Computers and Structures 89 (2011) 1352–1366

Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

Constrained structural design optimization via a parallel augmented Lagrangian particle swarm optimization approach

P.W. Jansen, R.E. Perez *

Department of Mechanical and Aerospace Engineering, Royal Military College of Canada, Kingston, ON, Canada K7K 7B4

A R T I C L E   I N F O

Article history:
Received 30 June 2010
Accepted 22 March 2011
Available online 20 April 2011

Keywords:
Structural optimization
Truss structures
Particle swarms
Constrained optimization
Augmented Lagrangian
Parallel computing

A B S T R A C T

This paper presents an extension to the basic particle swarm optimization approach for the solution of constrained engineering design optimization problems. The approach takes advantage of the PSO ability to find global optimum in problems with complex design spaces while directly enforcing feasibility of constraints using an augmented Lagrange multiplier method. Details in the algorithm implementation and properties are presented and the effectiveness of the approach is illustrated in different benchmark structural optimization test cases. Results show the ability of the proposed methodology to find better solutions for structural optimization tasks as compared to other optimization algorithms.

1. Introduction

In many practical applications, structural engineers are confronted with design problems that have multi-modal, non-convex, non-differentiable, and/or non-continuous design spaces and that are driven by a large number of non-linear equality and inequality constraints. It is desirable to have a flexible optimization approach which is able to traverse complex design spaces towards the global optimum while enforcing constraint feasibility.

Very efficient algorithms for the solution of constrained optimization problems currently exist such as the Sequential Unconstrained Minimization Techniques [1], the method of feasible directions [2,3], the method of Moving Asymptotes [4], and sequential quadratic programming [5]. Such algorithms exploit specific design problem properties such as differentiability and convexity which limit their application in the solution of structural design problems. Similarly, effective optimization approaches such as simulated annealing [6], genetic algorithms [7], particle swarms [8], ant colony optimization [9], bacterial foraging [10], imperialist competitive algorithm [11], charged system search [12], and cuckoo search algorithm [13] have been developed to traverse difficult design spaces towards the global optimum. All of these global optimization approaches, however, have been developed as unconstrained optimizers limiting their applicability to solve constrained structural design problems.

Among the different global optimizers, particle swarm optimization (PSO) has demonstrated its usefulness as an optimizer capable of finding global optimum in a variety of design applications in both structural design [14–16] and other fields of knowledge [17–25].

Different methods have been proposed to handle constraint optimization problems with the PSO algorithm. Constraint handling methods include violated design points redirection [26], the use of penalty methods [14] and adaptive penalty methods [27], extending the objective function with Lagrange multipliers [28], and problem reformulation into an unconstrained multi-objective formulation [29,30] or a series of sequential quadratic programming problems [31]. Many of these methods do not enforce constraint feasibility, limiting its possible use to practical applications.

In this paper, we present a parallel particle swarm optimization algorithm which uses a dynamic augmented Lagrangian multiplier method to enforce constraints. Specifically we build upon the work done by Sedlaczek and Eberhard [28] in extending the PSO to handle constraints with an augmented Lagrangian multiplier approach. We introduce the ability to parallelize function evaluations and analyze the effect that the number of inner iterations in an augmented Lagrangian multiplier implementation has on the quality of the solutions when used with the PSO. Furthermore, we propose the use of a dynamic inner iteration scheme which is demonstrated to reduce the computational cost while maintaining the accuracy and feasibility of results.

The development of this paper is as follows: in Section 2, we review the general PSO formulation and algorithm development, and discuss some convergence properties and improvements. Current

* Corresponding author.
E-mail addresses: Peter.Jansen@rmc.ca (P.W. Jansen), Ruben.Perez@rmc.ca (R.E. Perez).

0045-7949/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compstruc.2011.03.011

methods used to enforce constraints for the PSO are described in Section 3. In Section 4, we present details in the formulation, algorithmic implementation, and properties of a parallel PSO approach formulation which enforces constraints using an augmented Lagrange multiplier method. Section 5 presents different convex and non-convex constrained structural optimization case studies. The test cases are used to demonstrate the proposed algorithm behavior to its setting parameters, and compare the effectiveness of different algorithmic implementation options in finding optimal and feasible structural optimization solutions. Concluding remarks are presented in Section 6.

2. Particle swarm algorithm

Particle swarm optimization is a population-based and derivative-free global optimization method based on the adaptation process observed in flocking organisms, such as birds, bees, fish, when searching for regions with food availability with the ability to return to promising regions that have previously been discovered [8]. The adaptation process in the swarm process is stochastic in nature and depends on the local memory of each individual (particle) and the global memory of the population. Each particle movement in the design space is modelled using position and velocity information. Numerically, the position $\boldsymbol{x}_{k+1}^i$ and velocity $\boldsymbol{v}_{k+1}^i$ of a particle i at iteration $k+1$ is updated as:

$$\begin{aligned}\boldsymbol{x}_{k+1}^i &= \boldsymbol{x}_k^i + \boldsymbol{v}_{k+1}^i\Delta t,\\ \boldsymbol{v}_{k+1}^i &= w\boldsymbol{v}_k^i + c_1 r_1\left(\boldsymbol{p}_k^i - \boldsymbol{x}_k^i\right) + c_2 r_2\left(\boldsymbol{p}_k^g - \boldsymbol{x}_k^i\right),\end{aligned} \tag{1}$$

where $\boldsymbol{x}_k^i$ and $\boldsymbol{v}_k^i$ are respectively the position and velocity vector of the particles at iteration k, Δt is the time step value (considered unity in the present work), while r_1 and r_2 represent random numbers between 0 and 1, $\boldsymbol{p}_k^i$ represents the best particle position particle i has achieved so far, and $\boldsymbol{p}_k^g$ corresponds to the global best position found in the swarm up to iteration k so far [32]. The remaining three terms are setting parameters which affects the convergence behavior of the PSO algorithm [33–36]. The c_1 and c_2 are confidence parameters biasing a particle movement towards its own best solution or towards the global best solution found by the swarm, while w represents an inertia weight modifying the particle global/local search behaviour. Detailed illustration of the position and velocity update scheme can be found in [36], which also contains detailed analysis of the impact of the different setting parameters and derivation of a set of necessary and sufficient conditions that ensure stable behavior of the algorithm and guarantee convergence which is given by:

$$\begin{aligned}&0 < c_1 + c_2 < 4,\\ &\frac{(c_1 + c_2)}{2} - 1 < w < 1.\end{aligned} \tag{2}$$

Based on the particle and velocity updates as explained above, the steps of the algorithm can be outlined as follows:

1. Create an initial set of particle positions $\boldsymbol{x}_o^i$ and velocities $\boldsymbol{v}_o^i$ randomly distributed throughout the design space bounded by specified limits on each design variable and maximum velocities.
2. Evaluate the objective function values $f\left(\boldsymbol{x}_k^i\right)$ of each particle based on its position $\boldsymbol{x}_k^i$ in the design space.
3. Update the optimum particle position $\boldsymbol{p}_k^i$ at current iteration (k), if applicable, and global optimum particle position $\boldsymbol{p}_k^g$.
4. Update the position of each particle using its previous position and updated velocity vector as specified in Eq. (1).
5. Repeat steps 2–4 until the specified convergence criteria is met.

One aspect that can affect the efficiency of the PSO, to some degree, is the initial particle distribution of the swarm over the design space. Areas not initially covered may not get explored by the swarm, depending on the parameters used and the global and individual best solutions found at each iteration. Different approaches to initialize the particle positions have been investigated [16]. The simple approach of randomly distributing the initial positions and velocity vectors of each particle throughout the bounded design space has been proven successful in practice [14–16]. To achieve this, one can use Eqs. (3) and (4)

$$\boldsymbol{x}_0^i = \boldsymbol{x}_{min} + r(\boldsymbol{x}_{max} - \boldsymbol{x}_{min}), \tag{3}$$

$$\boldsymbol{v}_0^i = \boldsymbol{x}_{min} + r(\boldsymbol{x}_{max} - \boldsymbol{x}_{min}). \tag{4}$$

The term r in both equations represents a random number between 0 and 1, while $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$ represent the lower and upper bounds of the design variables, respectively. It should be noted that the PSO algorithm requires each design variable to have a lower and upper bound. The maximum velocity for each design variable is given by the maximum range of the particles $(\boldsymbol{x}_{max} - \boldsymbol{x}_{min})$ throughout the present work. The maximum velocity can be reduced to achieve a more local search behavior similar to small values in inertial weight. The current implementation of the algorithm allows for the specification of one or up to all of the initial positions of the swarm, if for example some prior knowledge is available about the design space, with the remaining particles being assigned randomly.

The basic algorithm described above has an unwanted property when $\boldsymbol{x}^i = \boldsymbol{p}^i = \boldsymbol{p}^g$ for any particle i. In this case, the velocity update (1) reduces to wv^i, so that when a particles position coincides with its best and the global best position, it will only move from this point if its velocity and w are non-zero. In the case where velocity is close to zero, all particles will stop moving when they catch up with the global best particle and the algorithm will converge prematurely. This position does not need to coincide with the global optimum and does not even have to be a local minima; this case merely indicates that all particles have converged on the best solution found so far. To address this problem van den Bergh [35] introduced a modification to the basic algorithm where the velocity and position update of the global best particle τ is changed to:

$$\boldsymbol{v}_{k+1}^{\tau} = -\boldsymbol{x}_k^{\tau} + \boldsymbol{p}_k^g + w\boldsymbol{v}_k^{\tau} + \rho_k(1 - 2r_2), \tag{5}$$

$$\boldsymbol{x}_{k+1}^{\tau} = \boldsymbol{p}_k^g + w\boldsymbol{v}_k^{\tau} + \rho_k(1 - 2r_2), \tag{6}$$

where r_2 is again a random number in the interval between 0 and 1. The term $-\boldsymbol{x}_k^{\tau}$ "resets" the particle position to the global best position and $w\boldsymbol{v}_k^{\tau}$ pushes the particle in the direction of the current search direction. To this, a random direction term is added from a sample space with side lengths $2\rho_k$ [35]. The addition of the parameter ρ causes the PSO algorithm to perform a random search in the vicinity of the best solution found so far by the swarm. The size of the area searched around the global best position is controlled by ρ, which gets updated according to:

$$\rho_{k+1} = \begin{cases} 2\rho_k & \textit{if } n_{successes} > s_c,\\ 0.5\rho_k & \textit{if } n_{failures} > f_c,\\ 2\rho_k & \textit{otherwise}, \end{cases} \tag{7}$$

where $n_{successes}$ and $n_{failures}$ denote the number of consecutive successes or failures, with a failure being defined as no change in the global best position, $f(p_k^g) = f(p_{k-1}^g)$, and s_c and f_c being threshold parameters. To ensure that Eq. (7) is well-defined, the number of successes or failures must be reset to zero when a failure or success occurs, respectively [35]. This adaptation scheme for the parameter ρ ensures that if it results in consecutive failures, then the search space is too large and is reduced. The same applies when the current value results in consecutive successes, which indicates the