 변화시키지는 않지만 근사화 정밀도를 향상시킬 수 있으므로 고려하였는데, 샘플링 방법에 종속적이므로 일반적으로 사용할 수는 없다는 사실을 배웠다. 전자의 방법도 감소비율을 정하는 데는 원칙이 아니라 시뮬레이션 경험이 필요하였다. 이 부분은 향후 추가 연구가 필요한 분야이다. 본 연구에서는 현재 설계 점을 항상 영역의 중앙에 위치시키고, 상태변수의 정밀도에 따라 양 끝에서 10%, 20%, 30%와 같이 일정 비율로 절단하여 영역을 축소시켜 진행하였다. IDFa의 절차를 정리하여 Fig. 2에 표시하였다.

근사화를 위한 샘플링 방법은 LHS(Latin Hypercube Sampling)를 선택하였다. LHS는 각 변수에 대하여 균일한 분포를 가지는 샘플을 작성하므로 근사모델을 만드는데 유리하였다. 다만 샘플링 할 때 난수를 생성

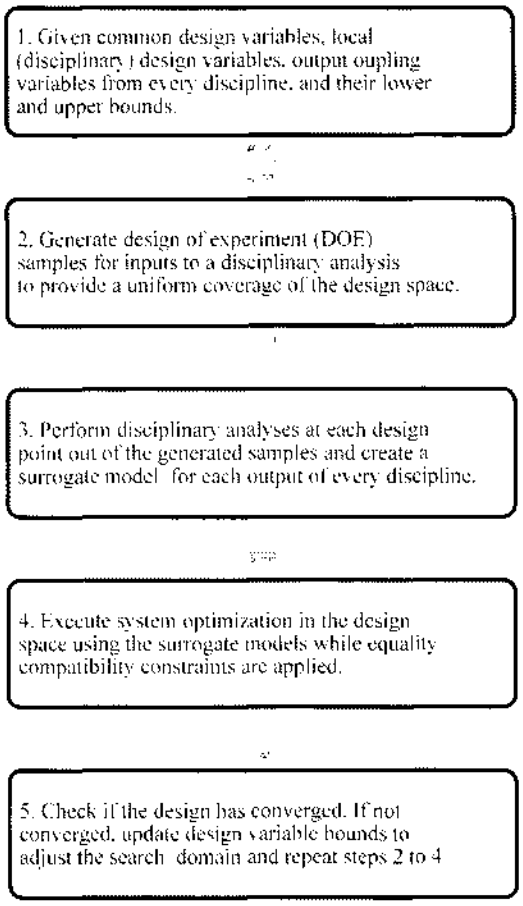


Fig. 2. Step-by-step procedure of IDFa.

시켜야 하는데, 근사모델의 반복 재현 성(repeatability)를 확보하려면 난수 발생기의 시드를 일정하게 정해야 할 필요가 있다. 그러나 샘플이 충분히 많지 않으면 발생된 난수에 따라 만들어지는 근사모델의 품질이 달라지는데 이에 따라 최적 해에 수렴하기도, 하지 않기도 하는 현상이 나타난다. BLISSa의 경우에 이 현상이 두드러졌다. 그래서 저자는 의도적으로 시드를 정하지 않고 충분히 많은 세트의 샘플에서 답이 나온 경우들 중에서 가장 좋은 결과를 수록하였다. IDFa의 경우는 시드에 상관없이 해가 구해졌다.

앞의 예제에 사용된 최적화 알고리즘은 Matlab 함수 FMINCON이고 최적화 수렴조건은 설계변수, 목적함수, 제한 조건 모두 10E-4로 하였으며 다른 파라메타는 모두 기본(default) 값을 사용하였다. 또한 LHS 샘플링을 위하여 LHSDESIGN 함수를 사용하였다. 근사화가 필요한 경우 크리깅을 구현하기 위하여 DACE[11]를 사용하였으며 regression 모델은 2차 다항

식인 regpoly2, correlation 모델은 Gaussian으로, 기타 파라메타는 기본값을 사용하였으며, 그 근사화 품질 평가는 상태변수 정밀도로 하고 최적설계 수렴조건과 동일한 10E-4을 적용하였다.

4. 확장형 문제에의 적용

IDFa의 성능을 다각적으로 평가하기 위해서 다중레벨 방법 중 우수한 BLISS, 순차처리에서 우수했던 IDF를 비교 대상으로 하여 다음 예제문제를 설정하였다. 문제는 식 (4)와 같이 정식화 되었으며, 이 문제에서는 부서의 수, 부서설계변수의 수, 연성상태변수의 수, 공통설계변수의 수 등을 자유롭게 변화시킬 수 있으므로, 그에 따른 성능의 변화를 조사할 수 있다.

$$\begin{aligned}
&\text{Find} \quad z, x \\
&\text{Min} \quad \sum_{i=1}^{Nz} z_i^2 + \sum_{i=1}^{Nd} \sum_{k=1}^{Ny} y_{i,k}^2 \\
&\text{subject to} \quad 1 - \frac{y_i}{c_i} \le 0 \quad \text{for} \quad i = 1, N_d \times N_y \\
&\text{where} \quad C_{y_i} y_i + C_{y_j} y_j = C_{x_i} x + C_{z_i} z \\
&\qquad \text{for the 'th discipline} \qquad (4)
\end{aligned}$$

이 문제는 기존 문헌[5,6]에서 사용하였던 문제로써, Nz 개수의 공통설계변수, 부서당 Nx 개수의 부서설계변수, 부서당 Ny 개수의 연성상태변수가 있다. 목적함수는 공통설계변수들의 제곱과 모든 연성상태변수들의 제곱의 합이고, 부등식구속조건은 모든 연성상태변수에 대하여 정의되었다. 마지막 줄의 상태방정식은 각 부서에 대하여 정의되는데, 공통 및 부서설계변수뿐만 아니라 해당부서 이외의 모든 다른 부서의 상태변수, yj에 의하여 연성관계가 형성되고 있다. Cyi, Cyj, Cxi, Czi는 모두 계수행렬이며, 그 크기는 각각 Ny*Ny, Ny*(Nd-1)*Ny, Ny*Nx, Ny*Nz이다. 모든 계수들은 난수를 발생시켜 변환하였으며, Cyj는 만들어진 행렬에서 한 행씩 건너서 음의 부호를 가지게 하였으며, 특정 변수의 숫자가 많아질 때는 그 개수만큼 스케일링하였다.

i번째 부서의 상태방정식의 입력과 출력은 Fig. 3과 같다. 입력은 공통설계변수, 부서설계변수, 그리고 다른 모든 부서로부터의 상태변수이고, 출력은 해당 부서의 상태변수가 된다.

이 예제에 사용된 최적화 알고리즘은 Matlab 함수 FMINCON이고 최적화 수렴조건은 설계변수, 목적함수, 제한 조건 모두 10E-8로 하였으며, 해에 수렴하지 않는 경우를 방지하기 위하여 최대 반복횟수를 999회로 제한하였다. 그 외의 다른 파라메타는 모두 기본값을 사용하였다. 근사화가 필요한 경우 LHS 샘플링을 위하여 LHSDESIGN함수를 사용하였고 크리깅을 구현하기 위하여 DACE[13]를 사용하였으며 regression 모델은 1차 다항식인 regpoly1, correlation 모델은 Gaussian으로, 기타 파라메타는 모두 기본값을 사용하였다. 또한, 상태변수의 수렴조건은 최적설계 수렴조건보다 작은 10E-9을 사용하였다.

설계변수들의 초기값은 모두 1로 하였고 각 변수들의 상, 하한 값들은 $-20 \le x \le 20, 0 \le y \le 10, 0 \le z \le 10$이다.

IDFa에서 샘플의 수는 2*(Nz+Ny*(Nd-1)+Nx+1)로 하였는데 선형 다항식으로 근사화 할 경우, 최소 샘플 수의 2배를 정하였다. 각 방법을 공정하게 비교하기 위해서는 샘플 수를 비슷한 수준으로 맞추어야 하는데, BLISSa에서는 샘플의 수를 2*(Nz+Ny*Nd+1)를 기준으로 하였다. 그러나 실제 시뮬레이션을 하면서 수렴을 위하여 샘플 수를 배가시켰다. 이에 대한 논의는 추후 예를 들어서 다시 하겠다.

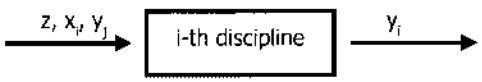


Fig. 3. Input and output variables of the i-th discipline.

확장 형 문제의 경우, 많은 시뮬레이션 결과 중 대표적인 것만을 본 논문에 수록하였다. 우선 설계변수나 상태변수의 개수가 일정할 때, 부서의 수가 변화하면 성능에 어떠한 영향이 있는지 조사해 본다. Table 2(a)는 컴퓨터환경이 순차처리만을 지원하는 경우이고, Table 2(b)는 연산 코어의 수가 충분히 많은 병렬처리를 고려한 경우이다. 표의 IDFa와 BLISSa의 경우 괄호 안의 숫자는 근사화에 사용된 샘플의 개수이다. 부서의 수가 2,4,8,16의 경우에는 선형 다항식 근사화에 필요한 수의 2배로 정했을 때, 두 방법 모두 잘 작동하였으나 32의 경우 BLISSa는 수렴이 잘 안되어 샘플 수를 배가 시켰다. 이는 부서의 수가 늘어나면서 BLISS 방법에서 요구하는 부서설계 문제의 해를 구하는 것이 어려워졌음을 의미한다. 설사 134개의 샘플로 해에 수렴했다 하더라도 연산 횟수는 IDFa에 비하여 약 20배에 달할 것으로 예측된다. 반면에 IDFa는 부서의 수가 증가하더라도 안정적으로 해에 수렴했다.

병렬처리의 경우, IDFa는 모두 1회 사이클에 해에 수렴하였다. 그럼에도 해석횟수가 2인 이유는 사이클 종료시점에서 근사함수의 품질을 평가하기 위하여 부서 해석을 1회 더 시행하였기 때문이다. 그리고 같은 문제에 대하여 BLISSa는 10번 이상 시행하여 가장 잘 나온 결과를 표에 담았다. IDF나 IDFa는 안정적으로 최적 해에 수렴하였다. Fig. 4는 Table 2를 그래프로 표현한 것이다. 부서의 수가 증가함에 따라 필요한 연산 횟수가 기하급수적으로 증가함을 알 수 있다. 그러나 Fig. 4(b)의 IDFa의 경우 부서의 수와 상관없이 2사이클 만에 해에 수렴했다. 이는 IDFa 방법이 부서의 수의 증감에 대해 상관없이 해에 수렴하고 있다는 것을 보여준다.

근사화 정밀도를 파악하기 위하여 최적 설계점에서 상태변수의 값에 대한 오차를 측정하였는데 부서 수가 증가함에 따라 IDFa의 경우 10E-14에서 10E-10, BLISSa의 경우 10E-13에서 10E-9의 범위 내에 있었다. 이는 근사화 정밀도가 최적설계 정밀도보다 높음을 의미한다.

근사화의 위 결과들을 종합하면 근사화를 도입한 최적화 방법이 IDF에 비하여 우수함을 확인할 수 있다. 다만 순차처리의 경우 BLISSa는 IDF에 비하여

Table 2. Numbers of analyses as Nxi = 2, Nyi = 2, Nz = 2, and Nd varies

(a) Sequential	2	4	8	16	32
IDF	44	304	840	3216	12576
IDFa	30(14)	92(22)	312(38)	1136(70)	4320(134)
BLISSa	216(14)	1076(22)	3873(38)	21262(70)	187532(268)

(b) Parallel	2	4	8	16	32
IDF	22	76	105	201	393
IDFa	2	2	2	2	2
BLISSa	10	16	22	35	121

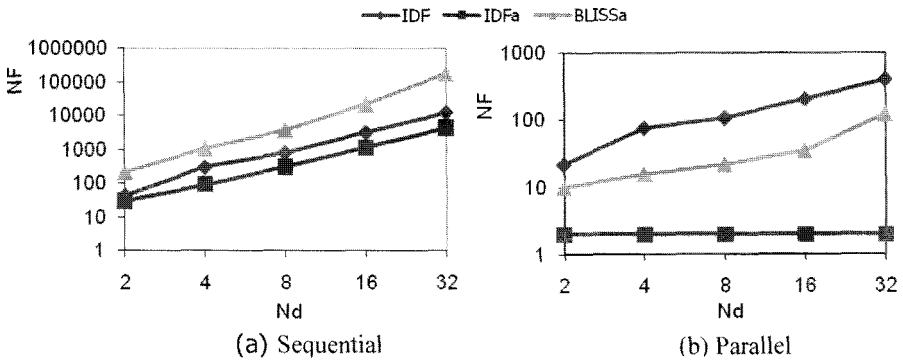


Fig. 4. Numbers of analyses as Nxi = 2, Nyi = 2, Nz = 2, and Nd varies.

Table 3. Numbers of analyses as Nd = 2, Nyi = 2, Nz = 2, and Nx varies

(a) Sequential	4	8	16	32	64	128	256
IDF	66	90	138	234	426	810	1578
IDFa	30	38	54	86	150	278	534
BLISSa	198	310	534	1016	1878	4255	8029

(b) Parallel	4	8	16	32	64	128	256
IDF	33	45	69	117	213	405	789
IDFa	2	2	2	2	2	2	2
BLISSa	7	11	19	52	67	391	646

우수함을 보이지 못하였다.

Table 3에서 부서설계변수의 증가에 따라 각 방법론의 성능을 비교하였으며, 순차처리의 경우가 Table 3(a)이고 병렬처리의 경우가 Table 3(b)이다. 이 경우는 원래 BLISS의 장점이 부서설계변수가 많은 경우에 효율적으로 작동되도록 고안되었으므로 IDFa와 성능을 비교하기 위한 것이다.

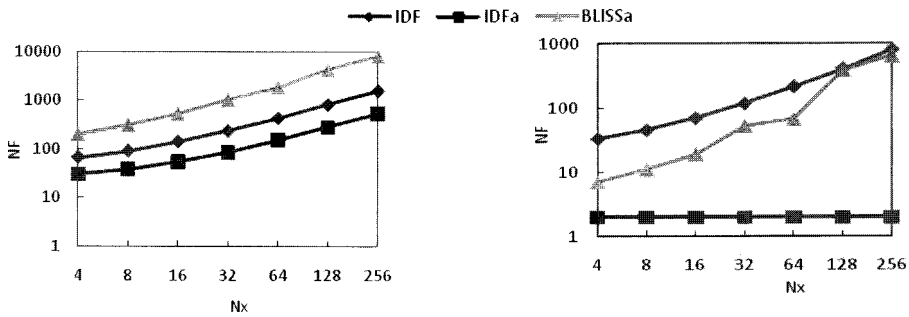
순차처리의 경우 예상한대로 BLISSa는 IDF나 IDFa보다 열등하다. 병렬처리의 경우는 BLISSa가 IDF보다는 우수하나 IDFa에 비해서는 매우 열등하다. 이 결과가 보여주는 것은 부서설계변수가 증가하더라도 BLISSa보다는 IDFa가 효율적이라는 점이다. Fig. 5에 각 방법에서의 부서설계변수의 수에 따른 연산 횟수의 변화를 표현하였으며 전반적인 경향은 Fig. 4와 비슷하다. 다만 IDFa의 경우 Fig. 4와 마찬가지로 2회 사이클에 해에 수렴했다.

확장 형 예제의 마지막 시도는 연성상태변수의 수를 변화시키는 것이다. 연성상태변수의 수가 증가하면 연성 상태방정식의 수가 증가하고 한 부서와 그 외 다른 모든 부서와의 연성이 동시에 증가하게 된다. 결과는 Table 4에 정리하였으며 Fig.6에 그래프로 표현하였다.

순차처리의 경우 IDFa, BLISSa, IDF의 순으로 부서해석 횟수가 많았다. 병렬처리의 경우는 IDF는 각 부서의 해석을 병렬로 처리하는 것 이외에는 병렬처리가 가능하지 않아서 부서의 수가 2개이면 순차 처리시 해석횟수의 반이 된다. IDFa와 BLISSa는 모두 한 시스템 사이클에 해에 수렴하였다. 이 경우에도 BLISSa는 여러 번 시행 중 성공한 경우만을 표에 적었다. 하지만 두 방법 모두 샘플의 개수를 기저함수를 선형함수로 할 때 필요한 최소숫자로 정하였는데, BLISSa의 Ny = 128, 256의 경우에는 1.5배 또는 2배의 수를 요구하였다. 256의 경우에는 알고리즘이 잘 수렴하지 않아서 최적화 수렴조건을 타 경우보다 10배 약화시켰다.

요약해 보면 부서의 수, 설계변수의 개수, 상태변수의 개수 등을 변화시켜본 결과, 순차처리이던 병렬처리이던 IDFa가 가장 우수한 성능을 보여주었다. BLISSa처럼 하위레벨의 최적화문제의 근을 근사화하는 것보다는 IDFa와 같이 연성변수를 근사화 하는 것이 효과적이라는 것을 확인할 수 있었다.

순차처리의 경우 부서의 수, 부서설계변수의 수, 연성상태변수의 수를 각각 8에서 32로 4배 증가시킬 때, 연산 횟수를 Table 2, 3, 4에서 찾아보면 IDFa는



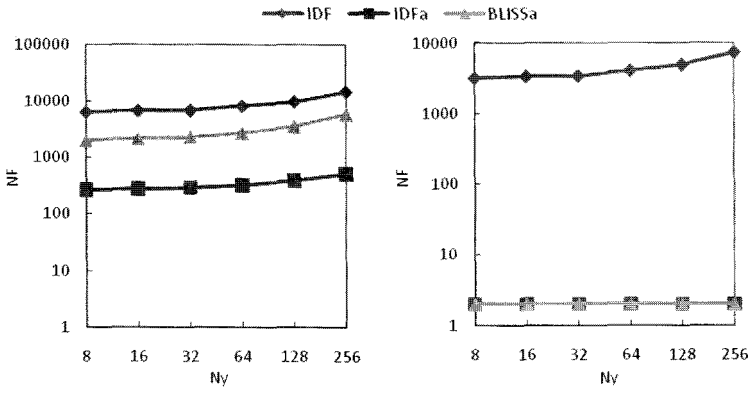
(a) Sequential (b) Parallel

Fig. 5. Numbers of analyses as Nd = 2, Nyi = 2, Nz = 2, and Nx varies.

Table 4. Numbers of analyses as Nd = 2, Nx = 4, Nz = 2, and Ny varies

(a) Sequential	8	16	32	64	128	256
IDF	6486	6854	6932	8280	9920	14876
IDFa	272	280	296	328	392	520
BLISSa	2028	2203	2316	2728	3600	5932

(b) Parallel	8	16	32	64	128	256
IDF	3243	3427	3466	4140	4960	7438
IDFa	2	2	2	2	2	2
BLISSa	2	2	2	2	2	2



(a) Sequential (b) Parallel

Fig. 6. Numbers of analyses as Nd = 2, Nx = 4, Nz = 2, and Ny varies.

13.8, 2.3, 3.3배, BLISSa는 48.4, 3.3, 1.1배가 됨을 알 수 있다. 부서나 변수의 수가 증가함에 따라 IDFa가 덜 민감하여 더 빠르게 해를 구하였음을 보여준다. 또한 부서 수의 변화가 변수들에 비하여 해를 구하는 데 필요한 연산 횟수에 더 큰 영향을 준다는 사실을 알 수 있다.

BLISSa는 샘플의 개수, 샘플링 시 난수의 시드, 가중치의 상, 하한 값 등에 매우 민감한 반응을 보였다. 표의 시뮬레이션 결과들은 수많은 시행 중 가장 잘 나온 결과를 담은 것이다. 실제 이 방법을 사용하려면 아주 많은 시행착오가 필요함을 배웠으며, 가중치의 상, 하한 값에 대해서는 설계자가 아무런 정보나 경험이 없으면 크게 줄 수 밖에 없는데 이는 근사화 품질을 저하시켜 최적 해의 수렴 속도가 떨어질 수 있다. Bliss는 각 부서의 문제가 설계문제이므로 복성 샘플의 경우 최적해가 존재하지 않는 경우가 발생할 수 있으므로 이에 대한 조치가 필요하다. 가장 손쉬운 방법은 샘플을 여유 있게 가지고, 해를 찾은 경우만을 근사화에 이용하는 것이다. 이 방법은 필요 이상의 샘플을 만들어야 하므로 부서해석의 횟수를 증가시킨다. BLISSa에서는 샘플에 따라 근사화 품질이 다르고 이에 따라 최적화 거동이 심하게 변화하므로 병렬처리가 허용하는 범위 내에서 가장 많이 샘플링하여 여러 개의 조합을 만들어 근사화 하는 방법도 좋은 접근방법이 될 수 있다.

이 예제의 결과를 종합해 보면, 병렬처리를 고려한다면 근사화를 도입한 최적화 방법이 기존 IDF보다 우수함이 확실하다. 순차처리의 경우에도 근사화 최적화 방법이 우수하였지만, BLISSa가 부서의 수가 증가하는 경우에는 가장 열등하였다.

5. 결론 및 향후 연구방향

선택한 예제에 대하여 결과를 정리해 보면, 순차처리에서는 IDF와 IDFa가 우수한 것으로, 병렬처리 환경에서는 IDFa가 우수함을 확인하였다. 부서의 수, 설계변수의 개수, 상태변수의 개수 등을 변화시켜본 결과, 순차처리이던 병렬처리이던 IDFa가 가장 우수한 성능을 보여주었다. BLISSa처럼 하위레벨의 최적화 문제의 근을 근사화하는 것보다는 IDFa와 같이 연성변수를 근사화 하는 것이 효과적이라는 것을 확인할 수 있었다. 순차처리의 경우 부서의 수, 부서설계변수의 수, 연성상태변수의 수를 부서나 변수의 수가 증가함에 따라 IDFa가 덜 민감하여 더 빠르게 해를 구하였음을 보여준다. 또한 부서 수의 변화가 변수들에 비하여 해를 구하는 데 필요한 연산 횟수에 더 큰 영향을 준다는 사실을 알 수 있다.

병렬처리환경에 적합하도록 개발된 근사화를 이용한 CO나 BLISS의 경우에 최적화문제가 중첩된 구조(Nested architecture)를 가지므로 IDFa에 비하여 연산 횟수가 많이 필요하게 된다. IDFa 의 단점은 연성변수 및 부서설계변수를 모두 시스템 레벨 최적화 문제에서 다루게 되므로 최적화문제의 개수는 한 개이지만, 그 규모가 커지는 점이다. 늘어난 변수의 수만큼 메모리가 더 필요하다는 점 이외에 최적화 문제의 복잡성은 저자가 사용한 예제에서는 문제가 부각되지 않았다.

근사최적화에서는 근사화의 품질이 중요한 요소인데 이 연구에서는 품질보다는 각 방법론의 구조에 대한 연구이므로 시스템 사이클이 수 회 또는 수십 회 이내에 해가 구해지도록 근사화 정밀도를 조절하였

다. 실제로 시스템 사이클이 많아지면 아무리 병렬처리를 하여도 부서 해석을 순차적으로 처리할 수밖에 없으므로 병렬처리의 장점을 최대한 활용할 수가 없다. 이에 비하여 IDFa는 어떠한 경우에도 병렬처리의 장점을 최대한 활용할 수 있었으며 컴퓨터 연산 코어의 개수가 많으면 많을수록 빨리 해를 구할 수 있다.

이 연구에서는 근사최적화의 과정에서 신뢰구간(trust-region) 기법을 사용하지 않았다. 왜냐하면 이 기법은 시스템 레벨의 한 사이클이 종료될 때마다 시스템 해석을 하여 목적함수의 변화 량을 계산하여 설계변수의 이동거리 제한 값을 정의하는데, MDA가 수행되어야 하므로 여기서 수렴하지 않을 수도 있고, 수렴하더라도 다시 많은 해석 연산 횟수가 필요하게 된다. 저자는 MDA를 수행하지 않고 근사화의 품질을 측정할 수 있는 개념을 제안하였으며, 이는 시스템 레벨의 최적화 과정의 수렴 성능을 향상시킬 수 있었다.

본 논문에 수록된 예제들에서 사용된 근사화 방법은 크리깅이다. 물론 저자는 여러 가지 다양한 근사화 방법들을 시도하였지만 주어진 예제들에서 안정적인 해를 주었던 크리깅을 선택하였다. 그러나 근사화 방법을 선택할 때는 최적화 방법론, 설계 문제 및 해석의 특성 등을 모두 고려하여야 할 것이다. 이 분야는 향후 연구 분야로 남겨져 있다. 다만, 한 가지 근사화 기법을 이용하는 것보다는 조합적인 방법, 즉 앙상블 모델링 등의 기법에 대한 연구가 최근 활발하게 진행되고 있어 우수한 결과들이 발표될 것으로 기대한다.

감사의 글

이 논문은 2009년도 서울시립대학교 연구년교수 연구비에 의하여 연구되었음.

기호 설명

Cyi, Cyj, Cxi, Czi : randomly generated coefficient matrices with sizes of Ny * Ny, Ny * (Nd-1) * Ny, Ny * Nx , Ny * Nz, respectively

Nd: number of disciplines

NF: number of function evaluations or analyses

Nx: number of disciplinary design variables per discipline

Nxi: number of disciplinary design variables for the i-th discipline

Ny: number of state variables per discipline

Nyi: number of state variables of the i-th discipline

Nz: number of common design variables

X: disciplinary design variable

Y: state variable

Z: common design variable

ε_y: convergence tolerance for state variables

참고문헌

1. 이장효, 이세정, "재구성이 가능한 다분야통합최적설계 프레임워의 개발," 한국 CAD/CAM학회논문집, 제14권, 제3호, pp. 207-216, 2009.
2. Allison, J. T., Kokkolaras M. and Papalambros P. Y., "On Selecting Single-Level Formulations for Complex System Design Optimization," *Trans. of ASME*, Vol. 129, pp. 898-906, 2007.
3. 이세정, 안문렬, "다분야통합최적설계 방법론의 병렬처리 성능 분석," 대한기계학회논문집A권, 제31권, 제12호, pp. 1150-1156, 2007.
4. Yi, S. I., Shin, J. K. and Park, G. J., "Comparison of MDO Methods with Mathematical Examples", *Struct. Multidisc. Optim.*, Vol. 35, pp. 391-402, 2008.
5. Tedford, N. P. and Martins, J. R. R. A., "Benchmarking Multidisciplinary Design Optimization Algorithms," *Optimization Engineering*, DOI 10.1007/s11081-009-9082-6, online 2009.
6. Marriage, C., Automatic Implementation of Multidisciplinary Design Optimization Architectures Using piMDO, MS Thesis, Aerospace Engineering, Univ. of Toronto, 2008.
7. Perez, R. E., Liu, H. H. T. and Behdinan K., "Evaluation of Multidisciplinary Optimization Approaches for Aircraft Conceptual Design," 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA Paper 2004-4537, Aug.30-Sep.1, Albany, N.Y., 2004.
8. Sellar, R. S., Batill, S. M. and Renaud, J. E., "Response Surface Based, Concurrent Subspace Optimization for Multidisciplinary System Design," Proc. of the 34th AIAA Aerospace Sciences Meeting and Exhibit (Reno, NV). AIAA, Reston, VA, 1996-0714.
9. Tedford, N., Comparison of MDO Architectures within a Universal Framework, MS Thesis, Aerospace Engineering, Univ. of Toronto, 2006.
10. Roth, B. and Kroo, I., "Enhanced Collaborative Optimization," 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, BC, September 6-8, 2008.
11. Roth, B. and Kroo, I., "Enhanced Collaborative Optimization: A Decomposition-based Method for Multidisciplinary Design," Proc. of ASME IDETC/

CIE, DETC2008-50038, Brooklyn, New York, Aug. 3-6, 2008.
12. Tosserams, S., Etman, L. F. P. and Rooda, J. E., "A Classication of Methods for Distributed System Optimization Based on Formulation Structure," *Struct. Multidisc. Optim.*, DOI 10.1007/s00158-008-0347-z, online, 2008.
13. Lophaven, S. N., Nielsen, H. B., Sondergaard, J., DACE - A MATLAB Kriging Toolbox Technical University of Denmark, Technical Report IMM-TR2002-12, 2002.

이 세 정

1976년~1980년 서울대학교 기계공학
1980년~1982년 한국과학기술원 기계공학 석사
1982년~1985년 현대건설
1986년~1989년 Pennsylvania State University 기계공학 박사
1990년 삼성중공업
1991년~1992년 한국생산성본부
2000년~2001년 Georgia Institute of Technology, Aerospace Engineering, 교환교수
2009년~2010년 Emory University, Biomedical Engineering, 교환교수
1993년~현재 서울시립대학교 기계정보공학과 교수