UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Chapter 2

Particle swarm optimization

2.1 Particle swarm optimization: A brief history

Computer simulations of the movement and behavior patterns of bird flocks and fish schools were first presented by Reynolds [7], and Heppner (a zoologist) and Grenander [8]. These simulations attempted to define the underlying rules of the movement dynamics of bird flocking and fish schooling, and were mainly reliant on the manipulation of inter-individual distances. These studies were the precursors to the particle swarm paradigm.

The particle swarm optimization algorithm (PSOA), was first introduced by Kennedy and Eberhart in 1995 [9, 10], and, compared to other well established population based evolutionary methods such as genetic algorithms, is still in it's infancy. It finds its roots in a variety of fields, which, among others, include artificial life and collective intelligence, chaos theory, fuzzy computing, sociobiology and, interestingly enough, psychology.

Kennedy and Eberhart inferred a likeness of the swarm behavior to human social behavior. This deduction followed on the observation that, like individual fish or birds adjust their movement patterns to maintain their position in a school or flock, humans tend to adjust their beliefs and attitudes to conform to their peers.

The PSOA models the exploration of a problem space by a population of agents or particles; the agents' success history influence their own search patterns and those of their peers. The search is focused toward promising regions by biasing each particle's velocity vector toward their own remembered best position as well as the communicated best ever swarm location. The importance of these two positions are weighed by two factors, aptly called the cognitive and social scaling parameters [11]. These two components are the among the main governing parameters of swarm behavior (and algorithm efficiency), and have been the subject of extensive study [12, 13, 14].

As various studies [11, 15, 16] have revealed, there are a number of shortcomings and limitations to the 'standard' PSOA as first proposed by Kennedy and Eberhart. Subsequently the PSOA has undergone rapid development, with several adaptations to improve performance and to apply the algorithm to other types of problems have been proposed. These

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

adaptations or variants will be discussed and analyzed in Chapter 3.

Particle swarm optimization, while also being a population based method, differs from other similar methods such as genetic algorithms, evolutionary programming and evolutionary strategies, (for a brief discussion of these and other related approaches see Appendix C), in the respect that the primary operator which drives the algorithm is not evolutionary based, but rather a set of rules which dictate the social interaction between members of the swarm. This interaction takes the form of the exchange of information regarding the fitness history of the swarm which is used to influence decisions relating to areas to be explored. There are however a number of similarities between evolutionary approaches and the PSOA, which were studied by Angeline [17] and Eberhart [18].

Furthermore, the PSOA is simpler, both in formulation and computer implementation, than the GA. In addition, the PSOA seems to outperform the GA for a number of difficult programming classes, notably the unconstrained global optimization problem [12, 16].

Previously, the PSOA has been applied to analytical test functions, mostly univariate or bivariate without constraints, by Shi and Eberhart [15] and Kennedy [19]. Kennedy also applied the algorithm to multimodal problem generators and used the PSOA as an optimization paradigm to simulate the ability of human societies to process knowledge [12].

Notwithstanding it's recent popularity, the PSOA has a number of drawbacks, one of which is the presence of problem dependent parameters. Previously, a number of workers have attempted to find 'universal' values for the PSOA parameters, the most recent being Carlisle and Dozier [20].

2.2 PSOA Applications

2.2.1 Neural network training via particle swarm optimization

The PSOA has been applied with success in the field of neural network (NN) training, with this type of problem among one of the first to be addressed by Kennedy and Eberhart [9]. The "training" of a neural network involves the minimization of the fitness error in the forward propagated result through the network by adjusting the weights of the network components.

Extensive research by others have been done in this field, some of the more notable the work by Van den Bergh and Engelbrecht who have applied several modifications to the algorithm which involve dividing the swarm into sub-components and using them in a cooperative manner to solve the NN training problem [21, 22]. Other examples where the PSOA has been used in a neural network context include the training of a NN to identify the presence of Parkinson's disease in patients [23] and the extraction of rules from a fuzzy neural network [24].

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2.2.2 Structural optimization

Lately, the PSOA was successfully applied to the optimal shape and size design of structures by Fourie and Groenwold [25, 26], where the design variables represent geometric properties of the structure and certain constraints are enforced (e.g. displacement limits or maximum allowed stress). The optimal topological design of problems is also addressed by Fourie and Groenwold [27].

2.2.3 Other applications

The PSOA has also been applied to a variety of other types of problems, among others the optimization of reactive power and voltage control in electrical distribution networks [28] and the practical distribution state estimation thereof [29]. It has also been applied successfully to the field of process biochemistry [30].

2.3 Particle swarm optimization algorithm formulation

We will now formulate the particle swarm algorithm as proposed by Kennedy and Eberhart [9, 10]. The algorithm is constructed as follows: Let us consider a flock of p particles or birds, each representing a possible solution point in the problem space D. For each particle i, Kennedy and Eberhart originally proposed that the position $\boldsymbol{x}^i$ is updated in the following manner:

$$\boldsymbol{x}^i_{k+1} = \boldsymbol{x}^i_k + \boldsymbol{v}^i_{k+1}, \tag{2.1}$$

with the velocity $\boldsymbol{v}^i$ calculated as follows:

$$\boldsymbol{v}^i_{k+1} = \boldsymbol{v}^i_k + c_1 r_1(\boldsymbol{p}^i_k - \boldsymbol{x}^i_k) + c_2 r_2(\boldsymbol{p}^g_k - \boldsymbol{x}^i_k). \tag{2.2}$$

Here, subscript k indicates an (unit) pseudo-time increment. $\boldsymbol{p}^i_k$ represents the best ever position of particle i at time k, with $\boldsymbol{p}^g_k$ representing the global best position in the swarm at time k. r_1 and r_2 represent uniform random numbers between 0 and 1. Kennedy and Eberhart proposed that the cognitive and social scaling parameters c_1 and c_2 are selected such that $c_1 = c_2 = 2$, in order to allow a mean of 1 (when multiplied by the random numbers r_1 and r_2). The result of using these proposed values is that the particles overshoot the target half the time.

Let us denote the best ever fitness value of a particle at $\boldsymbol{p}^i_k$ as f^i_{best} and the best ever fitness value of a particle at $\boldsymbol{p}^g_k$ as f^g_{best}.

The particle swarm optimization algorithm is now outlined as follows:

1. Initialize
   (a) Set constants k_{max}, c_1, c_2.
   (b) Randomly initialize particle positions $\boldsymbol{x}^i_0 \in \boldsymbol{D}$ in $\mathbb{R}^n$ for $i = 1, ..., p$.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

   (c) Randomly initialize particle velocities $0 \leq \boldsymbol{v}_0^i \leq \boldsymbol{v}_0^{max}$ for $i = 1, ..., p$.
   (d) Set $k = 1$

2. Optimize
   (a) Evaluate function value f_k^i using design space coordinates $\boldsymbol{x}_k^i$.
   (b) If $f_k^i \leq f_{best}^i$ then $f_{best}^i = f_k^i$, $\boldsymbol{p}_k^i = \boldsymbol{x}_k^i$.
   (c) If $f_k^i \leq f_{best}^g$ then $f_{best}^g = f_k^i$, $\boldsymbol{p}_k^g = \boldsymbol{x}_k^i$.
   (d) If stopping condition is satisfied then goto 3.
   (e) Update all particle velocities $\boldsymbol{v}_k^i$ for $i = 1, ..., p$ with rule (2.1).
   (f) Update all particle positions $\boldsymbol{x}_k^i$ for $i = 1, ..., p$ with rule (2.2).
   (g) Increment k.
   (h) Goto 2(a).

3. Terminate

The above algorithm is also represented by the flow diagram depicted in Figure 2.1.

2.4 Analysis of velocity rule

The manner in which the velocity rule influences an individual particle's position can be explained at the hand of Figures 2.2, 2.3, 2.4 and 2.5. If we examine the velocity rule (2.2), we note that the cognitive contribution to calculating the velocity is:

$$c_1 r_1 (\boldsymbol{p}_k^i - \boldsymbol{x}_k^i), \tag{2.3}$$

with c_1 the cognitive parameter, r_1 a random number between 0 and 1, and $\boldsymbol{p}_k^i$ and $\boldsymbol{x}_k^i$ the best fitness and current positions of particle i respectively.

If we consider a 2-dimensional search space (Figure 2.2) we can determine the search area to which the particle can possibly move in the next update of (2.1). Replacing the time increment k with a dimensional index for the moment, we see that the distances between $\boldsymbol{p}^i$ and $\boldsymbol{x}^i$ are $(p_1^i - x_1^i)$ and $(p_2^i - x_2^i)$ for dimension 1 and 2 respectively. By virtue of difference calculated in (2.3), the direction the particle will move will always be toward $\boldsymbol{p}^i$ if we neglect the previous velocity v_k. Since r_1 varies between 0 and 1, the maximum possible travel distance (with $v_k = 0$) for the particle during a single timestep for both dimensions are $c_1(p_1^i - x_1^i)$ and $c_1(p_2 - x_2)$ as indicated. The possible positions that can be occupied by the particle during the next timestep will form a line from the current particle position ($r_1 = 0$) toward $\boldsymbol{p}^i$ as r_1 is increased. This line will extend beyond $\boldsymbol{p}^i$ if $c_1 > 1$, allowing for the possibility of the particle overshooting.

Similarly, if we consider the social component of (2.2):

$$c_2 r_2 (\boldsymbol{p}_k^g - \boldsymbol{x}_k^i), \tag{2.4}$$

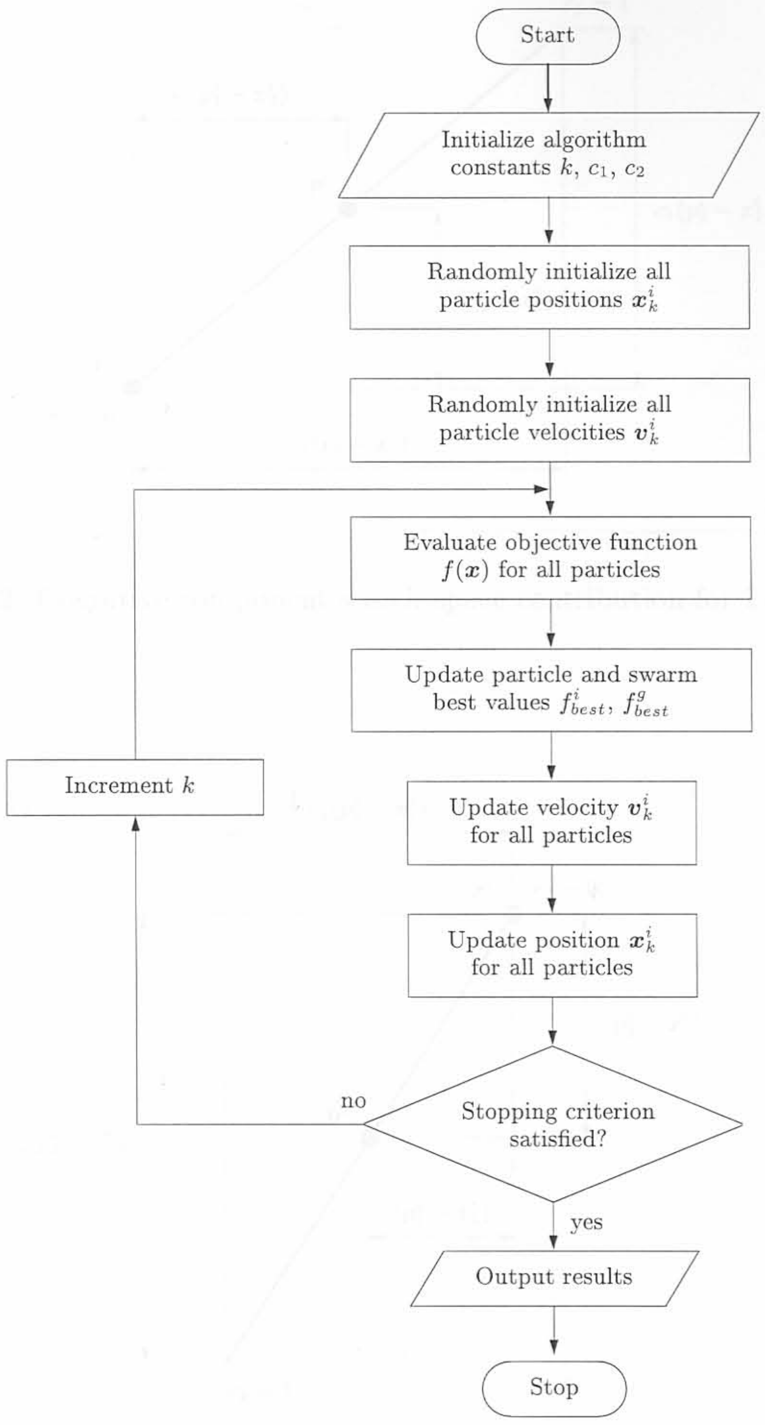
Start
Initialize algorithm constants k, c_1, c_2
Randomly initialize all particle positions $\boldsymbol{x}_k^i$
Randomly initialize all particle velocities $\boldsymbol{v}_k^i$
Evaluate objective function $f(\boldsymbol{x})$ for all particles
Update particle and swarm best values f_{best}^i, f_{best}^g
Update velocity $\boldsymbol{v}_k^i$ for all particles
Update position $\boldsymbol{x}_k^i$ for all particles
Stopping criterion satisfied?
no
Increment k
yes
Output results
Stop

Figure 2.1: Flow analysis of the classical particle swarm optimization algorithm

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

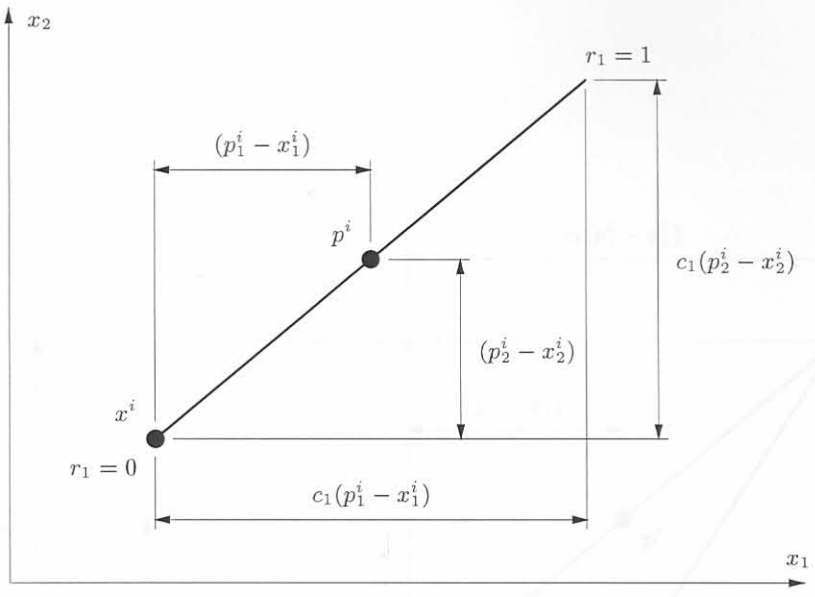

Figure 2.2: Cognitive component search space contribution for 2-D problem

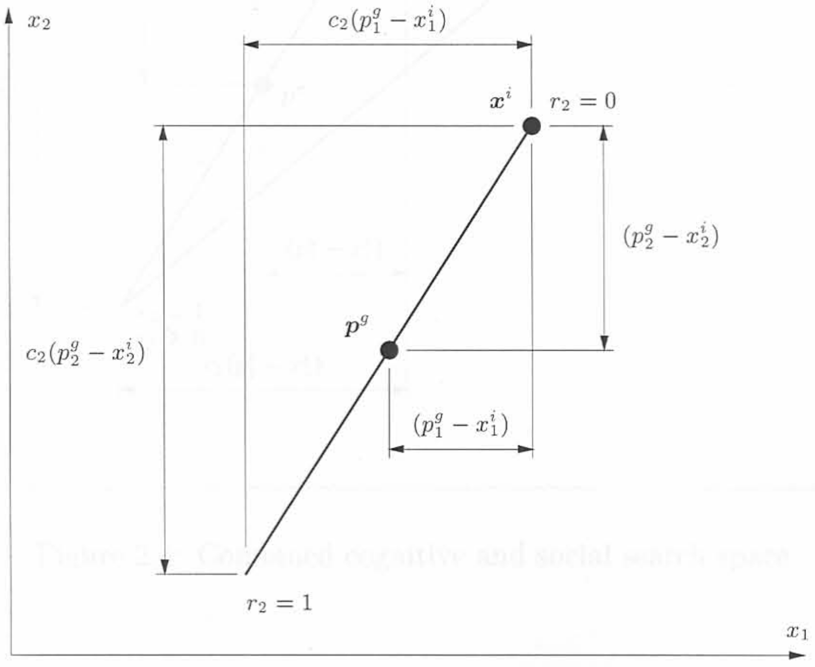

Figure 2.3: Social component search space contribution for 2-D problem

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

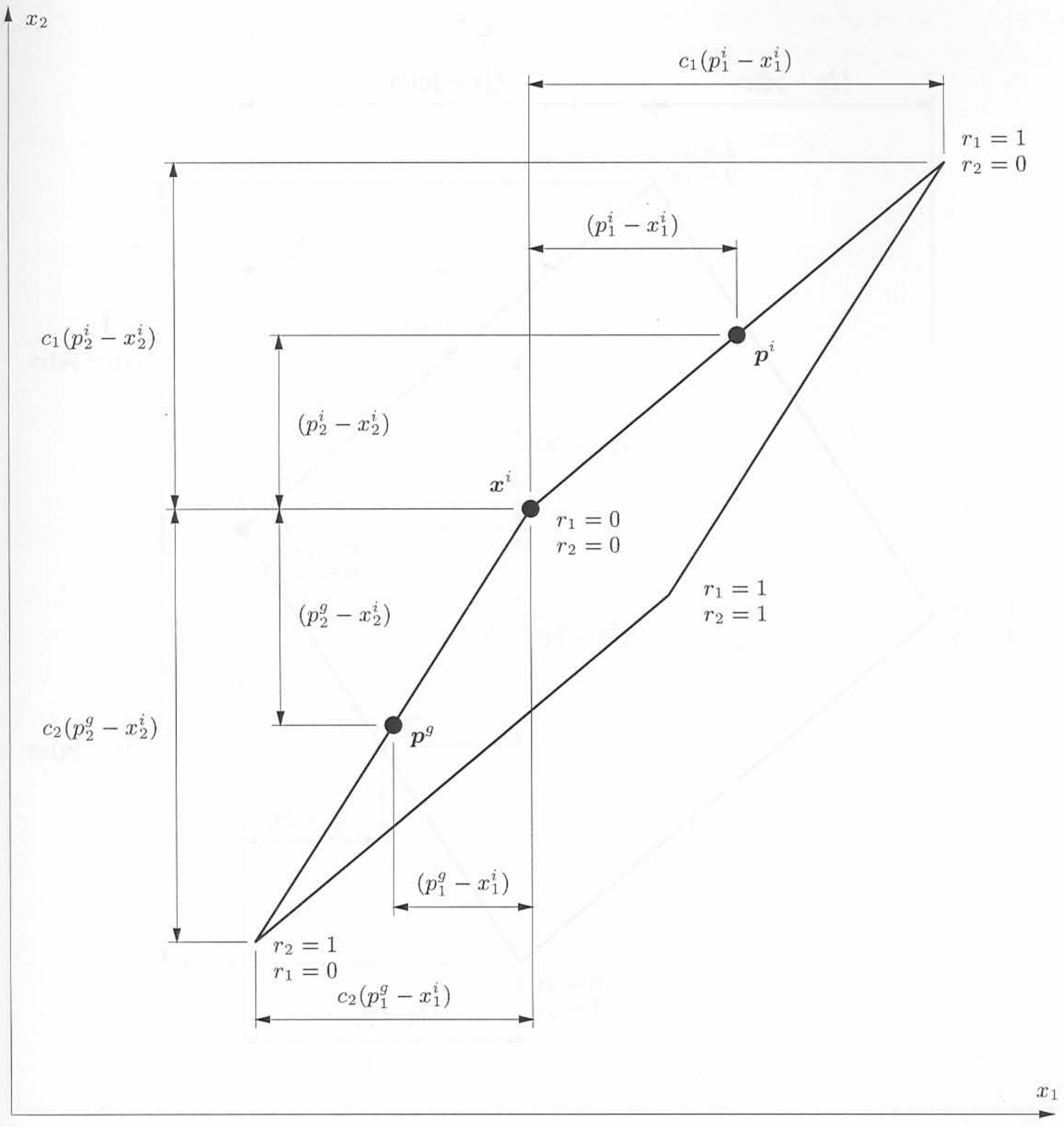

Figure 2.4: Combined cognitive and social search space

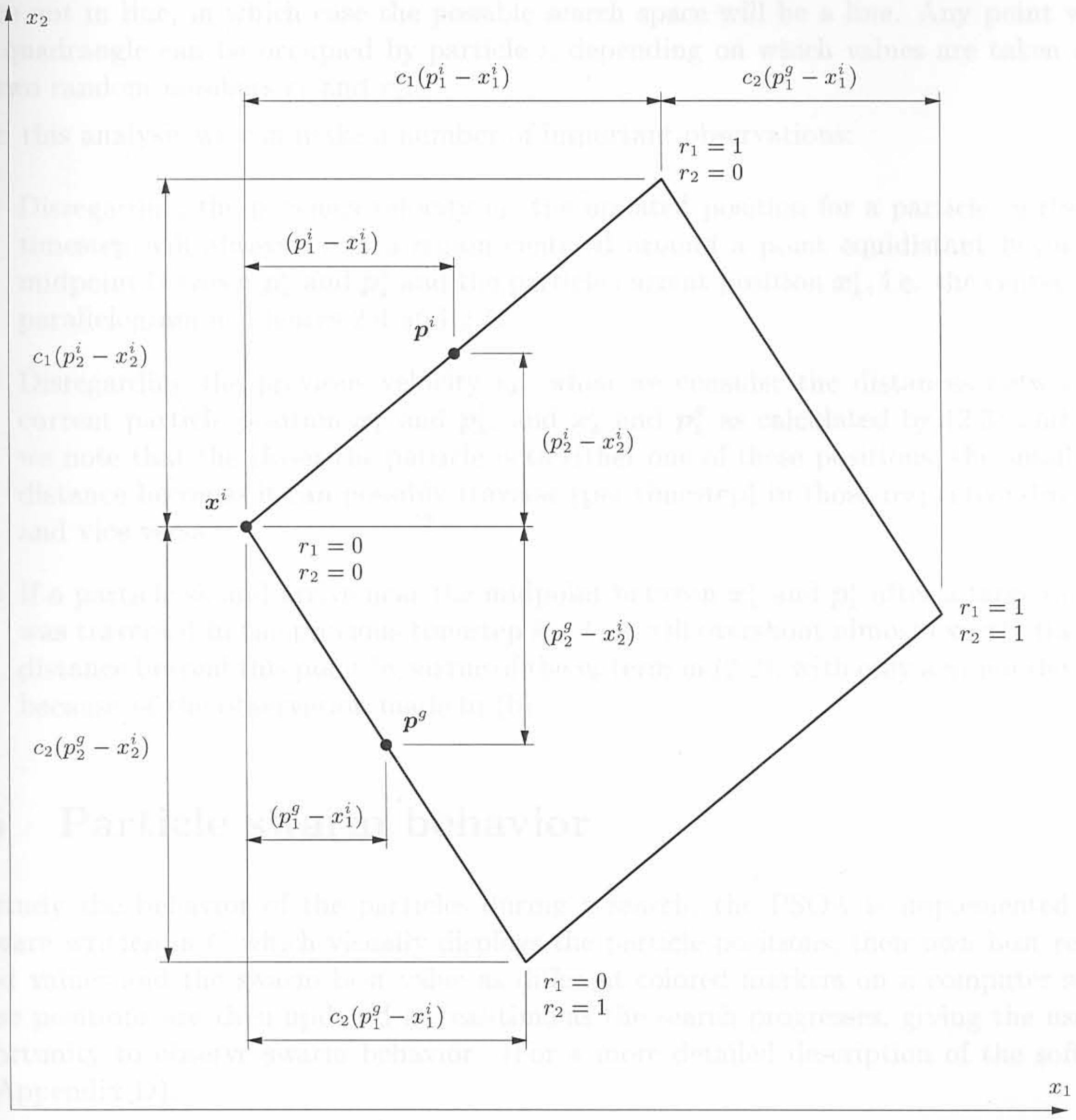

Figure 2.5: Alternative case of combined cognitive and social search space

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

we see that the maximum distance in this case becomes $c_2(p_1^g - x_1)$ and $c_2(p_2^g - x_2)$, depicted by Figure 2.3. The possible positions the particle i could occupy in the next timestep k is again a line originating from the current position toward $\boldsymbol{p}^g$. This line will also extend beyond $\boldsymbol{p}^g$ if $c_2 > 1$.

If we combine the cognitive and social search contributions we obtain the search area in Figure 2.4. This search area will form a parallelogram in 2-dimensional space if p^g, p^i and x^i are not in line, in which case the possible search space will be a line. Any point within this quadrangle can be occupied by particle i, depending on which values are taken on by the two random numbers r_1 and r_2.

From this analysis we can make a number of important observations:

(a) Disregarding the previous velocity v_k, the updated position for a particle in the next timestep will always be in a region centered around a point equidistant beyond the midpoint between $\boldsymbol{p}_k^i$ and $\boldsymbol{p}_k^g$ and the particle current position $\boldsymbol{x}_k^i$, i.e. the center of the parallelogram in Figures 2.4 and 2.5.

(b) Disregarding the previous velocity v_k, when we consider the distances between the current particle position $\boldsymbol{x}_k^i$ and $\boldsymbol{p}_k^i$, and $\boldsymbol{x}_k^i$ and $\boldsymbol{p}_k^g$ as calculated by (2.3) and (2.4), we note that the closer the particle is to either one of these positions, the smaller the distance becomes it can possibly traverse (per timestep) in those respective directions and vice versa.

(c) If a particle should arrive near the midpoint between $\boldsymbol{x}_k^i$ and $\boldsymbol{p}_k^i$ after a large distance was traversed in the previous timestep $k-1$, it will overshoot almost exactly the same distance beyond this point by virtue of the v_k term in (2.2), with only a minor deviation because of the observation made in (b).

2.5 Particle swarm behavior

To study the behavior of the particles during a search, the PSOA is implemented using software written in C which visually displays the particle positions, their own best remembered values and the swarm best value as different colored markers on a computer screen. These positions are then updated in real-time as the search progresses, giving the user the opportunity to observe swarm behavior. (For a more detailed description of the software, see Appendix D).

For the original PSOA, as formulated in this chapter, it was observed herein that the swarm best position $\boldsymbol{p}_k^g$ usually settles very quickly near the global optimum after jumping around in the problem space D during the initial timesteps. Any outlying particles then quickly distribute themselves evenly around this position. Depending on the nature of the problem either one of the following scenario's take place:

For convex optimization problems, such as the sphere function, the group best position $\boldsymbol{p}_k^g$ will start moving downslope as individual particles in the immediate region around $\boldsymbol{p}_k^g$ find improved fitness values and their best values $\boldsymbol{p}_k^i$ become $\boldsymbol{p}_k^g$. Throughout this movement

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

of $\boldsymbol{p}_k^g$ the swarm redistributes itself around this centerpoint. Once $\boldsymbol{p}_k^i$ reaches the minima's immediate neighborhood the particle best remembered positions $\boldsymbol{p}_k^i$ rapidly start converging toward $\boldsymbol{p}_k^g$, and the overall swarm diameter contracts. This leads to a progressively smaller area being searched, and the eventual convergence of the swarm toward the minima.

For non-convex functions or functions with excessive numerical noise however, the contraction rate of $\boldsymbol{p}_k^i$ toward $\boldsymbol{p}_k^g$ is either extremely slow or nonexistent. This causes the PSOA either to become very expensive in terms of computational effort (cost), and sometimes prevents convergence.

2.6 Summary

From the swarm behavior discussed in the foregoing, it is clear that some artificial means of contracting the particle swarm diameter needs to be effected by modifying or introducing a new operator into the standard PSOA. This enforcement of a progressively smaller search space for multi-modal or non-convex problems will force a localized search around the best remembered particle position and ultimately lead to convergence. Several methods of forcing progressively smaller search spaces with PSOA will be investigated in the next chapter.

An alternative is to hybridize the particle swarm with an efficient gradient based search algorithm which will perform the local search after the PSOA has found the approximate region of the minima. However, then another difficulty then arises, that of deciding when the particle swarm should be stopped and the local search algorithm started. On one side, a premature transition may lead to the gradient based algorithm converging in a local minima where more extensive search by the PSOA may have found the approximate region of the global minima, and on the other a late transition will lead to wasted function evaluations.

For the purpose of this thesis the PSOA will be used during both the global and local search phases, because it has the ability to perform adequately, and in some cases very well, for both stages. Also, since it is the intent of the author to obtain a comparison of the different variants of the PSOA in the next chapter, it will be desireable to do so without the influence of a gradient based method incorporated in the optimizer.