A constraint consensus memetic algorithm for solving constrained optimization problems

Noha M. Hamza*, Ruhul A. Sarker, Daryl L. Essam, Kalyanmoy Deb and Saber M. Elsayed

*aSchool of Engineering and Information Technology, University of New South Wales, Canberra, Australia; bBEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, Michigan, USA

(Received 1 May 2012; accepted 27 August 2013)

Constraint handling is an important aspect of evolutionary constrained optimization. Currently, the mechanism used for constraint handling with evolutionary algorithms mainly assists the selection process, but not the actual search process. In this article, first a genetic algorithm is combined with a class of search methods, known as constraint consensus methods, that assist infeasible individuals to move towards the feasible region. This approach is also integrated with a memetic algorithm. The proposed algorithm is tested and analysed by solving two sets of standard benchmark problems, and the results are compared with other state-of-the-art algorithms. The comparisons show that the proposed algorithm outperforms other similar algorithms. The algorithm has also been applied to solve a practical economic load dispatch problem, where it also shows superior performance over other algorithms.

Keywords: constrained optimization, constraint consensus, genetic algorithms, memetic algorithms, economic load dispatch

List of abbreviations

| Abbreviation | Meaning |
|--------------|---------|
| COPs         | Constrained optimization problems |
| GA           | Genetic algorithm |
| PSO          | Particle swarm optimization |
| DE           | Differential evolution |
| ES           | Evolutionary strategies |
| CC           | Constraint consensus |
| DBmax        | Direction based maximum |
| FDfar        | Feasibility distance far |
| CSP          | Constraint satisfaction problems |
| CCMA         | Constraint consensus memetic algorithm |
| ELDP         | Economic load dispatch problem |
| GA-MPC       | GA with multi-parent crossover |
| GA-MPC-CC    | GA-MPC combined with basic CC |
| GA-MPC-FDfar | GA-MPC combined with FDfar |

*Corresponding author. Email: noha.hamza@student.adfa.edu.au

© 2013 Taylor & Francis
1. Introduction

Solving constrained optimization problems (COPs) (Wang and Cai 2012a, 2012b) has been an important research topic in the optimization, operation research and computer science domains. In solving COPs, an acceptable solution means that it must be feasible and either optimal or near optimal. An optimal solution in a COP may exist either on the boundary of, or well inside, the feasible space. In this case, if an infeasible solution vector is assisted to move towards a feasible region, it will have a higher chance of quickly reaching the optimality or near optimality in the search process.

Over the past few decades, a variety of population-based algorithms has been introduced for solving COPs, such as the genetic algorithm (GA) (Goldberg 1989), particle swarm optimization (PSO) (Kennedy and Eberhart 1995), differential evolution (DE) (Storn and Price 1995; Wang and Cai 2011), evolutionary strategy (ES) (Rechenberg 1973; Wang and Cai 2011) and evolutionary programming (Fogel, Owens, and Walsh 1966). In these algorithms, different mechanisms have been applied to either reduce or remove the constraint violation of infeasible individuals. In the evolutionary computation domain, these methods are widely known as constraint handling techniques. Some of the popular methods are: feasibility rules (Deb 2000), the \( \varepsilon \)-constrained method (Takahama and Sakai 2010), stochastic ranking (Runarsson and Xin 2000), multi-objective concepts (Mezura-Montes and Coello 2005; Cai and Wang 2006; Wang et al. 2007a, 2007b), novel penalty functions (Jianhua et al. 2007) and novel special operators (Leguizamon and Coello 2009). These techniques assist the ranking and selection process. However, once the ranking and selection are done, they have no direct influence in the actual search process (that involves crossover, mutation and/or local search method). This means that, after selection of parents, there is no difference between the actual search process in unconstrained and constrained optimization problems.

In the literature on optimization, there are specialized methods (such as projection algorithms) that deal with the feasibility issues as a search process (Censor and Zenios 1997). The projection methods basically guide the infeasible solutions to move towards the feasible space. There are various projection methods, including orthogonal, sequential, simultaneous, component averaging and constraint consensus (CC). The orthogonal projection of an infeasible point, so as to satisfy a constraint, is defined as the closest feasible point (Xiao et al. 2003). In the sequential projection algorithm, the current infeasible point is updated for each violated constraint in turn (Chinneck 2004). In simultaneous projection algorithms, the current infeasible point is updated for all the violated constraints simultaneously, and then the final movement to feasibility is calculated. In gradient projection algorithms, to move from a current infeasible point to a feasible point, the constraint violation and the gradient of that point are required (Chinneck 2004). The component averaging projection algorithms consider the fact that the calculation for the final movement vector should contain only the constraints that contain particular variables, as not all of the constraints contain all of the variables (Censor, Gordon, and Gordon 2001). The projection algorithms have been proved to converge for feasibility problems when the constraints are inequalities with a convex region (Censor and Zenios 1997). The component-averaging projection algorithms have also been proven to converge in cases that simultaneously include linear equality constraints and convexity (Censor, Gordon, and Gordon 2001). CC methods (Censor, Gordon, and Gordon 2001) are a variant of the simultaneous component-averaging gradient-projection algorithms that provide approximate solutions to feasibility problems which include nonlinear and non-convex
The basic CC method is widely used in practice because it can reach a feasible region quickly, it is simple as it does not require any line search or matrix inversions, and the method has been shown to be robust, even in the presence of non-convex constraints in infeasibility. There are several variations of the CC methods, such as the direction-based maximum (DBmax) algorithm and the feasibility distance far (FDfar) algorithm. Empirical testing has shown the superiority of DBmax followed by FDfar (Ibrahim and Chinneck 2008). Note that the constraint satisfaction approaches that are applied to the constraint satisfaction problem (CSP) (Kumar 1992) are based on different principles, and the overlap between CSP and COP application domains is very small. These approaches have also been applied in conjunction with GA for solving CSPs (Brailsford, Potts, and Smith 1999).

In this research, the real-valued constrained optimization problems are considered, so a real-valued encoding is used to represent the variables. The main contribution of this research is the development of a new GA-based algorithm, for constrained optimization, that has the capability to assist the movement of infeasible individuals towards the feasible region, using a mix of three CC methods (the basic CC, DBmax and FDfar). The algorithm is named the constraint consensus memetic algorithm (CCMA). The sequential quadratic programming (SQP) technique has also been applied as the local search method in CCMA.

To analyse the contributions of the CC methods and the local search technique to CCMA, a study has been conducted using a GA in combination with these CC methods. To do this, all the variants of the algorithms start with a randomly generated population. Based on the level of constraint violations, the individuals are then divided into two groups (feasible and infeasible). Next, the CC methods are applied to some of the infeasible individuals until the stopping criteria are met. The resulting individuals are then merged with all of the other individuals and then the GA procedure is applied. For the variant of the mix of CC methods, all three methods are used in one algorithm framework by assigning one method to each individual, and the number of infeasible solutions that will be assigned to each method is adaptively updated. All algorithms have been analysed by solving 24 specialized test problems (Liang et al. 2005). CCMA has also been applied to solve another well-known set of test problems with 36 problem instances (Mallipeddi and Suganthan 2010b). The proposed CCMA has been compared with other state-of-the-art algorithms. The results show that, in most cases, the algorithm can achieve either better or equal solutions with significantly reduced computational time. The algorithm has further been applied for solving an economic load dispatch problem (ELDP) that is an important scheduling application for effective power system operation and management (Happ 1977; Chowdhury and Rahman 1990; Vlachogiannis and Lee 2009). The mathematical representation of this problem forms a complex nonlinear constrained optimization model. The experimental results from different instances of this problem show that the proposed algorithm is superior to other state-of-the-art algorithms.

This article is organized as follows. After the Introduction, Section 2 describes the CC methods used with GA in this article. Section 3 presents the GA with an overview of its parameters. Section 4 demonstrates the design of the proposed methods. The experimental results and the analysis of those results are presented in Section 5. In Section 6, the ELDP is described and its results are then analysed. Finally, the conclusions are given in Section 7.

2. Constraint consensus methods

A considerable number of projection algorithms has been proposed to solve feasibility problems with a set of convex constraints. However, the CC methods (Censor and Zenios 1997; Chinneck 2008; Ibrahim and Chinneck 2008) use an inexpensive heuristic for solving feasibility problems with nonlinear and non-convex constraints. Here, the main idea is to assist an infeasible point to move towards the feasible region, by making a consensus among the currently violated constraints,
The process of CC starts with the construction of a feasibility vector for each violated constraint at a selected infeasible point. The feasibility vector estimates the move from the current location to the closest feasible point for each violated constraint. The calculation of the feasibility vectors is exact for linear constraints, but it is a linear estimation (known as a linear feasibility vector) for nonlinear constraints. The linear feasibility vector moves the infeasible point in a parallel direction to the gradient of the violated constraint. The step size of the movement is determined by using a first order Taylor series expansion of the constraint function around the current point (Chinneck 2004). The linear feasibility vector is calculated according to the following equation for both linear and nonlinear constraints, in which this equation is derived from the first order Taylor series (Ibrahim and Chinneck 2008; Hamza et al. 2011):

$$ (x_{k+1} - x_k) = \frac{-g_i(x_k)}{\nabla g_i(x_k)^T} \nabla g_i(x_k)^T $$

(1)

where $g_i(x_k) : R^n \rightarrow R$ is the $i$th constraint, $\nabla g_i(x_k)^T$ is the transposition of its gradient, and $\nabla g_i(x_k)^T$ is its length; $x_k$ is the current infeasible point at the $k$th generation, and $x_{k+1}$ is the estimated feasible point for each violated constraint at the $(k + 1)$th generation.

The feasibility vectors for all of the violated constraints are combined into a consensus vector which is then used to update the current position by applying the proposed movement. The CC procedures are repeated until certain stopping conditions are met (Chinneck 2004), such as: (1) the length of every feasibility vector is less than the feasibility distance tolerance ($\alpha$) (e.g., $10^{-6}$); (2) the length of the consensus vector is less than the movement tolerance, $\beta$, which means that the consensus vector is too short to make good progress; or (3) the number of generations, $\mu$, exceeds a predefined value. The details of these conditions can be found in Ibrahim and Chinneck (2008). An unsuccessful termination of the algorithm may happen as a result of the last two conditions, and those conditions are generated when the convergence is too slow. The algorithm may also exit unsuccessfully if the length of the consensus vectors across generations is not decreasing quickly enough (Jain and Elmaraghy 1997).

There are several variations of the CC method. The difference among these variations is the way in which the consensus vector is constructed. The basic CC treats all eligible feasibility vectors equally and the proposed movement is created by averaging the non-zero components of the feasibility vectors. DBmax decides the direction of the movement based on the size of the largest proposed movement in each of the positive or negative directions. Then, the largest movement is set as the consensus vector. FDfar sets the consensus vector based on the length of each feasibility vector, in which it selects the longest feasibility vector to be the consensus vector with the goal of generating the most rapid movement towards feasibility. Any components that do not appear in the selected longest feasibility vector are set by averaging, as in the basic CC scheme.

In the highly nonlinear constrained problems, the linear feasibility vector has two drawbacks: it is an approximation and the calculation of gradient is expensive. To overcome these problems, an augmentation technique was proposed as an alternative way to build the feasibility vector (Smith 2011). The calculation of the augmented feasibility vector uses a predictor–corrector approach, to make an adjustment to the length of the previous consensus vector (i.e., predictor) without any changes in the search direction. This calculation is like the Secant method for root finding (Tjalling 1995). The augmentation technique increases the speed and the success rate of the CC methods by increasing the accuracy of the feasibility vector for nonlinear constraints. This technique works as follows: (1) in the predictor step, the feasibility vector is calculated according to Equation (2), and the consensus vector is then constructed to determine the search direction; and (2) in the
correction step, for every nonlinear constraint, an augmented feasibility vector (Smith 2011) is used instead of gradient. The step size is determined by using the values of the constraint functions at the current and previous generations. When using the augmentation technique, finding a point with a smaller constraint violation consumes fewer calculations in comparison to when using the linear feasibility vectors. The augmented feasibility vector is calculated according to the following equation:

\[(x_{k+2} - x_{k+1}) = \frac{-g_i(x_{k+1})}{g_i(x_{k+1}) - g_i(x_k)}(x_{k+1} - x_k)\]  

where \(g_i(x_{k+1}): \mathbb{R}^n \rightarrow \mathbb{R}\) is the \(i\)th constraint, \((x_{k+1} - x_k)\) is the previous consensus vector, \(x_k\) is the initial infeasible point at the \(k\)th iteration, \(x_{k+1}\) is the current point at the \((k + 1)\)th iteration, and \(x_{k+2}\) is the new point at the \((k + 2)\)th iteration.

After calculating the augmented feasibility vector, the consensus vector is constructed using one of the CC methods. In this article, the augmentation technique is used with each CC method.

The weakness of the basic CC method is that the method of combining individual feasibility vectors can be fooled by special circumstances, especially when a set of feasibility vectors pulling in one direction is counterbalanced by another set pulling in the opposite direction, thereby resulting in a very short consensus vector which leads to a slower convergence (Ibrahim and Chinneck 2008). Also, the consensus vector may be too short if the violated constraints seem to be parallel, so the feasibility vectors will be almost opposites of each other. Moreover, the weakness of FDfar is that it may repair the most violated constraint by cycling between the constraints (Smith 2011). The effect of such problems is much less significant when CC is used with an evolutionary algorithm.

3. Genetic algorithm

In this article, a genetic algorithm with multi-parent crossover (GA-MPC) has been considered that outperforms all other existing evolutionary algorithms in solving benchmark COPs (Elsayed, Sarker, and Essam 2011a), as well as different real-world applications (Elsayed, Sarker, and Essam 2011b).

In GA-MPC, a population of individuals is randomly generated and the best \(k\) individuals are saved into an archive pool (based on their constrained violations and/or fitness functions). According to Elsayed, Sarker, and Essam (2011b), the value \(k = PS/2\) (where \(PS\) is the population size) is used. A tournament selection then takes place and the selected vectors are saved in a selection pool. The crossover operator is applied to generate new offspring in which three parents are selected based on a selection rule and then the selected individuals are checked for duplication. However, if any duplication is found, one of them is replaced with a randomly selected individual from the selection pool. For the purpose of the crossover operation, the three selected individuals are ranked from the best \((x_1)\) to the worst \((x_3)\) based on their constrained violations and/or fitness functions. Three new offspring are then generated from three individuals as follows:

\[o_1 = x_1 + \tilde{z} \times (x_2 - x_3)\]  
\[o_2 = x_2 + \tilde{z} \times (x_3 - x_1)\]  
\[o_3 = x_3 + \tilde{z} \times (x_1 - x_2)\]  

The parameter \(\tilde{z}\) is a normally distributed random number with a mean value \(p\) and a standard deviation \(\sigma\). After the three individuals are generated, to improve the diversity in the new population, a randomized operator is then applied on each of the generated offspring. This is done by choosing, with a given probability, one of the variables (it is assumed that there are \(n\) variables/dimensions)
from the considered offspring, and replacing it with the corresponding one of a random individual from the archive pool. This step acts as a mutation process. Finally, the mutated offspring are merged with the individuals in the archive pool, by selecting the best group of individuals for the next generation.

4. A mix of constraint consensus methods with genetic algorithm

In this article, the above GA is combined with a mix of CC methods. The process for mixing CC methods is described below.

4.1. Genetic algorithm with multi-parent crossover with a mix of constraint consensus methods

In the proposed algorithm, an initial population with a size $PS$ is generated. The $PS$ individuals are then divided into two groups: the feasible individuals group and the infeasible individuals group. The infeasible solutions are then sorted, in ascending order, based on their constraint violation. The best $P$ infeasible solutions, i.e. individuals with lower constraint violations, are divided into three groups ($group_i$, $i = 1, 2$ and $3$) of equal size ($n_i$, $i = 1, 2$ and $3$), in which the individuals in $group_1$ are better than the individuals in group $group_2$ and $group_3$, and the individuals in $group_2$ are better than the individuals in $group_3$. The basic CC method is applied to the individuals in $group_1$, FDfar is applied to the individuals in $group_2$ and DBmax to the last group of individuals. Each method is processed for one generation and new solutions replace old ones. As DBmax has the ability to quickly move the distant individuals to the feasible region, it has been applied to the worst group of infeasible individuals (i.e. those far away from the feasible region). After each generation, and before applying genetic search operators, the size of each group is adaptively updated based on its success. The updating procedure is discussed in Section 4.2. The population is also updated by merging the new individuals with the remaining individuals (both feasible and infeasible). The genetic operators of GA-MPC are then applied, as described in Section 3. The selection process is discussed in Section 4.3. The algorithm is continued until the overall stopping criteria are met. The basic steps of the proposed algorithm (GA-MPC-M) are presented in Algorithm 1.

Algorithm 1. Steps of GA-MPC-M algorithm

**Step 1:** In generation $t = 0$, generate an initial random population ($PS$ is the number of individuals in the population). The variables in each individual $i$ are generated as follows:

$$x_{ij} = L_j + \text{rand} \times (U_j - L_j)$$

where $L_j, U_j$ are the lower and upper bound for decision variable $x_{ij}$ and rand is a random number, rand $\in [0, 1]$.

**Step 2:** Divide $PS$ into two groups. The first group contains the feasible individuals, if any, and the second group contains infeasible individuals, if any.

**Step 3:** At each generation, if there are infeasible individuals, select the best $P$ of them, and divide them into three subgroups ($group_i$, $i = 1, 2$ and $3$) of equal size ($n_i$, $i = 1, 2$ and $3$); **else**; go to **Step 5**.

**Step 4:** For each infeasible individual selected in **Step 3**, generate a random number ($rand$) between 0 and 1; if $rand < 0.5$, **then** apply the CC methods as follows: (a) basic CC to individuals in $group_1$, (b) FDfar to individuals in $group_2$, and (c) DBmax to individuals in $group_3$. 
Step 5: The new $P$ individuals, obtained from Step 4, will replace the older ones, and then the average improvement for each CC method is calculated (as described in Subsection 4.2) and the size $n_i$ is changed adaptively in each generation.

Step 6: Apply the genetic operators such as selection, crossover and mutation as of GA-MPC (as discussed in Section 3).

Step 7: Stop if the termination criterion is met; else set $t = t + 1$, and go to Step 3.

4.2. The success measure

As discussed earlier, three CC methods are considered in this article. Initially, they are applied to equal numbers of infeasible individuals, which means $n_1 = n_2 = n_3$. However, these numbers may be changed depending on the success of the CC methods in the current generation. This means that the size of the subgroup can be increased for a well-performing CC method and similarly it can be decreased for a poorly performing CC method. After applying a CC method, if an individual has improved, a success measure is increased by 1. Using this concept, the size of the subgroups ($n_i$) can be revised as follows:

Step 1: Set success $S_{cci} = 0$, where $cci = 1$, 2 and 3 are for the CC methods basic CC, FDfar and DBmax, respectively.

Step 2: For each $cci$, $y = 1 : n_i$ (for all individuals in a subgroup $n_i$)

If the infeasible solution has been improved, then

$$S_{cci} = S_{cci} + 1$$

Step 3: Calculate the proportion of total success for each $cci$: $\text{Prop}_{cci} = S_{cci} / (S_{cc1} + S_{cc2} + S_{cc3})$, and then

Step 4: $n_i = \text{Prop}_{cci} / P$, where $P$ is the number of the infeasible individuals that should be handled by all the CC methods.

4.3. Selection process

In this article, the selection of the individuals (Deb 2000) is considered as follows: (1) between two feasible solutions, the fitter one (according to the fitness function) is better; (2) a feasible solution is always better than an infeasible one; and (3) between two infeasible solutions, the one having the smaller sum of its constraint violation is preferred. The equality constraints are also transformed to inequalities of the following form, where $\varepsilon$ is a small value:

$$|h_e(\vec{x})| - \varepsilon \leq 0, \text{ for } e = 1, \ldots, E$$

(6)

where $E$ is the number of equality constraints.

5. Experimental results

In this section, the computational results of the proposed algorithms are presented and analysed, using two sets of specialized benchmark problems: (1) 24 problems from the IEEE Congress on Evolutionary Computation 2006 (CEC2006) and (2) 36 instances of 18 problems [each with 10 dimensions (10D) and 30 dimensions (30D)] from CEC2010. The details of the first set of test problems, with their known best fitness values, are shown in Table 1.
Table 1. Details of the 24 test problems.

| Problem | D  | Objective function              | ratio (%) | LI | NI | LE | NE | a                  | Optimal         |
|---------|----|---------------------------------|-----------|----|----|----|----|--------------------|-----------------|
| g01     | 13 | Quadratic                       | 0.0000    | 9  | 0  | 0  | 0  | 6                 | −15.000000000   |
| g02     | 20 | Rational of trigonometric and polynomial | 99.9971   | 2  | 0  | 0  | 0  | 1                 | −0.8036191042   |
| g03     | 10 | Polynomial                      | 0.0000    | 0  | 0  | 1  | 1  | 1                 | −1.0005010000   |
| g04     | 5  | Quadratic                       | 52.1230   | 0  | 0  | 0  | 0  | 2                 | −30665.53867178 |
| g05     | 4  | Cubic                           | 0.0000    | 2  | 0  | 3  | 3  | 3                 | 512.4967140071  |
| g06     | 2  | Cubic                           | 0.0066    | 0  | 2  | 0  | 0  | 2                 | −6961.813875580 |
| g07     | 10 | Quadratic                       | 0.0003    | 3  | 5  | 0  | 0  | 6                 | 24.3062090681   |
| g08     | 2  | Rational of trigonometric and polynomial | 0.8560    | 0  | 2  | 0  | 0  | 0                 | −0.0958250415   |
| g09     | 7  | Polynomial                      | 0.5121    | 0  | 4  | 0  | 0  | 2                 | 680.6300573745  |
| g10     | 8  | Linear                          | 0.0010    | 3  | 3  | 0  | 0  | 6                 | 7049.2480205286 |
| g11     | 2  | Quadratic                       | 0.0000    | 0  | 0  | 1  | 1  | 1                 | 0.7499000000    |
| g12     | 3  | Quadratic                       | 4.7713    | 0  | 1  | 0  | 0  | 0                 | −1.0000000000   |
| g13     | 5  | Exponential                     | 0.0000    | 0  | 0  | 3  | 3  | 3                 | 0.0539415140    |
| g14     | 10 | Polynomial of logarithm         | 0.0000    | 0  | 3  | 0  | 3  | 3                 | −47.7648884595  |
| g15     | 3  | Quadratic                       | 0.0000    | 0  | 1  | 1  | 2  | 2                 | 961.7150222899  |
| g16     | 5  | Polynomial of rational function | 0.0204    | 4  | 3  | 0  | 0  | 4                 | −1.9051552586   |
| g17     | 6  | Nonlinear                       | 0.0000    | 0  | 0  | 4  | 4  | 4                 | 8853.5396748064 |
| g18     | 9  | Quadratic                       | 0.0000    | 0  | 13 | 0  | 0  | 6                 | −0.8660254038   |
| g19     | 15 | Polynomial                      | 33.4761   | 0  | 5  | 0  | 0  | 0                 | 32.6585929502   |
| g20     | 24 | Linear                          | 0.0000    | 6  | 2  | 12 | 16 | 16                | 0.2049794002    |
| g21     | 7  | Linear                          | 0.0000    | 0  | 1  | 15 | 6  | 15                | 193.7245100700  |
| g22     | 22 | Linear                          | 0.0000    | 0  | 1  | 11 | 19 | 19                | 236.4309755040  |
| g23     | 9  | Linear                          | 0.0000    | 0  | 2  | 3  | 1  | 6                 | −400.0551000000 |
| g24     | 2  | Linear                          | 79.6556   | 0  | 2  | 0  | 0  | 2                 | −5.5080132716   |

Note: ratio = estimated ratio between the feasible region and the search space; LI = number of linear inequality constraints; NI = number of nonlinear inequality constraints; LE = number of linear equality constraints; NE = number of nonlinear equality constraints; a = number of active constraints; Optimal = optimal objective value of every test problem.

For the parameter settings, PS is set to 100 individuals, and the number of infeasible individuals that follow the CC methods, P, equals 10% of the infeasible solutions. For GA, the crossover rate is 100%, the probability of the randomized operator is 10% and $\tilde{z}$ (shown in Equations 3–5) is a random Gaussian number with mean value equal to 0.7 and a standard deviation of 0.1. These parameter settings are the same as those used in Elsayed, Sarker, and Essam (2011b, 2011c). The parameters for the stopping criteria of the CC methods are $\alpha = 10^{-6}$, $\beta = 0.0001$ and $\mu = 10$.

5.1. GA-MPC with a single constraint consensus method

In this subsection, the results of the GA-MPC algorithm are briefly discussed and they are compared with GA-MPC combined with a single CC method, namely the basic CC method (GA-MPC-CC), FDFar (GA-MPC-FDFar) and DBmax (GA-MPC-DBmax). In each version, the initial population is divided into two groups (feasible and infeasible). The CC method is then applied to the best $P$ infeasible individuals. The new individuals are merged with the remaining individuals (not considered by the CC method), and the GA-MPC algorithm is then applied to all the individuals. The detailed results [best, median, mean, worst and standard deviation (St.d)] can be found in the supplementary document (Appendix A) available through the web link given at the end of this article. The results are based on 25 independent runs.

From the literature, it is known that there is no feasible solution for g20, while for g22 it is rare to find a feasible solution. Therefore, these two problems are excluded from the comparisons. With regard to the best values obtained, it is found that GA-MPC-CC, GA-MPC-FDFar and GA-MPC-DBmax are better than GA-MPC only for five, four and four test problems, respectively,
while GA-MPC-CC, GA-MPC-FDfar and GA-MPC-DBmax are able to obtain the same best solutions as GA-MPC for 16, 16 and 15 test problems, respectively. However, GA-MPC-CC, GA-MPC-FDfar and GA-MPC-DBmax are inferior to GA-MPC for one, two and three problems, respectively. Considering the average results, it is found that GA-MPC-CC, GA-MPC-FDfar and GA-MPC-DBmax are superior to GA-MPC only for five test problems each, while GA-MPC-CC, GA-MPC-FDfar and GA-MPC-DBmax are able to obtain the same average results as GA-MPC only for 14 test problems each. Lastly, GA-MPC-CC, GA-MPC-FDfar and GA-MPC-DBmax are inferior to GA-MPC only for three test problems each.

To compare the above algorithms, statistical significance testing was performed. A non-parametric test, the Wilcoxon signed rank test (Corder and Foreman 2009) was chosen, which allows the difference between paired scores to be judged when the assumptions required by the paired-samples t test, such as that the population should be normally distributed, cannot be made. As a null hypothesis, it is assumed that there is no significant difference between the best and/or mean values of two samples, whereas the alternative hypothesis is that there is a significant difference in the best and/or mean fitness values of the two samples, with 5% significance level. The sign of the difference between the two independent samples is used to classify cases into one of two samples: differences below zero (negative rank $R^-$) or above zero (positive rank $R^+$); the corresponding $p$ value is also calculated. Based on the test results/rankings, one of three signs (+, − or ≈) is assigned for the comparison of any two algorithms (shown in the last column), where the ‘+’ sign means that the first algorithm is significantly better than the second, the ‘−’ sign means that the first algorithm is significantly worse and the ‘≈’ sign means that there is

![Figure 1](https://example.com/figure1.png)

Figure 1. Convergence plot of GA-MPC-CC, GA-MPC-DBmax and GA-MPC-FDfar for g01, g04, g10 and g18.
no significant difference between the two algorithms. The test statistics show that there is no significant difference between GA-MPC and each of the three variants discussed above.

As samples, a few convergence plots for these three variants for several problems are presented in Figure 1. These figures reveal that GA-MPC-FDfar is the best during the early generations, while GA-MPC-CC is better later on. Generally speaking, the integration of the CC methods with GA clearly results in an improvement.

During the evolution process, it is important to analyse the percentage of feasible solutions that are obtained by GA-MPC-CC, GA-MPC-FDfar and GA-MPC-DBmax in comparison to GA-MPC only. It is found that GA-MPC-CC increases the percentage of feasible solutions by 1.42%, while GA-MPC-FDfar and GA-MPC-DBmax increase the percentage of feasible solutions by 2.11% and 2.23%, respectively. It is clear that the inclusion of CC methods with GA increases the feasibility rate.

5.2. Genetic algorithm with multi-parent crossover with a mix of constraint consensus methods

The detailed experimental results are shown in the supplementary document (Appendix A). With regard to the best values obtained, it is found that the algorithm GA-MPC with a mix of CC methods (GA-MPC-M) was better than GA-MPC for five test problems and was able to obtain the same best solutions for 15 test problems. The algorithm was inferior to GA-MPC for only two test problems. Based on the average results, GA-MPC-M is superior to GA-MPC for seven problems and is equal for 14 test problems. The algorithm is inferior for one test problem only. Considering the Wilcoxon signed rank test, with regard to the best results, there is no significance between the two algorithms, whereas for the average results, GA-MPC-M is clearly superior to GA-MPC. Note that the algorithm GA-MPC-M increases the number of feasible solutions found during the search process by 3.43%, and it also requires 4.040% lower computational time than GA-MPC.

5.3. Constraint consensus memetic algorithm

In this section, the final form of the proposed algorithm is discussed. In this algorithm, a local search technique to GA-MPC-M has been applied for further improving the solution quality. Note that the integration of a local search with GA is recognized as being a memetic algorithm. This version of the algorithm is named CCMA.

SQP (Powell 1978) is used as the local search technique in CCMA. The idea of SQP is to model a problem, at a given approximate solution, by a quadratic programming subproblem. The solution to this subproblem is then used to construct a better approximation (new point) (Boggs and Tolle 1995). As this method may be viewed as an extension of Newton and quasi-Newton methods to the constrained optimization setting, the SQP methods share many of the characteristics of Newton-like methods, such as that when the iterates are close to the solution, a rapid convergence can be achieved, and when the iterates are far from a solution, a possible eccentric behaviour, which needs to be carefully controlled, can occur (Boggs and Tolle 1995).

CCMA follows the same steps as GA-MPC-M (presented in Algorithm 1), with an additional step. In this additional step, one random individual is selected and the SQP technique is applied to it for up to a defined number of fitness evaluations (FEs) (here it is equal to 500 FEs). To reduce the chance of becoming trapped into a local optimum, SQP is applied periodically, i.e. once every 25 generations. This means that at every 25th generation, after completing Step 6, the local search technique will be executed.

Similarly to the previous few sections, CCMA was used to solve the CEC2006 competition problem set. The results of these problems are presented in the supplementary document (Appendix A).
In Appendix A, it is found that CCMA is able to obtain the optimal solution for all CEC2006 test problems, except for one ($g_{14}$) in which the difference is negligible ($\approx 3.0E-8$), with a 100% feasibility ratio. If the above difference is considered as negligible, from the summary of comparisons presented in Table 2, CCMA shows better performance than GA-MPC-CC, GA-MPC-DBmax, GA-MPC-FDfar, GA-MPC-M and GA-MPC, in terms of the best fitness value. Based on the statistical test, CCMA is superior to all algorithms except for GA-MPC-FDfar, in which there is no significant difference. With regard to the average results, there is no significant difference between CCMA and GA-MPC-CC, GA-MPC-DBmax, GA-MPC-FDfar, GA-MPC-M and GA-MPC.

In terms of the computational time, CCMA consumes, on average, only 9.174 seconds. This means that CCMA saves 35.048%, 36.304%, 36.317%, 34.350% and 32.243% of the computational time in comparison to GA-MPC-DBMax, GA-MPC-FDfar, GA-MPC-CC, GA-MPC and GA-MPC-M respectively.

In Figure 2, a few convergence patterns for the algorithms are shown. It can be seen that CCMA performs the best, in that it is able to reach the optimal solution very quickly. In particular, a slightly different pattern is shown in Figure 2(d), where although the convergence of CCMA in the early generations is slow, it can reach the optimal solution before any other algorithms.

### 5.4. Comparison to the state-of-the-art algorithms

In this section, the performance of CCMA is first compared, based on CEC2006 test problems, with other state-of-the-art algorithms, including self-adaptive multi-operator differential evolution (SAMODE) (Elsayed, Sarker, and Essam 2011d), ensemble constraint handling techniques based on evolutionary programming (ECHTEP2) (Mallipeddi and Suganthan 2010a), modified differential evolution (MDE) (Mezura-Montes, Velazquez-Reyes, and Coello Coello 2006), adaptive penalty formulation based on genetic algorithm (APF-GA) (Tessema and Yen 2009), adaptive trade-off model with evolution strategy (ATMES) (Wang et al. 2008) and multimembered evolution strategy (SMES) (Mezura-Montes and Coello 2005). It must be mentioned here that CCMA used the same number of FEs (240 K) as SAMODE, ECHTEP-2, ATMES and SMES, while MDE and APF-GA used 500,000 FEs. The parameter $\epsilon$ was set to 1.0E-04 for all algorithms, except for ATMES and SMES, in which the parameter $\epsilon$ was set to 5.0E-06 and 4.0E-04, respectively. The detailed results for all the above algorithms are shown in the supplementary document (Appendix B), available through the web link given at the end of this article.
The comparisons for the quality of the solutions are summarized in Table 3. It must also be mentioned here that ATMES and SMES solved only the first 13 test problems; therefore, they were compared for those 13 problems only. Considering the best fitness, CCMA obtained either better or equal, but not worse, fitness values compared with SAMODE, ECHT-EP2, MDE, APF-GA, ATMES and SMES. In term of average fitness, CCMA obtained a higher number of better solutions than the others. Based on the statistical test, CCMA is better than APF-GA and SMES, while there is no significant difference between CCMA and SAMODE, ECHT-EP2, MDE and ATMES. Considering the average results, CCMA is superior to SMES, while there is no significant
Table 4. Comparison between CCMA and ε DEag.

| Comparison | Criteria | 10D | 30D |
|------------|----------|-----|-----|
| CCMA – to – ε DEag | Better | Tie | Worse | R− | R+ | Dec. | p | Better | Tie | Worse | R− | R+ | Dec. | p |
| Best       | 5       | 12  | 1    | 15  | 6   | ≈    | 0.345 | 17   | 1   | 0    | 153 | 0   | +    | 0.0003 |
| Average    | 8       | 7   | 3    | 43  | 23  | ≈    | 0.374 | 15   | 0   | 3    | 147 | 24  | +    | 0.007  |

Note: Dec. = statistical decision.

difference between CCMA and SAMODE, ECHT-EP2, MDE APF-GA and ATMES. It is now clear that the overall performance of CCMA is either better than or equal to SAMODE, ECHT-EP2, MDE, APF-GA, ATMES and SMES.

5.5. Solving the CEC2010 test problems

In this section, the algorithm was tested on another set of problems, known as the CEC2010 competition problems on constrained optimization. This set contains 18 test problems, each with two instances (one 10D and the other 30D). Each problem was run 25 times with the maximum number of FEs being equal to 200,000 and 600,000 for the 10D and 30D problems, respectively. The details of this problem set can be found in Mallipeddi and Suganthan (2010b). The proposed algorithm has been compared with the winning algorithm of the CEC2010 competition, namely ε DEag (Takahama and Sakai 2010). The detailed results are presented in the supplementary document (Appendix C), available through the web link given at the end of this article.

With regard to the feasibility ratio, CCMA obtained a 100% feasibility ratio for both 10D and 30D, while ε DEag reached 100% and 95.11% for 10D and 30D, respectively.

Based on the quality of the solutions obtained, a comparison summary is presented in Table 4. From this table, for the 10D test problems, CCMA is better than ε DEag, with regard to the best values obtained, for five test problems, while it is worse for only one test problem. Considering the average results, for the 10D problems, CCMA is better than ε DEag for eight test problems, while it is worse for three problems. On the other hand, for the 30D test problems, CCMA is able to obtain better best solutions than ε DEag for 17 problems, while ε DEag is not better for any test problems. With regard to the average results, CCMA is superior to ε DEag for 15 problems, while it is worse for three problems.

Furthermore, based on the statistical test, CCMA is superior to ε DEag when considering both the best and average results obtained on the 30D test problems, while there is no significant difference in the 10D test problems.

6. Static economic load dispatch problems

The purpose of this section is to demonstrate that the proposed algorithm can effectively solve not only the benchmark problems, but also real-world complex constrained optimization problems. For this reason, a popular practical problem, which is known as the static ELDP, has been chosen.

The ELDP is an important issue for any power-generating system, which basically determines the commitment of each generating unit, either at each time interval of the planning horizon, or for a given period of time (Dahal, Aldridge, and Galloway 2007). The current problem involves only thermal power plants, where the aim is to minimize the total fuel cost while allocating power loads to different plants. The mathematical model of such problems contains both equality and inequality constraints. The objective function is usually derived from the relevant production cost.
equations, which can be approximated as a quadratic function of the active power outputs from the generating units (Chaturvedi, Pandit, and Srivastava 2008; Cheng-Chien 2008; Vlachogiannis and Lee 2009). It can be represented as follows:

\[ F = \sum_{i=1}^{N_G} f_i(P_i) \]  

where \( f_i(P_i) = a_i P_i^2 + b_i P_i + c_i \), \( i = 1, 2, \ldots, N_G \) is the expression for the cost function corresponding to the \( i \)th generating unit, and \( a_i, b_i \) and \( c_i \) are the cost coefficients; \( P_i \) is the real power output (MW) of the \( i \)th generator that corresponds to period \( t \), and \( N_G \) is the number of on-line generating units to be dispatched. The cost function for a unit with a valve point loading effect is calculated using:

\[ f_i(P_i) = a_i P_i^2 + b_i P_i + c_i + |e_i \sin(f_i(P_i^{\min} - P_i))| \]  

where \( e_i \) and \( f_i \) are the cost coefficients corresponding to the valve point loading effect.

This problem is subject to different constraints depending upon the assumptions and practical implications, as shown below.

1. **Power balance constraints or demand constraints**: These constraints are based on the principle of equilibrium between total system generation (\( \sum_{i=1}^{N_G} (P_i) \)) and total system loads (\( P_D \)) and losses (\( P_L \)). That is,

\[ \sum_{i=1}^{N_G} (P_i) - P_D - P_L = 0 \]  

where \( P_L \) is obtained using \( B \)-coefficients, given by

\[ P_L = \sum_{i=1}^{N_G} \sum_{j=1}^{N_G} P_i B_{ij} P_j + \sum_{i=1}^{N_G} B_{0i} P_i + B_{00} \]  

2. **Generator constraints**: The output power of each generating unit has a lower and an upper bound. This constraint is represented by a pair of inequality constraints:

\[ P_i^{\min} \leq P_i \leq P_i^{\max} \]  

where \( P_i^{\min} \) and \( P_i^{\max} \) are the lower and upper bounds, respectively, for the power output of the \( i \)th generating unit.

3. **Ramp rate limits**: It is commonly assumed that the adjustments of the power output are instantaneous. However, under practical circumstances, for all the on-line units, ramp rate limits restrict the operating range for adjusting the generator operation between two operating periods. Hence, generation may increase or decrease with corresponding upper and downward ramp rate limits. Thus, units are constrained by these ramp rate limits, such as: \( P_i - P_i^{t-1} \leq UR_i \), if power generation increases; and \( P_i^{t-1} - P_i \leq DR_i \) if power generation decreases, where \( P_i^{t-1} \) is the power generation of unit \( i \) in the previous period, and \( UR_i \) and \( DR_i \) are the upper and lower ramp rate limits, respectively. The inclusion of ramp rate limits modifies the generator operation constraints as follows:

\[ \max(P_i^{\min}, UR_i - P_i) \leq P_i \leq \min(P_i^{\max}, P_i^{t-1} DR_i) \]

4. **Prohibited operating zone**: The generating units may have certain ranges where the operation is restricted on the grounds of the physical limitations of machine components or instability, e.g. due to steam valves or vibration in shaft bearings. Consequently, discontinuities are
produced in the cost curves that correspond to the prohibited operating zones. Hence, there is a desire to avoid operations in these zones in order to economize production. Symbolically, for a generating unit \( i \),

\[
P_{pz} \leq P_i \leq \bar{P}_{pz}
\]  

(13)

where \( P_{pz} \) and \( \bar{P}_{pz} \) are the lower and upper limits of a given prohibited zone for unit \( i \). In this research, five cases of the ELDP are considered from the literature (Chaturvedi, Pandit, and Srivastava 2008; Cheng-Chien 2008; Vlachogiannis and Lee 2009): (1) Case 1: 6 generating units of IEEE 30-bus system; (2) Case 2: 13 generating units; (3) Case 3: 15 generating units; (4) Case 4: 40 generating units; and (5) Case 5: 140 generating units.

6.1. Computational study

The proposed algorithm (CCMA) has been applied to solve the above five cases with the same parameters used in earlier sections. The results obtained were compared with two other state-of-the-art algorithms: differential evolution with multiple strategies (SAMODE) (Elsayed, Sarker, and Essam 2011a) and adaptive differential evolution algorithm (IDE) (Asafuddoula, Ray, and Sarker 2011). All algorithms used 150,000 FEs. The detailed results are shown in Table 5.

From Table 5, it is clear that the best fitness values of CCMA are better than the same values for SAMODE and IDE for four and five cases, respectively, while this value is equal to SAMODE for only one case. Based on the average results, CCMA is better than SAMODE and IDE for four and five cases, respectively, but it is inferior to SAMODE for only one case. With regard to the standard deviation, CCMA is better than SAMODE and IDE for three and five cases, respectively, but it is inferior to SAMODE for two cases. As shown in Table 6, considering the Wilcoxon signed

| Case | Algorithm | Best     | Median   | Mean     | Worst    | St.d     |
|------|-----------|----------|----------|----------|----------|----------|
| 1    | CCMA      | 1.544419E+04 | 1.544426E+04 | 1.544432E+04 | 1.544473E+04 | 1.64024E-01 |
|      | SAMODE    | 1.544419E+04 | 1.544422E+04 | 1.544423E+04 | 1.544433E+04 | 3.73331E-02 |
|      | IDE       | 1.544453E+04 | 1.547957E+04 | 1.548207E+04 | 1.556846E+04 | 3.04773E+01 |
| 2    | CCMA      | 1.793590E+04 | 1.793806E+04 | 1.793842E+04 | 1.794183E+04 | 1.66375E+00 |
|      | SAMODE    | 1.804684E+04 | 1.816215E+04 | 1.818559E+04 | 1.843584E+04 | 1.07141E+02 |
|      | IDE       | 1.822472E+04 | 1.856815E+04 | 1.855026E+04 | 1.878500E+04 | 1.41515E+02 |
| 3    | CCMA      | 3.258984E+04 | 3.262220E+04 | 3.261784E+04 | 3.262806E+04 | 1.0887E+01  |
|      | SAMODE    | 3.272195E+04 | 3.273663E+04 | 3.273758E+04 | 3.275306E+04 | 7.77791E+00 |
|      | IDE       | 3.274420E+04 | 3.285861E+04 | 3.285940E+04 | 3.305473E+04 | 6.84137E+01 |
| 4    | CCMA      | 1.218831E+05 | 1.223528E+05 | 1.223528E+05 | 1.228948E+05 | 2.82452E+02 |
|      | SAMODE    | 1.227875E+05 | 1.240997E+05 | 1.240997E+05 | 1.254635E+05 | 8.07258E+02 |
|      | IDE       | 1.240690E+05 | 1.255323E+05 | 1.255323E+05 | 1.274304E+05 | 8.362557E+02 |
| 5    | CCMA      | 1.813423E+06 | 1.838665E+06 | 1.838665E+06 | 1.867553E+06 | 1.30223E+04 |
|      | SAMODE    | 1.889131E+06 | 1.959675E+06 | 1.987672E+06 | 2.363069E+06 | 1.25565E+05 |
|      | IDE       | 1.890671E+06 | 1.924213E+06 | 1.925098E+06 | 1.982961E+06 | 2.34290E+04 |

Table 6. Wilcoxon signed rank test results for CCMA against SAMODE and IDE.

| Algorithms | Criteria      | \( R^- \) | \( R^+ \) | Decision | \( p \)   |
|------------|---------------|-----------|-----------|----------|-----------|
| CCMA – to –SAMODE | Best fitness  | 10        | 0         | +        | 0.068     |
|             | Average fitness | 14     | 1         | +        | 0.080     |
| CCMA – to –IDE  | Best fitness  | 15        | 0         | +        | 0.043     |
|             | Average fitness | 15       | 0         | +        | 0.043     |
rank test with the 10% significance level, CCMA is superior to both the algorithms with regard to the best and average solutions.

7. Conclusion and future work

During the past few decades, many evolutionary algorithms have been introduced to solve constrained optimization problems. In those algorithms, reaching feasibility or near feasibility quickly is not an easy task, especially when the feasible region is tiny. In this research, different GA variants are introduced that are able to reach the feasible region quickly. They are: (1) the basic CC method with a genetic algorithm with multi-parent crossover (GA-MPC); (2) the DBmax method with GA-MPC; (3) the FDfar method with GA-MPC; and (4) an ensemble of the basic CC, DBmax and FDfar methods with GA-MPC.

In the ensemble version, which is known as GA-MPC-M, each CC method (the basic CC, DBmax and FDfar) is assigned adaptively to certain individuals. With regard to both the quality of solutions and the computational time, this algorithm was clearly better than the GA-MPC algorithm. The GA-MPC-M algorithm was further improved by adding a local search (SQP), which was called CCMA. CCMA was found to be superior to GA-MPC-CC, GA-MPC-DBmax, GA-MPC-FDfar, GA-MPC-M and GA-MPC in terms of both the quality of results and the computational time required. CCMA was found to be competitive, if not better, in comparison to SAMODE, ECHT-EP2, MDE, APF-GA, ATMES and SMES. In addition to this, CCMA was tested on the CEC2010 constrained problems, with both 10 and 30 dimensions. The results showed that it was better than the best algorithm in the CEC2010 competition.

CCMA has also been applied to solve a real-world complex constrained optimization problem. From the comparison of results, CCMA clearly dominated two other well-known algorithms. The statistical testing supports this finding.

For future work, it is intended that CCMA will be applied to other optimization problems, different parameters will be analysed and further attempts will be made to improve the performance of the algorithm.

Supplemental data

Supplemental data for this article can be accessed at http://dx.doi.org/10.1080/0305215X.2013.846336

References

Asafuddoula, M., T. Ray, and R. Sarker. 2011. “An Adaptive Differential Evolution Algorithm and Its Performance on Real World Optimization Problems.” IEEE Congress on Evolutionary Computation, New Orleans, USA, 1057–1062.
Boggs, P. T., and J. W. Tolle. 1995. “Sequential Quadratic Programming.” Acta Numerica 4: 1–51.
Brailsford, S. C., C. N. Potts, and B. M. Smith. 1999. “Constraint Satisfaction Problems: Algorithms and Applications.” European Journal of Operational Research 119 (3): 557–581.
Cai, Z., and Y. Wang. 2006. “A Multiobjective Optimization-Based Evolutionary Algorithm for Constrained Optimization.” IEEE Transactions on Evolutionary Computation 10 (6): 658–675.
Censor, Y., D. Gordon, and R. Gordon. 2001. “Component Averaging: An Efficient Iterative Parallel Algorithm for Large and Sparse Unstructured Problems.” Parallel Computing 27 (6): 777–808.
Censor, Y., and S. Zenios. 1997. Parallel Optimization: Theory, Algorithms, and Applications. New York: Oxford University Press.
Chaturvedi, K. T., M. Pandit, and L. Srivastava. 2008. “Self-Organizing Hierarchical Particle Swarm Optimization for Nonconvex Economic Dispatch.” IEEE Transactions on Power Systems 23 (3): 1079–1087.
Cheng-Chien, K. 2008. “A Novel Coding Scheme for Practical Economic Dispatch by Modified Particle Swarm Approach.” IEEE Transactions on Power Systems 23 (4): 1825–1835.
Tjalling, Y. J. 1995. “Historical Development of the Newton–Raphson Method.” *SIAM Review* 37 (4): 531–551.

Vlachogiannis, J. G., and K. Y. Lee. 2009. “Economic Load Dispatch—A Comparative Study on Heuristic Optimization Techniques with an Improved Coordinated Aggregation-Based PSO.” *IEEE Transactions on Power Systems* 24 (2): 991–1001.

Wang, Y., and Z. Cai. 2011. “Constrained Evolutionary Optimization by Means of (μ + λ)-Differential Evolution and Improved Adaptive Trade-off Model.” *Evolutionary Computation* 19 (2): 249–285.

Wang, Y., and Z. Cai. 2012a. “Combining Multibjective Optimization with Differential Evolution to Solve Constrained Optimization Problems.” *IEEE Transactions on Evolutionary Computation* 16 (1): 117–134.

Wang, Y., and Z. Cai. 2012b. “A Dynamic Hybrid Framework for Constrained Evolutionary Optimization.” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 42 (1): 203–217.

Wang, Y., Z. Cai, G. Guo, and Y. Zhou. 2007a. “Multiobjective Optimization and Hybrid Evolutionary Algorithm to Solve Constrained Optimization Problems.” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 37 (3): 560–575.

Wang, Y., Z. Cai, Y. Zhou, and W. Zeng. 2008. “An Adaptive Tradeoff Model for Constrained Evolutionary Optimization.” *IEEE Transactions on Evolutionary Computation* 12 (1): 80–92.

Wang, Y., H. Liu, Z. Cai, and Y. Zhou. 2007b. “An Orthogonal Design Based Constrained Evolutionary Optimization Algorithm.” *Engineering Optimization* 39: 715–736.

Xiao, Y., Y. Censor, D. Michalski, and J. Galvin. 2003. “The Least-Intensity Feasible Solution for Aperture-Based Inverse Planning in Radiation Therapy.” *Annals of Operations Research* 119: 183–203.