

Chapter 2

Glowworm Swarm Optimization: Algorithm Development

In this chapter, the development of the glowworm swarm optimization (GSO) algorithm is presented. Initially, the basic working principle of GSO is introduced, which is followed by a description of the phases that constitute each cycle of the algorithm. GSO, in its present form, has evolved out of several significant modifications incorporated into the earlier versions of the algorithm. Many ideas were considered in the development process before converging upon the current GSO version. Some of the important steps in this evolution are briefly discussed. Next, some convergence proofs related to the luciferin update rule of GSO are provided. Later, simulations are used to illustrate the basic working of GSO. Finally, some comparisons of GSO with ACO and PSO are provided based on their underlying principles, algorithmic aspects, and applications. Experimental comparisons of GSO with a variant of PSO are deferred to Chap. 4.

2.1 The Glowworm Swarm Optimization (GSO) Algorithm

GSO starts by distributing a swarm of agents randomly in the search space. Agents are modeled after glowworms and, hereafter, they will be called glowworms. Further, they are endowed with other behavioral mechanisms that are not found in their natural counterparts. Accordingly, the basic working of the algorithm is the result of an interplay between the following three mechanisms:

1. **Fitness broadcast:** Glowworms carry a luminescent pigment called *luciferin*, whose quantity encodes the fitness of their locations in the objective space. This allows them to glow at an intensity that is proportional to the function value being

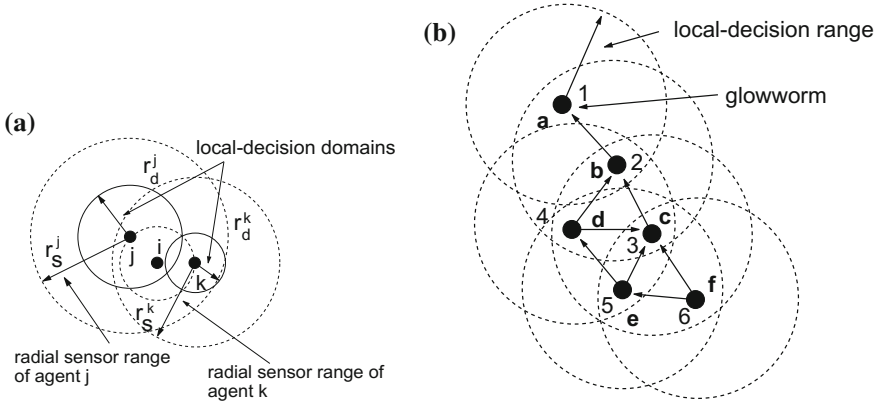


Fig. 2.1 **a** $r_d^k < d(i, k) = d(i, j) < r_d^j < r_s^k < r_s^j$. Agent i is in the sensor range of (and is equidistant to) both j and k . But, they have different decision-domains. Hence, only j uses the information of i . **b** Emergence of a directed graph based on the relative luciferin level of each agent and availability of only local information. Agents are ranked according to the increasing order of their luciferin values. For instance, the agent a whose luciferin value is highest is ranked '1' in the figure

optimized. It is assumed that the luciferin level of a glowworm as sensed by its neighbor does not reduce due to distance.¹

2. **Positive taxis:** Each glowworm is attracted by, and moves toward, a single neighbor whose glow is brighter than that of itself; when surrounded by multiple such neighbors, it uses a probabilistic mechanism (described in Sect. 2.1.1) to select one of them.
3. **Adaptive neighborhood:** Each glowworm uses an adaptive neighborhood to identify neighbors; it is defined by a local-decision domain that has a variable range r_d^i bounded by a hard-limited sensor range r_s ($0 < r_d^i \leq r_s$). A suitable heuristic is used to modulate r_d^i (described in Sect. 2.1.1). A glowworm i considers another glowworm j as its neighbor if j is within the neighborhood range of i and the luciferin level of j is higher than that of i .

Note that the glowworms depend only on information available in the local-decision domain to decide their movements. For instance, in Fig. 2.1a, glowworm i is in the sensor range of (and is equidistant to) both j and k . However, j and k have different neighborhood sizes, and only j uses the information of i . Figure 2.1b shows the formation of a directed graph based on the relative luciferin level of each agent and on the availability of only local information. Each glowworm selects, using a probabilistic mechanism, a neighbor that has a luciferin value higher than its own and moves toward it. These movements, that are based only on local information and selective neighbor interactions, enable the swarm of glowworms to partition

¹In natural glowworms, the brightness of a glowworm's glow as perceived by its neighbor reduces with increase in the distance between the two glowworms.

into disjoint subgroups that steer toward, and meet at, multiple optima of a given multimodal function.

The significant difference between GSO and most earlier approaches to multimodal function optimization problems is the adaptive local-decision domain, which is used effectively to locate multiple peaks.

2.1.1 Algorithm Description

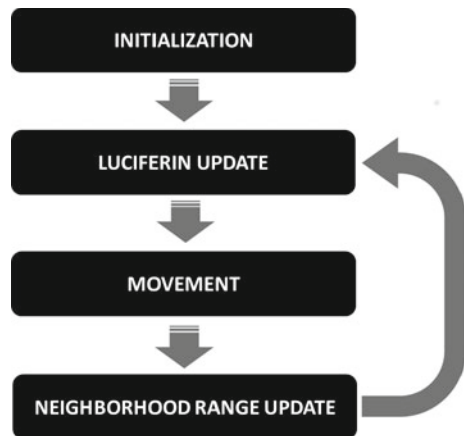
The exposition of the algorithm is presented for maximization problems. However, the algorithm can be easily modified and used to find multiple minima of multimodal functions. GSO starts by placing a population of n glowworms randomly in the search space so that they are well dispersed. Initially, all the glowworms contain an equal quantity of luciferin ℓ_0 . Each cycle of the algorithm consists of a luciferin update phase, a movement phase, and a neighborhood range update phase (Fig. 2.2). The GSO algorithm is given in Fig. 2.3.

Luciferin update phase: The luciferin update depends on the function value at the glowworm position. During the luciferin-update phase, each glowworm adds, to its previous luciferin level, a luciferin quantity proportional to the fitness of its current location in the objective function space. Also, a fraction of the luciferin value is subtracted to simulate the decay in luciferin with time. The luciferin update rule is given by:

$$\ell_i(t+1) = (1 - \rho)\ell_i(t) + \gamma J(x_i(t+1)) \quad (2.1)$$

where, $\ell_i(t)$ represents the luciferin level associated with glowworm i at time t , ρ is the luciferin decay constant ($0 < \rho < 1$), γ is the luciferin enhancement constant

Fig. 2.2 The phases that constitute each cycle of the GSO algorithm



GLOWWORM SWARM OPTIMIZATION (GSO) ALGORITHM

```

Set number of dimensions =  $m$ 
Set number of glowworms =  $n$ 
Let  $s$  be the step size
Let  $x_i(t)$  be the location of glowworm  $i$  at time  $t$ 
deploy_agents_randomly;
for  $i = 1$  to  $n$  do  $\ell_i(0) = \ell_0$ 
 $r_d^i(0) = r_0$ 
set maximum iteration number =  $iter\_max$ ;
set  $t = 1$ ;
while ( $t \leq iter\_max$ ) do:
{
    for each glowworm  $i$  do: % Luciferin-update phase
         $\ell_i(t) = (1 - \rho)\ell_i(t - 1) + \gamma J(x_i(t))$ ; % See Eq. (2.1)

    for each glowworm  $i$  do: % Movement-phase
    {
         $N_i(t) = \{j : d_{ij}(t) < r_d^i(t); \ell_i(t) < \ell_j(t)\}$ ;
        for each glowworm  $j \in N_i(t)$  do:
             $p_{ij}(t) = \frac{\ell_j(t) - \ell_i(t)}{\sum_{k \in N_i(t)} \ell_k(t) - \ell_i(t)}$ ; % See Eq. (2.2)
             $j = select\_glowworm(\vec{p})$ ;
             $x_i(t + 1) = x_i(t) + s \left( \frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|} \right)$  % See Eq. (2.3)
             $r_d^i(t + 1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\}\}$ ; % See Eq.
(2.4)
        }
         $t \leftarrow t + 1$ ;
    }
}

```

Fig. 2.3 The GSO algorithm

and $J(x_i(t))$ represents the value of the objective function at glowworm i 's location at time t .

Movement phase: During the movement phase, each glowworm decides, using a probabilistic mechanism, to move toward a neighbor that has a luciferin value higher than its own. That is, glowworms are attracted to neighbors that glow brighter. Figure 2.1b shows the directed graph among a set of six glowworms based on their relative luciferin levels and availability of only local information. For instance, there are four glowworms (a , b , c , and d) that have relatively higher luciferin level than glowworm e . Since e is located in the sensor-overlap region of c and d , it has only two possible directions of movement. For each glowworm i , the probability of moving

toward a neighbor j is given by:

$$p_{ij}(t) = \frac{\ell_j(t) - \ell_i(t)}{\sum_{k \in N_i(t)} \ell_k(t) - \ell_i(t)} \quad (2.2)$$

where, $j \in N_i(t)$, $N_i(t) = \{j : d_{ij}(t) < r_d^i(t); \ell_i(t) < \ell_j(t)\}$ is the set of neighbors of glowworm i at time t , $d_{ij}(t)$ represents the Euclidean distance between glowworms i and j at time t , and $r_d^i(t)$ represents the variable neighborhood range associated with glowworm i at time t . Let glowworm i select a glowworm $j \in N_i(t)$ with $p_{ij}(t)$ given by (2.2). Then, the discrete-time model of the glowworm movements can be stated as:

$$x_i(t+1) = x_i(t) + s \left(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|} \right) \quad (2.3)$$

where, $x_i(t) \in R^m$ is the location of glowworm i , at time t , in the m -dimensional real space R^m , $\|\cdot\|$ represents the Euclidean norm operator, and s (>0) is the step size.

Neighborhood range update phase: Each agent i is associated with a neighborhood whose radial range r_d^i is dynamic in nature ($0 < r_d^i \leq r_s$). The fact that a fixed neighborhood range is not used needs some justification. When the glowworms depend only on local information to decide their movements, it is expected that the number of peaks captured would be a function of the radial sensor range. In fact, if the sensor range of each agent covers the entire search space, all the agents move to the global optimum and the local optima are ignored. Since we assume that a priori information about the objective function (e.g., number of peaks and inter-peak distances) is not available, it is difficult to fix the neighborhood range at a value that works well for different function landscapes. For instance, a chosen neighborhood range r_d would work relatively better on objective functions where the minimum inter-peak distance is more than r_d rather than on those where it is less than r_d . Therefore, GSO uses an adaptive neighborhood range in order to detect the presence of multiple peaks in a multimodal function landscape.

Let r_0 be the initial neighborhood range of each glowworm (that is, $r_d^i(0) = r_0 \forall i$). To adaptively update the neighborhood range of each glowworm, the following rule is applied:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\}\} \quad (2.4)$$

where, β is a constant parameter and n_t is a parameter used to control the number of neighbors.

The quantities ρ , γ , s , β , n_t , and ℓ_0 are algorithm parameters for which appropriate values have been determined based on extensive numerical experiments and are kept fixed in this book (Table 2.1). The quantity r_0 is made equal to r_s in all the experiments. Thus, n and r_s are the only parameters that influence the algorithm behavior (in terms of the total number of peaks captured) and need to be selected.

Table 2.1 Values of algorithmic parameters that are kept fixed for all the experiments

ρ	γ	β	n_t	s	ℓ_0
0.4	0.6	0.08	5	0.03	5

The following example is used to demonstrate the variable nature of the neighborhood. For the purpose of simplicity, a static placement of glowworms is considered and it is observed how the neighborhood range of the glowworm at $(0, 0)$ varies according to (2.4). In Figs. 2.4 and 2.5, notice that the neighborhood range gets adjusted until the glowworm acquires the number of neighbors that is specified by the parameter n_t ($=3$). Figure 2.4a shows an initial placement where the glowworm at $(0, 0)$ is isolated. It increases its neighborhood range (Fig. 2.4b) until it either acquires a set of three neighbors or reaches the maximum range. In Fig. 2.5a, the glowworm is crowded by a large number of neighbors ($|N_i(t)| > n_t$) that causes the neighborhood range to shrink until $|N_i(t)| = n_t$. Figure 2.5b shows the variation of the neighborhood range as a function of the iteration number.

When a glowworm is located far away from a peak, typically it has only a few neighbors or it may even be isolated (as in Fig. 2.4a). Rule (2.4) aids a glowworm in this situation to find neighbors by increasing its neighborhood range at each iteration. In contrast, when the glowworm is located in the vicinity of multiple peaks, it is more likely that the glowworm is surrounded by a large number of neighbors (as in Fig. 2.5a). Consequently, rule (2.4) restricts its visibility to a few neighbors so that its movement is biased toward a nearby peak.

2.1.2 Evolving Graph Architecture of GSO

The constraints of sensor range r_s and adaptive decision-range r_d^i , that are imposed on each glowworm, give rise to two different graphs²—an undirected graph N_c and a directed graph N_d —of the same set of glowworms. From the algorithm's description, it is clear that agents in GSO do not maintain fixed-neighbors during their movements, which means that new links may form, and existing links may break, between any two glowworms. Therefore, the graphs N_c and N_d are dynamic in nature. The example in Fig. 2.6 is used to describe the evolving graph architecture of agents in a glowworm swarm.

A group of 16 glowworms are randomly deployed in a search space that consists of three sources placed at different locations. Note from Fig. 2.6 that the graph N_c is connected. If the glowworms use a constant local-decision domain whose range is

²Let $G(V, E)$ be a graph with vertex set $V = \{v_1, \dots, v_n\}$ and edge set $E = \{(v_i, v_j) : v_i, v_j \in V\}$. If E is a set of unordered pairs, then G is said to be an undirected graph. If E is a set of ordered pairs, then G is said to be a directed graph. The graph G is said to be connected if it has a path between each distinct pair of vertices v_i and v_j where by a path (of length m) is meant a sequence of distinct edges of G of the form $(v_i, k_1), (k_1, k_2), \dots, (k_m, v_j)$.

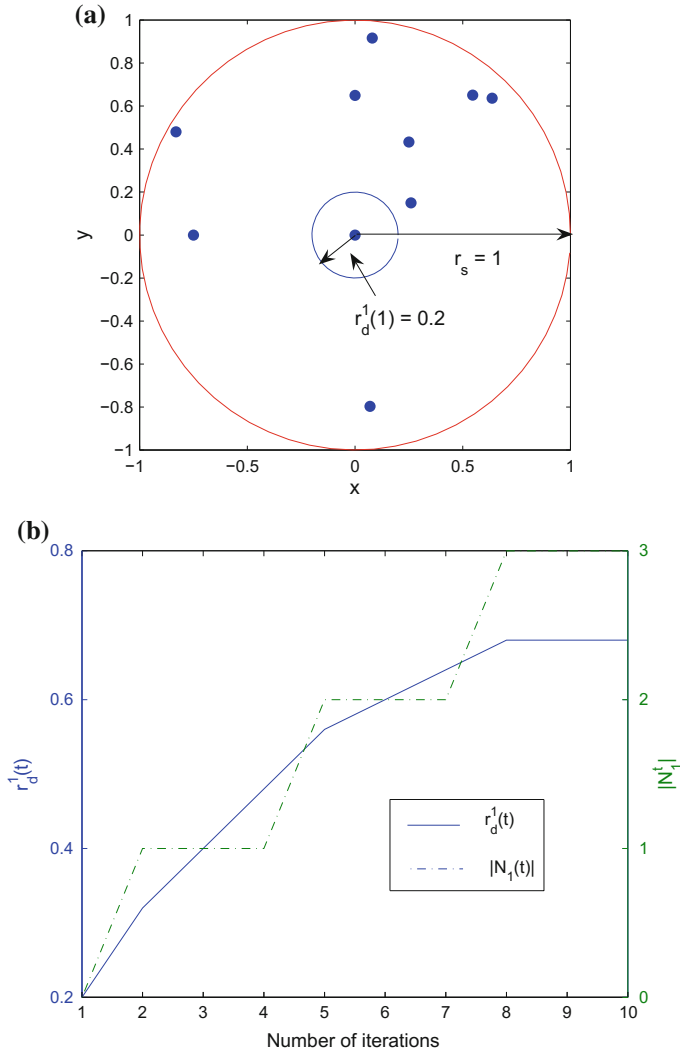


Fig. 2.4 **a** Initial placement where the glowworm at (0, 0) is isolated. **b** Plot of $r_d^1(t)$. Values of $r_s = 1$ and $n_t = 3$ are used

equal to the maximum sensing range r_s , all the glowworms converge to the global peak. This observation is supported by the simulation example presented in Sect. 2.4. However, note that the graph N_d is partitioned into two disjoint weakly connected components N_d^1 and N_d^2 , which can be explained in the following way. When the glowworms use an adaptive decision-domain whose range is updated according to (2.4), the glowworms adjust their neighborhood ranges until they acquire a pre-specified number of neighbors ($n_t = 2$ in the example shown in Fig. 2.6). This property enables

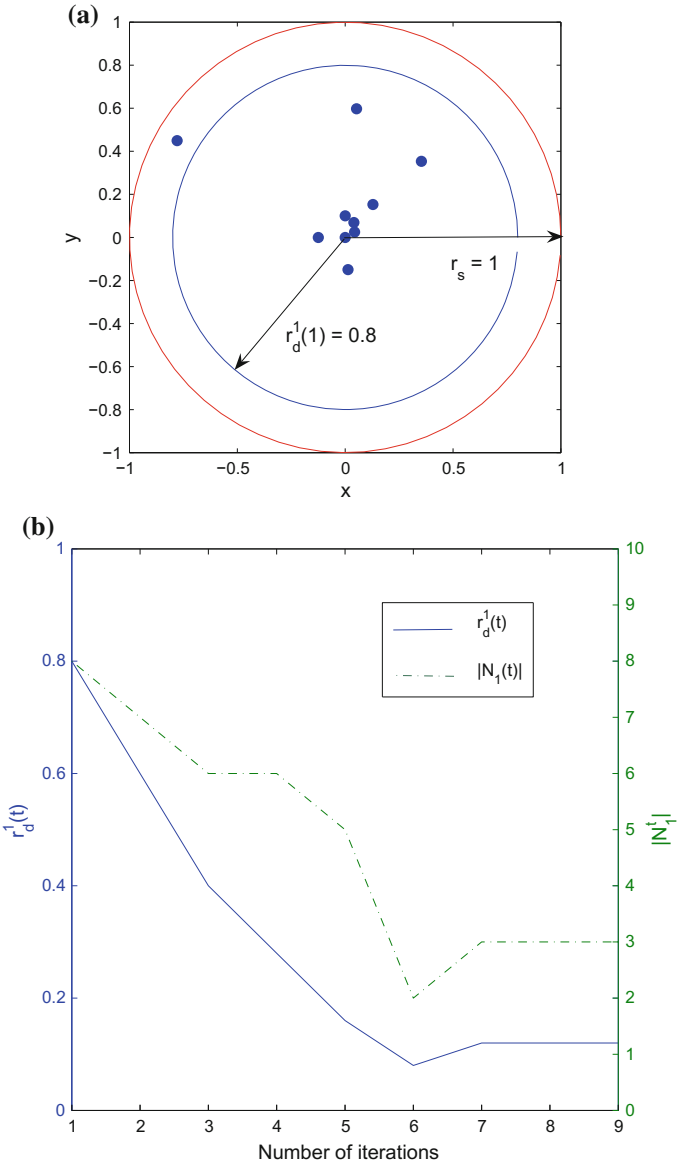


Fig. 2.5 **a** Initial placement where the glowworm at (0, 0) is crowded by a large number of glowworms. **b** Plot of $r_d^1(t)$. Values of $r_s = 1$ and $n_l = 3$ are used

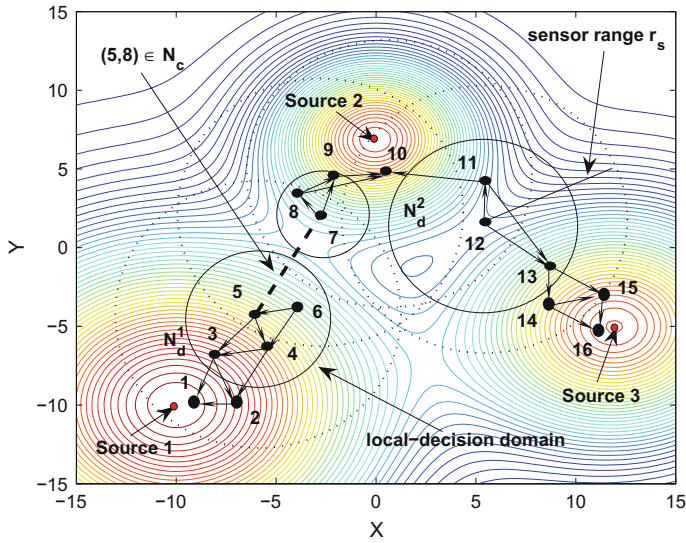


Fig. 2.6 Networks resulting from the constraints of sensing range r_s and adaptive decision-range r_d^i that are imposed on the agents in GSO

each glowworm to select its neighbors so that its movements get biased toward the nearest peak. This individual agent behavior results in a collective behavior of agents comprising an automatic splitting of the whole group into disjoint subgroups and the eventual convergence of each subgroup of agents to a nearby peak. In Fig. 2.6, glowworms 5, 6, 8, 9, and 10 are within the sensing-range of glowworm 7. However, glowworm 7 considers only glowworms 8 and 9 as neighbors. Therefore, its movements get biased toward Source 2, while it avoids moving toward glowworm 5, even though glowworm 5 has a higher luciferin value than that of itself. Accordingly, the subgroups of glowworms $\{1, 2, 3, 4, 5, 6\}$, $\{7, 8, 9, 10\}$, and $\{13, 14, 15, 16\}$ move toward Source 1, Source 2, and Source 3, respectively. Note that glowworms 11 and 12 may move toward either Source 2 or Source 3. The local-search and convergence of each subgroup to a nearby source location is achieved by the *leapfrogging* effect inherent in GSO, which is explained next.

2.1.3 Leapfrogging Behavior Enables Local Search

According to the basic GSO algorithm, in any given iteration the glowworm with the maximum luciferin remains stationary. The above property may lead to a dead-lock situation where all the glowworms in the vicinity of a peak converge to the glowworm

that is located closest to the peak. Since the agent movements are restricted to the interior region of the convex-hull, all the glowworms converge to a glowworm that attains maximum luciferin value during its movements within the convex-hull. As a result, all the glowworms are trapped away from the peak. However, the discrete nature of the movement update rule automatically takes care of this problem. In fact, during the movement phase, each glowworm moves a distance of finite step size s toward a neighbor. Hence, when the distance between a glowworm i approaching a neighbor j is less than s , i leapfrogs over the position of j and becomes a leader to j . In the next iteration, i remains stationary and j overtakes the position of i thus regaining its leadership. This process of role interchange between i and j repeats, giving rise to a local-search behavior of the glowworm pair along a single ascent direction. A group of glowworms uses the same principle to perform an improved local-search and eventually converge to the peak location. The leapfrogging behavior of the agents in GSO can be observed in simulations in Sect. 2.4 (Fig. 2.8d).

2.2 Evolution of GSO

GSO, in its present form, has evolved out of several significant modifications incorporated into the earlier versions of the algorithm [103, 105, 107, 109, 112] that led to improvement in algorithmic performance from one version to the next. Some of the important steps in this evolution are briefly discussed for the sake of completion and also to convey the fact that many ideas were considered in the process of developing the GSO algorithm before converging upon the current version.

The algorithm was first introduced in [105]. The equations that modelled the luciferin-update, probability distribution used to select a neighbor, movement update, and local-decision range update are given below:

$$\ell_i(t+1) = \max\{0, (1 - \rho)\ell_i(t) + \gamma J(x_i(t+1))\} \quad (2.5)$$

$$p_j(t) = \frac{\ell_j(t)}{\sum_{k \in N_i(t)} \ell_k(t)} \quad (2.6)$$

$$x_i(t+1) = x_i(t) + s \left(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|} \right) \quad (2.7)$$

$$r_d^i(t+1) = \frac{r_s}{1 + \beta D_i(t)} \quad (2.8)$$

where,

$$D_i(t) = \frac{N_i(t)}{\pi r_s^2} \quad (2.9)$$

is the neighbor-density of agent i at iteration t and β is a constant parameter.

The local-decision range update rule (2.8) faces a problem at an instant when a glowworm i has no neighbors in its current local-decision domain range $r_d^i(t)$ but has some neighbors within its sensor range r_s . In particular, when the glowworm i is isolated, $D_i(t) = 0$. From (2.8), $r_d^i(t+1) = r_s$. Suppose glowworm i acquires some neighbors at $t+1$, $D_i(t+1) \neq 0$. If a large value of β is used, (2.8) gives $r_d^i(t+2) \approx 0$. Therefore, the above kind of neighbor-distribution associated with i results in an oscillatory behavior of the decision range, with r_d^i switching between r_s and a value closer to zero. Therefore, all glowworms in a neighborhood situation as described above move only in alternative iterations slowing down the convergence of the algorithm approximately by a factor of two.

To solve the above problem, the local-decision update rule in [106] is modified by forcing a nonzero lower bound on the decision range such that each glowworm improves its own chances of finding a neighbor at every instant. The modified update rule for $r_d^i(t)$ is given by:

$$r_d^i(t+1) = \alpha + \frac{r_s - \alpha}{1 + \beta N_i(t)} \quad (2.10)$$

where, α represents the lower bound of the decision domain range.

A new update rule is proposed in [104] where an explicit threshold parameter n_t is used to control the number of neighbors at each iteration. A substantial enhancement in performance is noticed by using this rule:

$$r_d^i(t+1) = \begin{cases} r_d^i(t) + \beta_1 |N_i(t)|, & \text{if } |N_i(t)| \leq n_t \\ r_d^i(t) - \beta_2 |N_i(t)|, & \text{otherwise} \end{cases} \quad (2.11)$$

where, β_1 and β_2 are constant parameters.

A further improvement in algorithmic performance is observed by using the present decision-range update rule (2.4), which was introduced in [107]. This can be attributed to the fact that the second term in (2.4), which either increments or decrements $r_d^i(t)$, is proportional to the difference between the desired number of neighbors n_t and the actual number of neighbors $|N_i(t)|$.

Since actual values of luciferin (instead of differences in luciferin values as used in the current version) are used in the probability distribution formula (2.6), the luciferin cannot take negative values. This is taken care of in the luciferin update formula (2.5), by forcing a zero lower bound on the luciferin value. However, the algorithm does not work in regions where the objective function has negative values unless the function is shifted appropriately. In order to address these problems, actual luciferin values in the probability distribution formula are replaced by relative luciferin values in [112]. As a consequence, the luciferin values are allowed to take negative values. Therefore, the luciferin-update formula was accordingly modified to its current form given in (2.1) where the constraint of zero lower bound on the luciferin values is removed [112].

2.3 Convergence Results

Some theoretical proofs on the working of the luciferin update rule are derived in this section. These theoretical results give an insight into how the luciferin levels of glowworms vary as a function of time during the execution of the algorithm. First, it is proved that, due to luciferin decay, the maximum luciferin level τ_{max} is bounded asymptotically. Secondly, it is shown that the luciferin ℓ_j of all glowworms co-located at a peak X_i converge to the same value ℓ_i^* .

Theorem 2.1 *Assuming that the luciferin update rule in (2.1) is used, the luciferin level $\ell_i(t)$ for any Glowworm i is bounded above asymptotically as follows:*

$$\lim_{t \rightarrow \infty} \ell_i(t) \leq \lim_{t \rightarrow \infty} \ell^{max}(t) = \left(\frac{\gamma}{\rho}\right) J_{max} \quad (2.12)$$

where, J_{max} is the global maximum value of the objective function.

Proof Given that the maximum value of the objective function is J_{max} and the luciferin update rule in (2.1) is used, the maximum possible quantity of luciferin added to the previous luciferin level at any iteration t is γJ_{max} . Therefore, at Iteration 1, the maximum luciferin of any Glowworm i is $(1 - \rho)\ell_0 + \gamma J_{max}$. At Iteration 2, it is $(1 - \rho)^2\ell_0 + [1 + (1 - \rho)]\gamma J_{max}$, and so on. Generalizing the process, at any iteration t , the maximum luciferin $\ell^{max}(t)$ of any Glowworm i is then given by:

$$\ell^{max}(t) = (1 - \rho)^t \ell_0 + \sum_{k=0}^{t-1} (1 - \rho)^k \gamma J_{max} \quad (2.13)$$

Clearly,

$$\ell_i(t) \leq \ell^{max}(t) \quad (2.14)$$

Since $0 < \rho < 1$, from (2.13) we have that

$$\text{as } t \rightarrow \infty, \ell^{max}(t) \rightarrow \left(\frac{\gamma}{\rho}\right) J_{max} \quad (2.15)$$

Using (2.14) and (2.15), we have the result in (2.12). \square

Theorem 2.2 *For all glowworms i co-located at peak-locations X_j^* associated with objective function values $J_j^* \leq J_{max}$ (where, $j = 1, 2, \dots, m$, with m as the number of peaks), if the luciferin update rule in (2.1) is used, then $\ell_i(t)$ increases or decreases monotonically and asymptotically converges to $\ell_j^* = \left(\frac{\gamma}{\rho}\right) J_j^*$.*

Proof According to (2.1), $\ell_i(t) \geq 0$ always. The stationary luciferin ℓ_j^* associated with peak j satisfies the following condition:

$$\ell_j^* = (1 - \rho)\ell_j^* + \gamma J_j^* \Rightarrow \ell_j^* = \left(\frac{\gamma}{\rho}\right) J_j^* \quad (2.16)$$

If $\ell_i(t) < \ell_j^*$ for Glowworm i co-located at peak-location X_j^* , then using (2.16) we have

$$J_j^*(t) > \left(\frac{\rho}{\gamma}\right) \ell_i(t) \quad (2.17)$$

Now,

$$\ell_i(t+1) = (1 - \rho)\ell_i(t) + \gamma J_j^* \quad (2.18)$$

$$\begin{aligned} &> (1 - \rho)\ell_i(t) + \gamma \left(\frac{\rho}{\gamma}\right) \ell_i(t) \\ \Rightarrow \ell_i(t+1) &> \ell_i(t) \end{aligned} \quad (2.19)$$

that is, $\ell_i(t)$ increases monotonically.

Similarly, if $\ell_i(t) > \ell_j^*$ for Glowworm i co-located at peak-location X_j^* , then using (2.16) and (2.18), it is easy to show that

$$\ell_i(t+1) < \ell_i(t) \quad (2.20)$$

that is, $\ell_i(t)$ decreases monotonically.

Now, the convergence of the sequence $\ell_i(t)$ is proved by showing that the fixed point ℓ_j^* of the system in (2.18) is asymptotically stable. From (2.18), the following relation can be deduced:

$$\begin{aligned} \left| \ell_i(t) - \frac{\gamma}{\rho} J_j^* \right| &= (1 - \rho) \left| \ell_i(t-1) - \frac{\gamma}{\rho} J_j^* \right| \\ &= (1 - \rho)^2 \left| \ell_i(t-2) - \frac{\gamma}{\rho} J_j^* \right| \end{aligned} \quad (2.21)$$

Proceeding in a similar way, we get

$$|\ell_i(t) - \ell_j^*| = (1 - \rho)^t |\ell_i(0) - \ell_j^*| \quad (2.22)$$

Therefore,

$$\begin{aligned} \lim_{t \rightarrow \infty} |\ell_i(t) - \ell_j^*| &= \lim_{t \rightarrow \infty} (1 - \rho)^t |\ell_i(0) - \ell_j^*| \\ &= 0, \text{ since } 0 < (1 - \rho) < 1 \end{aligned} \quad (2.23)$$

From (2.23), it is clear that the luciferin $\ell_i(t)$ of Glowworm i , co-located at a peak-location X_j^* , asymptotically converges to ℓ_j^* . \square

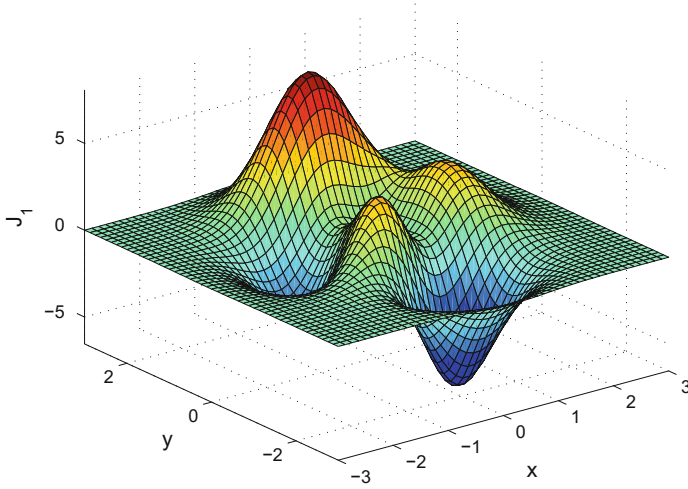


Fig. 2.7 Peaks function $J_1(x, y)$ [184] used in simulations to demonstrate the basic working of GSO. The function has three local maxima located at $(0, 1.58)$, $(-0.46, -0.63)$, and $(1.28, 0)$

2.4 Simulation Experiments to Illustrate GSO

Simulation experiments³ demonstrating the capability of the glowworm algorithm to capture multiple peaks of a number of benchmark multimodal functions are deferred to Chap. 4. Here, the basic working of the algorithm is demonstrated using the *peaks* function, which is a function of two variables, obtained by translating and scaling Gaussian distributions [184]:

$$J_1(x, y) = 3(1 - x)^2 e^{-[x^2 + (y+1)^2]} - 10 \left(\frac{x}{5} - x^3 - y^5 \right) e^{-(x^2 + y^2)} - \left(\frac{1}{3} \right) e^{-[(x+1)^2 + y^2]} \quad (2.24)$$

The peaks function $J_1(x, y)$ has multiple peaks and valleys (Fig. 2.7). Local maxima are located at $(0, 1.58)$, $(-0.46, -0.63)$, and $(1.28, 0)$ with different peak function values. A set of 50 glowworms are randomly deployed in a two-dimensional workspace of size 6×6 square units.

³Movies of some of the simulations presented in this book can be viewed at this link: https://www.youtube.com/watch?v=_vhSu4xBoFs.

2.4.1 Simulation Experiment 1: Constant Local-Decision Domain Range

As a first step, the radial range r_d^i of each glowworm is kept constant, in order to characterize the sensitivity of the number of peaks detected to the size of the local-decision domain. As noted earlier, the local-decision range greatly influences the determination of various peaks. When the decision-range is more than 2, all the glowworms move to the global maximum. Figure 2.8a–c show the emergence of the solution, after 200 iterations, when the local-decision range r_d^i of all glowworms is kept constant at 2 (only one peak is captured), 1.8 (two peaks are captured), and 1.5 (all three peaks are captured), respectively.

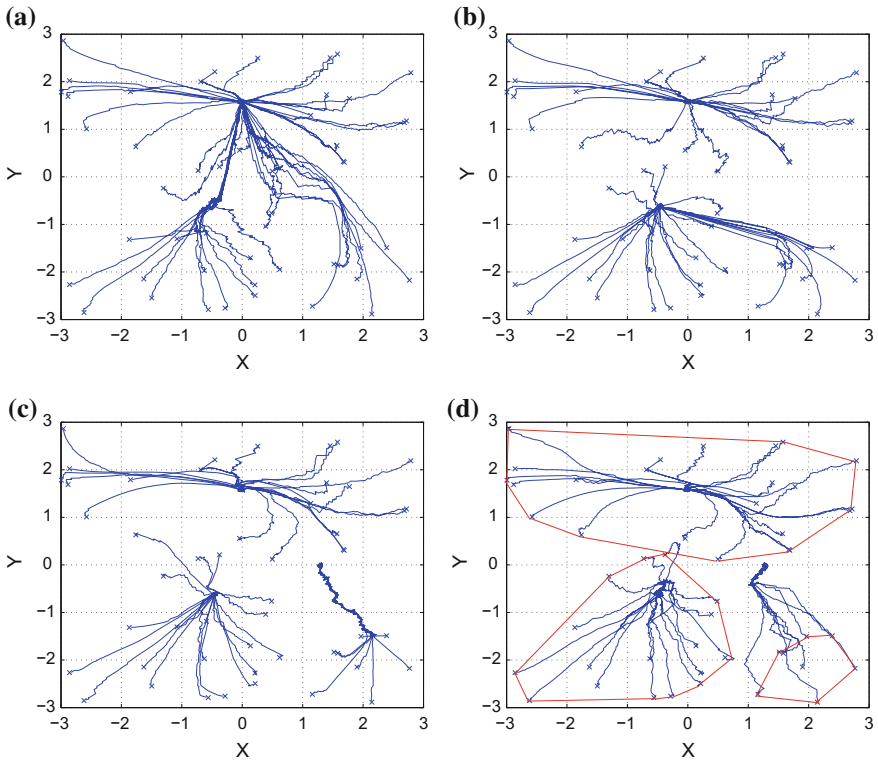


Fig. 2.8 Emergence of solution after 200 iterations for different cases: **a** The decision range is kept constant with $r_d^i = 2$ (only one peak is captured). **b** The decision range is kept constant with $r_d^i = 1.8$ (only two peaks are captured). **c** The decision range is kept constant with $r_d^i = 1.5$ (all three peaks are captured). **d** Decision range is made adaptive according to (2.4), with $r_d^i(0) = 3$ (all three peaks are captured)

2.4.2 Simulation Experiment 2: Adaptive Local-Decision Domain Range

Figure 2.8d shows the emergence of the solution when the local-decision domain range is made to vary according to (2.4) at each iteration t . During this simulation, a value of $r_d^i(0) = 3$ is chosen for each glowworm i . Note that all the peaks are detected within 200 iterations. In particular, 23, 19, and 8 glowworms get co-located at the maxima of $(0, 1.58)$, $(-0.46, -0.63)$, and $(1.28, 0)$, respectively. Figure 2.9a shows the luciferin history of each glowworm. Note that after the steady state is reached, all the glowworms, co-located at a particular location, possess the same luciferin quantity. According to Theorem 2, the value of ℓ_j of all glowworms j , co-located at a peak-location X_i^* , is given by $(\frac{\gamma}{\rho})J_i^*$. It was observed that the value of ℓ at the three peak-locations $(0, 1.58)$, $(-0.46, -0.63)$, and $(1.28, 0)$ are 12.15 $(=(0.6/0.4) \times 8.1)$, 5.67 $(=(0.6/0.4) \times 3.78)$, and 5.4 $(=(0.6/0.4) \times 3.6)$, respectively. Note that the luciferin values obtained in simulations are in exact agreement with their analytical values given by Theorem 2 (Fig. 2.9a).

Figure 2.9b shows the initial placement and co-location of all the glowworms on the equi-contour plot of the objective function. Figure 2.10a, b show the number of neighbors and the local-decision range of glowworm 12, respectively. Initially, the value of $r_d^{12}(t)$ decreases as glowworm 12 has more than five neighbors ($n_t = 5$). Note that between $t = 74$ and $t = 94$, the decision-range rises sharply up to the maximum sensing range r_s . The reason is evident from Table 2.2, where the number of glowworms that are within the decision-range of glowworm 12 is much higher than n_t , but among them, the number of glowworms that have a higher luciferin value (and hence, the number of neighbors) is less than n_t . However, at $t = 94$, the decision-range starts shrinking again, as it acquires a set of 13 neighbors.

2.4.3 Simulation Experiment 3: Effect of Presence of Forbidden Region

GSO has an advantage in situations where presence of forbidden regions in the workspace lead to loss of gradient information and local-gradient based algorithms fail to work. Figure 2.11 depicts a situation where the glowworm A is unaware of the source location because of the occlusion created by the forbidden region O. However, the inter-agent communication between neighbors makes global information available locally to the agent, providing the agent with feasible directions to move toward the source. This situation is simulated by biasing the random initial placement of the glowworms to ensure that none of the glowworms is deployed in a circular (forbidden) region of radius 1 unit centered at $(0.6, -0.6)$. The size of the forbidden region

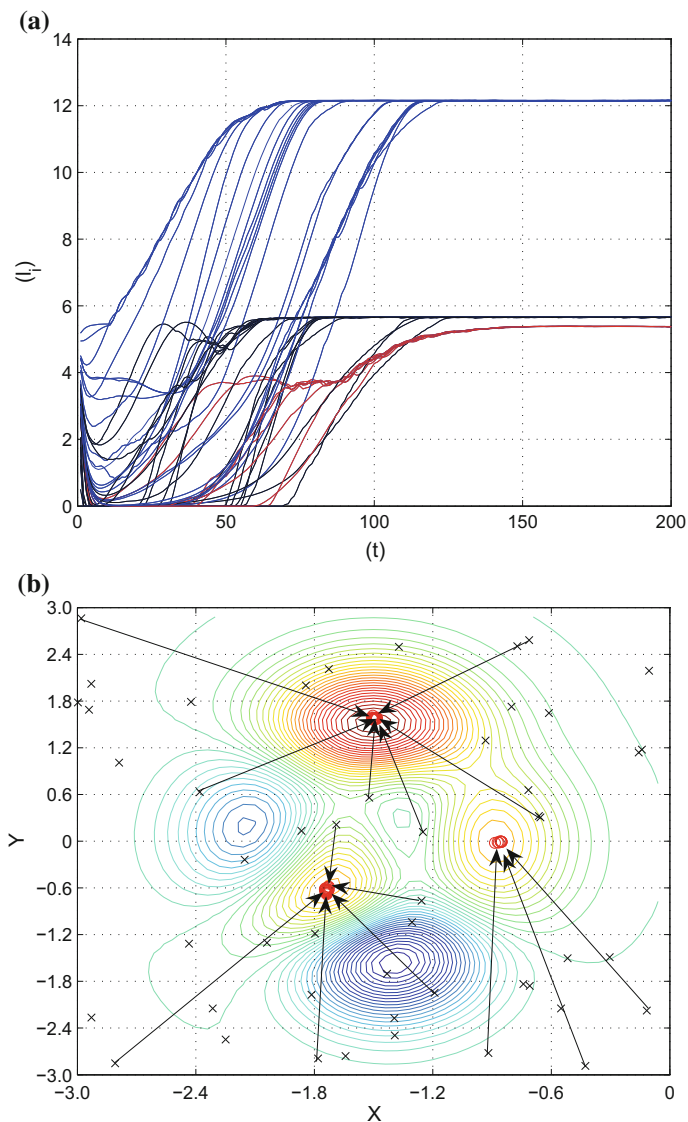


Fig. 2.9 **a** Luciferin-histories of the glowworms. **b** The equi-contour plot with the initial placement (shown by \times -marks) and final co-location (shown by the *three clusters of dots*) of the glowworms at the peak locations

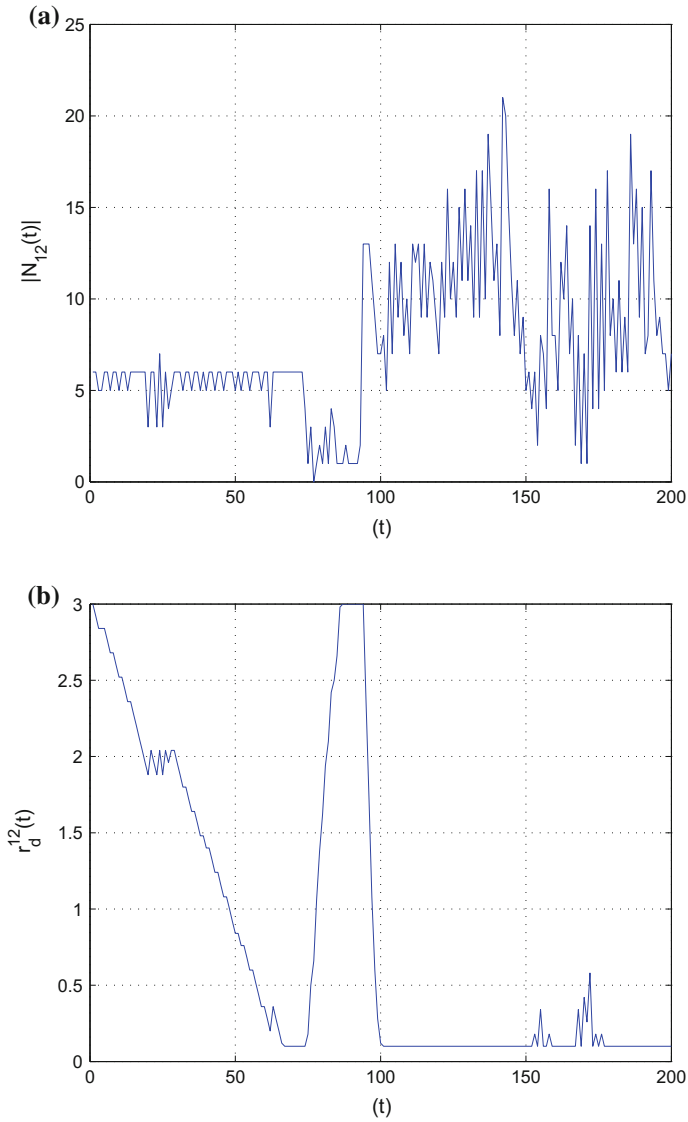


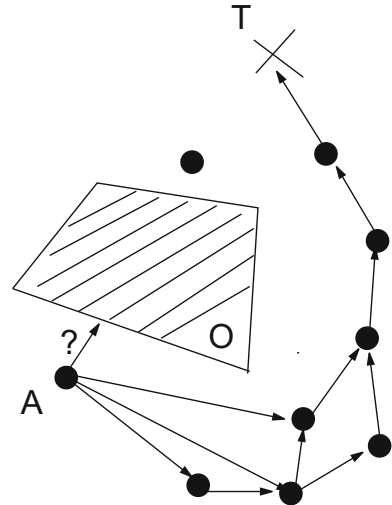
Fig. 2.10 **a** Plot of $|N_{12}(t)|$ as a function of iteration number t . **b** Plot of $r_d^{12}(t)$ as a function of iteration number t

Table 2.2 Positions and luciferin levels of glowworms that are within the local-decision range of glowworm 12 at $t = 74, 80$, and 94 iterations, respectively. The symbol $*$ on the superscript of a luciferin value of glowworm j represents that the corresponding glowworm is a neighbor of glowworm 12 at that instant

Glowworm	$X_i(74)$	$\ell_i(74)$	$X_i(80)$	$\ell_i(80)$	$X_i(94)$	$\ell_i(94)$
1	$(-0.02, 1.60)$	12.059	$(-0.02, 1.55)$	12.144	$(0.01, 1.58)$	12.156*
3	$(-0.03, 1.60)$	12.136*	$(-0.02, 1.59)$	12.149	$(0.00, 1.58)$	12.155*
8	$(-0.04, 1.59)$	12.144*	$(-0.02, 1.58)$	12.144	$(0.00, 1.58)$	12.154*
12	$(-0.02, 1.59)$	12.133	$(-0.01, 1.56)$	12.152	$(-0.02, 1.57)$	12.145
16	$(0.15, 1.60)$	11.470	$(-0.01, 1.57)$	12.070	$(-0.02, 1.57)$	12.150*
19	$(-0.03, 1.60)$	12.142*	$(-0.02, 1.58)$	12.155*	$(-0.02, 1.58)$	12.158*
23	$(0.10, 1.61)$	11.716	$(-0.02, 1.58)$	12.119	$(0.00, 1.57)$	12.153*
24	$(0.13, 1.60)$	11.588	$(-0.03, 1.60)$	12.102	$(-0.02, 1.57)$	12.155*
26	$(0.29, 1.56)$	10.551	$(0.18, 1.58)$	11.575	$(0.00, 1.58)$	12.147*
31	$(-0.03, 1.60)$	12.131	$(-0.02, 1.59)$	12.148	$(-0.01, 1.57)$	12.151*
33	$(0.18, 0.16)$	11.361	$(0.00, 1.58)$	12.038	$(0.00, 1.57)$	12.151*
43	$(-0.04, 1.59)$	12.140*	$(-0.02, 1.59)$	12.146	$(0.00, 1.57)$	12.152*
44	$(0.037, 1.60)$	11.935	$(-0.01, 1.55)$	12.138	$(0.00, 1.58)$	12.153*
45	$(-0.03, 1.60)$	12.124	$(-0.02, 1.60)$	12.141	$(0.01, 1.59)$	12.151*

and sensor range are chosen such that a glowworm located on the boundary of the forbidden region cannot sense another glowworm that is located on the other side of the region. Also, the forbidden region is placed such that the peak located at $(1.28, 0)$ lies within the forbidden region. A constant local-decision range ($r_d^i(t) = 1.5, \forall t$) that is equal to the maximum sensor range is used. The simulation result in Fig. 2.12a shows that, following the biased-random deployment as described above, there is no instance when a glowworm enters the forbidden region, and since one of the peaks is obscured by the forbidden region, only two peaks are detected. Figure 2.12b shows the glowworms that lie within the decision-domain of glowworm 26 at iteration $t = 1$. Since the region of intersection S (refer to Fig. 2.12b) of its decision-domain and the forbidden region is devoid of any neighbors, glowworm 26 avoids entrance into the forbidden region. Table 2.3 shows the initial positions and luciferin levels of glowworm 26 and its neighbors. Since it obtains feasible directions of movement at every time step, it continues to move by avoiding the forbidden region and finally gets co-located at the global-maximum.

Fig. 2.11 Situation where inter-agent communication helps a glowworm to select a feasible direction toward the source



2.4.4 Effect of Step-Size on Convergence

In Fig. 2.12a, the peak-location $(-0.46, -0.63)$ lies outside the convex-hull C . Due to the constant step-size, all the glowworms climb the gradient-hill by performing the leapfrogging behavior as described in Sect. 2.1.3 and get co-located at the peak in their vicinity. Similar behavior of capturing a peak that lies outside the convex-hull of a group of glowworms can also be observed in the simulation result of Fig. 2.8d. In order to obtain the solution within the desired tolerance ϵ range, the step size s should be less than the tolerance value. However, this may lead to a slower convergence of the algorithm. This issue can be taken care of, by starting the algorithm with a coarse (large) step-size and progressively reducing the step-size, to a value lesser than ϵ , toward the convergence phase. For instance, when the step size $s(t)$ was updated as $s(t) = q^t s(0)$, with $q = 0.96$ and $s(0) = 0.2$, convergence was achieved within 50 iterations ($s(50) = 0.026$) as opposed to 200 iterations in the constant step-size case. However, this experimental result serves only as an illustrative example and various models for the adaptive step-size have to be explored in order to make it work on a wide variety of problems.

The step size is also a function of the size of the search space. For relatively large search spaces, we need to start with a larger step size. However, there is a chance of missing closely spaced peaks depending on the number of peaks in a given area. In these cases, an additional search procedure in each local region of a peak captured with a smaller step size could reveal the presence of more peaks.

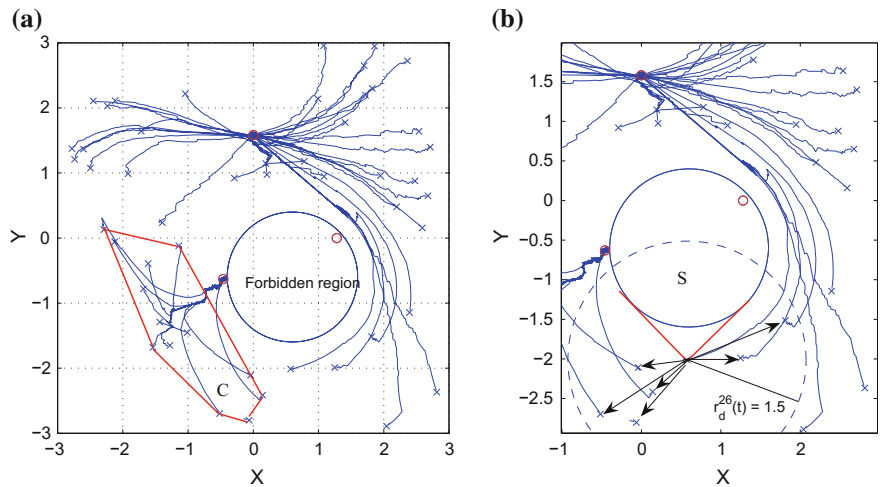


Fig. 2.12 **a** Response of the glowworm algorithm to the presence of a circular forbidden region of radius 1 unit centered at (0.6, −0.6). Emergence of the solution after 500 iterations; $r_d^i = 1.5$ **b** glowworm 26 has six feasible directions to move at Iteration 1

Table 2.3 Initial positions and luciferin levels of glowworm 26 and its neighbors in Fig. 2.12

Glowworm	$X_i(1)$	$\ell_i(1)$
6	(−0.5171, −2.6962)	2.7226
13	(−0.0405, −2.1075)	0.6371
21	(1.2484, −1.9892)	2.2977
26	(0.5735, −2.0126)	0.6069
32	(0.1413, −2.4149)	1.7583
36	(1.813, −1.5145)	2.9805
42	(−0.0651, −2.8011)	2.6762

2.5 Comparison of GSO with ACO and PSO

The GSO technique is a population-based algorithm and falls under the category of swarm intelligence methods. The algorithm shares some common features with the ant-colony optimization (ACO) and particle swarm optimization (PSO) algorithms but is different in many aspects that help in achieving simultaneous detection of multiple local optima of multimodal functions. This is a problem not directly addressed by ACO or PSO techniques. Generally, ACO and PSO techniques are used for locating global optimum. However, our objective is to locate as many of the peaks as possible. This requirement is the main motivation for formulating the GSO technique. Since the primary goal of GSO (that of capturing multiple peaks simultaneously) is different from that of ACO and PSO, comparisons with these techniques should be

based on variants of ACO and PSO that are adapted to capture multiple peaks. In the literature, PSO has been adapted for this purpose for the case of multiple peaks with equal function values [168]. PSO variants such as Niche-PSO [20] and species-based PSO [117] have been designed for capturing multiple local optima of multimodal functions. In this section, the comparisons are restricted mainly with respect to the underlying principles, algorithmic aspects, and applications. Experimental comparisons with Niche-PSO are given in Chap. 4.

2.5.1 ACO and GSO

Generally, ACO techniques are used and found to be effective in a discrete setting where gradient-based algorithms do not work too well. In this book, GSO is applied to the continuous domain because gradient-based algorithms do not produce satisfactory results when multiple peaks are sought.

Traveling salesman problem (TSP) is a typical example of an ACO application where pheromone update on city routes closely mimics the trail-laying phenomenon found in foraging of ants. In general, most conventional ant-algorithms involve adding a pheromone value on specific routes where the agents visit. The pheromone level on these visited routes decay with time in order to emulate the evaporation behavior of actual pheromones. Even though the luciferin update mechanism draws inspiration from these stigmergic principles followed by ants, there is a significant modification in its implementation and how the luciferin information is used by the agents in GSO. The difference mainly arises from the fact that the luminescence does not stay at places visited by glowworms (unlike pheromones that remain associated with routes visited by ants), but moves along with them. Therefore, the luciferin level of a glowworm indicates the net improvement made by it while traversing a path that emerges from an initial location to its current location. A glowworm whose location makes relatively more net improvement in the objective function during its movements has a higher luciferin level and attracts more neighbors toward it than others, thus enabling agents to move toward favorable places of the environment. Clearly, the number of luminescence sources is equal to the number of glowworms. While the pheromonal decay in ACO algorithms serves to avoid premature convergence to, and eventual removal of, suboptimal paths/solutions, the luciferin decay component in GSO controls the luciferin values of the glowworms and ensures that the luciferin values are always bounded. Note that, when the solution is reached, the number of luminescence locations will be approximately equal to the number of maxima of the objective function as most of the agents settle at one of the multiple peaks.

Now, GSO is compared to a variant of ACO, which was introduced by Bilchev and Parmee [15] in order to address continuous optimization problems. In this ACO approach, a finite set of regions are randomly placed in the search space. Each path between the nest and a region i is associated with a virtual pheromone $\tau_i(t)$ at each iteration t . Initially, $\tau_i(t = 0) = \tau_0$ for all agents. The probability that an agent selects region i is given by:

$$p_i(t) = \frac{\tau_i^\alpha(t)\eta_i^\beta(t)}{\sum_{j=1}^N \tau_j^\alpha(t)\eta_j^\beta(t)} \quad (2.25)$$

where, $\eta_i(t)$ reflects the local-desirability of a portion of the solution, α and β represent relative weights, and N is the number of regions. The agent then moves to the selected region's center, measures the value of the objective function at that point, moves a short distance in a random direction, shifts the region's center to the new point if it finds an improvement in the solution, and then comes back to the nest. The pheromone update associated with the region is given by

$$\tau_i(t+1) = \begin{cases} (1-\rho)\tau_i(t) + \gamma\Delta J, & \text{if } \Delta J > 0 \\ (1-\rho)\tau_i(t), & \text{Otherwise} \end{cases} \quad (2.26)$$

where, ΔJ is the improvement made in the solution, ρ is the pheromone evaporation constant, and γ is a proportionality constant.

This process is repeated with a new probability distribution according to (2.25). With the increase in number of iterations, the pheromone concentration associated with *inferior* regions decay (and may disappear eventually) and *good* regions get reinforced with time, finally converging to the solution.

The virtual pheromones associated with the various regions in the above variant of ACO technique can be likened to the luciferin carried by the glowworms in the GSO technique. However, the crucial difference lies in the manner in which the stigmergic communication is used by the agents to make decisions. In this ACO variant, each agent at the nest selects a region based on a probability distribution (2.25) which is a function of the pheromone levels associated with all the N regions. In contrast, each glowworm in GSO broadcasts its own luciferin value to other agents and uses the luciferin information available only in its neighborhood to probabilistically select a neighbor with higher luciferin value. While the selected region's center is shifted, along a random direction, to a new point in this ACO variant, each glowworm deterministically moves one step toward the selected neighbor. The main differences between GSO and ACO are summarized in Table 2.4.

2.5.2 PSO and GSO

Particle swarm optimization (PSO) is a population-based stochastic search algorithm [29, 98]. In PSO, a population of solutions $\{X_i : X_i \in R^m, i = 1, \dots, N\}$ is evolved, which is modeled by a swarm of particles that start from random positions in the objective function space and move through it searching for optima; the velocity V_i of each particle i is dynamically adjusted according to the *best so far* positions visited

Table 2.4 Comparison of ACO and GSO

	Standard ACO	GSO
1	Effective in <i>discrete</i> setting [48]	Applied to <i>continuous</i> domain
2	Global optimum	Multiple optima of <i>equal</i> or <i>unequal</i> values
	Special variant of ACO [15]	
1	Cannot be applied when ants (agents) have limited sensing range	Useful for applications where robots have limited sensor range
2	<i>Global</i> information used	<i>Local</i> information used
3	Pheromones associated with <i>paths</i> from nest to regions	Luciferin carried by and associated with <i>glowworms</i>
4	Pheromone information used to select <i>regions</i>	Luciferin information used to select <i>neighbors</i>
5	Shifting of selected region's center in a <i>random</i> direction	<i>Deterministic</i> movements toward selected neighbor

by itself (P_i) and the whole swarm (P_g). The position and velocity updates of the i^{th} particle are given by:

$$V_i(t+1) = V_i(t) + c_1 r_1 (P_i(t) - X_i(t)) + c_2 r_2 (P_g(t) - X_i(t)) \quad (2.27)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2.28)$$

where, c_1 and c_2 are positive constants, referred to as the *cognitive* and *social* parameters, respectively, r_1 and r_2 are random numbers that are uniformly distributed within the interval $[0, 1]$, $t = 1, 2, \dots$, indicates the iteration number. PSO uses a memory element in the velocity update mechanism of the particles. Differently, the notion of memory in GSO is incorporated into the incremental update of the luciferin values that reflect the cumulative goodness of the path followed by the glowworms.

GSO shares a feature with PSO: in both algorithms a group of particles/agents are initially deployed in the objective function space and each agent selects, and moves in, a direction based on respective position update formulae. While the directions of a particle movements in original PSO are adjusted according to its own and global best previous positions, movement directions are aligned along the line-of-sight between neighbors in GSO. In PSO, the net improvement in the objective function at the iteration t is stored in $P_g(t)$. However, in GSO, a glowworm with the highest luciferin value in a populated neighborhood indicates its potential proximity to an optimum location.

Figure 2.13a–c illustrate the basic concepts behind PSO and GSO and bring out the differences between them. Figure 2.13a shows the trajectories of six agents, their current positions, and the positions at which they encountered their personal best and the global best. Figure 2.13b shows the GSO decision for an agent which has four neighbors in its dynamic range of which two have higher luciferin value and thus only two probabilities are calculated. Figure 2.13c shows the PSO decision, which

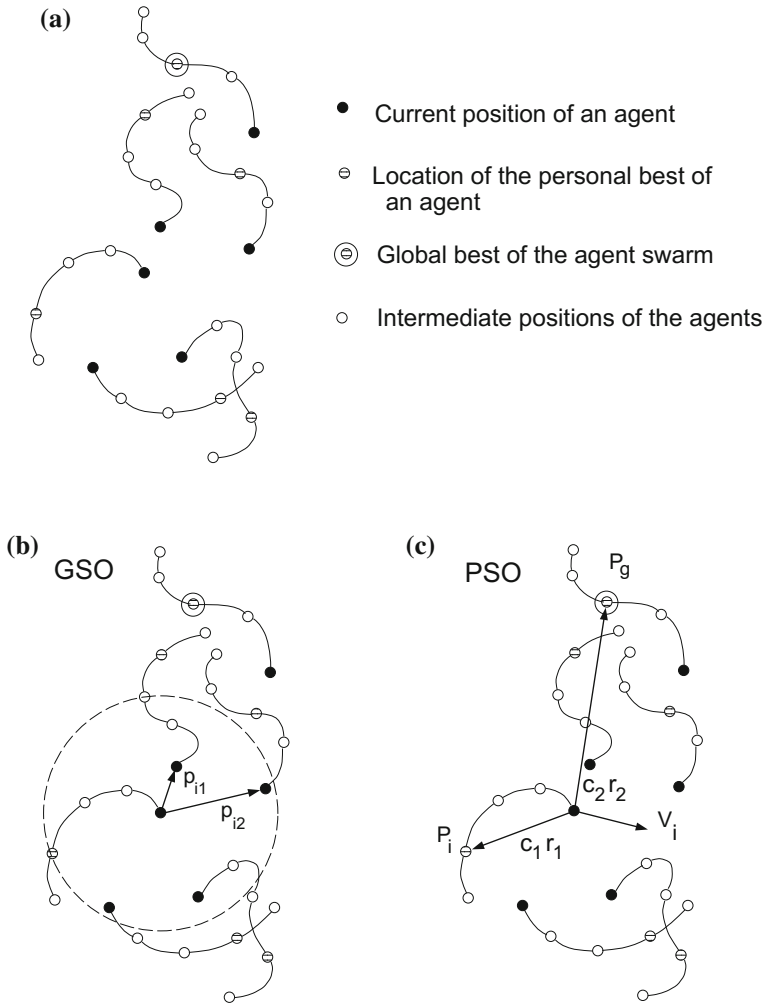


Fig. 2.13 **a** Trajectories of six agents, their current positions, and the positions at which they encountered their personal best and the global best. **b** GSO decision for an agent which has four neighbors in its dynamic range of which two have higher luciferin value and thus only two probabilities are calculated. **c** PSO decision which is a random combination between the current velocity of the agent, the vector to the personal best location, and the vector to the global best location [113]

is a random combination between the current velocity of the agent, the vector to the personal best location, and the vector to the global best location.

In PSO, the next velocity direction and magnitude is dependent on combination of the agent's own current velocity, and randomly weighted global best vector and personal best vector. While this is implementable in a purely computational platform, implementation of this algorithm in robotics platform or a platform containing realistic agents would demand large speed fluctuations, presence of memory of the

Table 2.5 PSO versus GSO

	PSO	GSO
1	Net improvement in the objective function at iteration t stored in the global best position $P_g(t)$	A glowworm with the highest luciferin value in the swarm indicates its potential proximity to an optimum
2	Direction of movement based on previous best positions	Agent movement along line-of-sight with a neighbor
3	Neighborhood range covers the entire search space	Maximum range hard limited by finite sensor range
4	Dynamic neighborhood based on k nearest neighbors (in a local variant of PSO)	Adaptive neighborhood based on varying range
5	Limited to numerical optimization models	Can be applied to multiple source localization in addition to numerical optimization

personal best position of each agent, and knowledge of the global best position which requires global communication with all other agents. In the local variant of PSO, P_g is replaced by the best previous position encountered by particles in a local neighborhood. In one of the local variants of PSO, the dynamic neighborhood is achieved by evaluating the first k nearest neighbors. However, such a neighborhood topology is also limited to computational models only and is not applicable in a realistic scenario where the neighborhood size is defined by the limited sensor range of the mobile agents.

In GSO, the next direction of movement of an agent is determined by the position of the higher luciferin carrying neighbors within a dynamic decision range and the weights are determined by the actual values of the luciferin level. Thus, in GSO the implementation is much simpler as the algorithm demands communication only with a limited number of neighbors and therefore does not require to retain personal best position information, nor does it require to collect data from all agents to determine the global best position.

Conceptually, the fact that GSO does not use the global best position or the personal best position and the fact that it uses information only from a dynamic neighbor set helps it to detect local maxima, whereas PSO gets easily attracted to the global maxima.

The above figures and explanation imply that GSO is completely different from PSO although they have some minor commonalities. The main differences between GSO and PSO are summarized in Table 2.5.

2.6 Summary

In this chapter, the development of glowworm swarm optimization (GSO) algorithm was presented. The underlying ideas behind the GSO technique, the notion of an adaptive local-decision domain, and the steps involved in the implementation of the basic GSO algorithm were described. Some theoretical results were presented that gave an insight into how the luciferin levels of glowworms vary as a function of time. Later, a simulation example was used to illustrate the basic capability of GSO for solving multimodal function optimization problems; several aspects of GSO like the effect of using an adaptive local-decision domain, local search due to leapfrogging behavior, and the effect of forbidden region on agent behavior were demonstrated. Finally, some comparisons of GSO with other bio-inspired optimization techniques were described. In the next chapter, the theoretical performance of a simplified GSO model and some illustrative simulations to support the theoretical findings are presented.

2.7 Thought and Computer Exercises

Exercise 2.1 The original GSO algorithm is described for solving maximization problems. Show that a minor modification to the definition of a neighbor is sufficient to extend GSO to minimization problems. Can you think of any other method to achieve the same purpose? (Hint: Consider changing the sign of the terms in the luciferin update rule given in (2.1)). Support your claims through numerical simulations (Use the MATLAB code provided in the Appendix). For this purpose, first run the ‘GSO.m’ file and verify that it captures the three maxima of the Peaks function (2.24), which is specified in the ‘UpdateLuciferin.m’ file. Now, update the code with each modification and confirm if the algorithm is able to capture the minima of this multimodal function. Comment on your observations.

Exercise 2.2 The parameter ρ (luciferin decay constant) in the luciferin update rule (2.1) simulates the decay of luciferin with time. What value of ρ results in a memoryless variant of GSO? Run the GSO code on a few multimodal test functions for different values of ρ while keeping other parameters constant. Analyze the impact of ρ on the emergence of solution in each case via different plots. Does the impact of ρ change if the parameter n (number of agents) is varied?

Exercise 2.3 In the movement update of GSO, each glowworm uses a probabilistic mechanism (2.2) to select a brighter neighbor and moves a small step closer to it (2.3). A new variant of GSO can be obtained by letting the glowworm to deterministically select the max-neighbor (refers to the glowworm with the maximum luciferin value in its neighborhood) instead of moving based on probability. Explain via different simulation scenarios how the emergence of solutions changes when this GSO variant is used.

Exercise 2.4 The parameter n_t sets the desired number of neighbors for each glowworm (2.4). Comment on the impact of n_t on the connectivity of the agent graph. For example, what happens when $n_t = 0$? What happens when $n_t = n - 1$? Use simulation examples to support your arguments.

Exercise 2.5 Consider the multimodal function given below (2.29) in a search space $[-3, 3] \times [-3, 3]$:

$$J(x, y) = 3 \left(e^{-b[(x+\frac{d}{2})^2+y^2]} - e^{-b[x^2+y^2]} + e^{-b[(x-\frac{d}{2})^2+y^2]} \right) \quad (2.29)$$

2.5.1 The above function has two peaks at $(-\frac{d}{2}, 0)$ and $(\frac{d}{2}, 0)$. The variable d controls the distance between the two peaks. The function profile for various values of d is shown in Fig. 2.14. Run the GSO code for $d = 1$ and $b = 10$. Use Table 2.1 to set the values of $\rho, \gamma, \beta, n_t, s$, and ℓ_0 . Set $n = 100$ and $r_s = 3$. Terminate the run after 1000 iterations (This large value is used only to ensure proper convergence. A better set of terminal conditions will be described in Chap. 4). Is the algorithm able to capture both the peaks? If yes, compute the solution error w.r.t. each peak (Hint: Solution error may be given by the distance between the true peak location and the average location of all the glowworms co-located at that peak. A more formal definition is deferred to Chap. 4).

2.5.2 Decrease d in steps of 0.1 and repeat the run for each case. Plot the solution error for each peak as a function of d . Comment on your observations. What is the minimum value of d for which the algorithm is able to capture both the peaks within a error tolerance of 0.05?

2.5.3 Set $d = 0.1$ and repeat the run. What do you observe? Explain why this happens (Hint: Plot the multimodal function for this value of d).

2.5.4 The parameter b controls how the slope of the function profile changes in the vicinity of each peak. Set $b = 50$ and plot the multimodal function again. Notice the presence of two peaks now. Verify if the algorithm is able to capture the two peaks in this new regime. Comment on your findings. If the algorithm fails, can you suggest any changes to the GSO parameters that enable the algorithm to distinguish between the peaks? (Hint: Keep decreasing n_t in steps of 1 and see what happens).

Exercise 2.6 Consider a modification of the Two-peaks function used in the previous exercise as shown below:

$$\begin{aligned} J(x, y) = & 6e^{-10[(x-0.5)^2+(y-0.5)^2]} - e^{-10[x^2+y^2]} \\ & + 4e^{-10[(x-0.5)^2+(y+0.5)^2]} + 3e^{-10[(x+0.5)^2+(y+0.5)^2]} \\ & + 2e^{-10[(x+0.5)^2+(y-0.5)^2]} \end{aligned} \quad (2.30)$$

By inspection of (2.30), it is clear that the above function has four peaks. Identify these peak locations. Where does the global peak occur? Now, run the GSO code

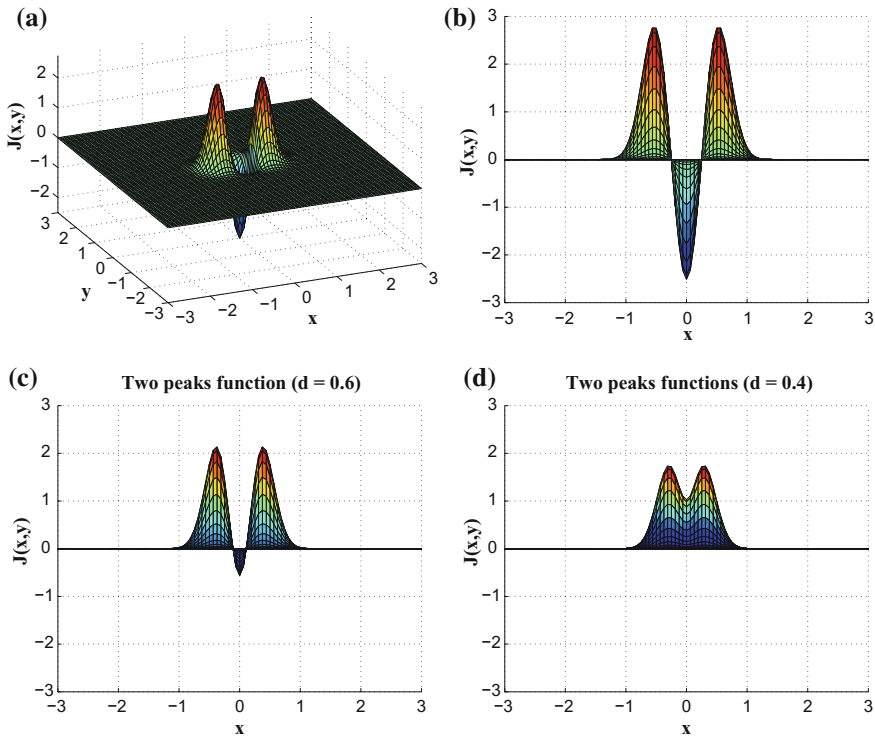


Fig. 2.14 The Two-peaks function for various values of d : **a** $d = 1$ (3D plot). **b** $d = 1$ (2D plot). **c** $d = 0.6$. **d** $d = 0.4$. The value of b is fixed at 10 in all three cases

using the nominal GSO parameters used in the previous exercise for the following two regimes:

1. Adaptive neighborhood case: Run the original GSO with $r_s = 3$ and record the number of peaks captured.
2. Constant neighborhood case: Modify (2.4) to $r_d^i(t+1) = r_s$ (This change can be made in 'Act.m', line 17) and run the modified GSO for fixed values of r_s ($=3, 2, 1$, and 0.5). Record the number of peaks captured in each case.

Use your observations to comment on the impact of adaptive neighborhood on the performance of GSO (Fig. 2.15).

Additional Notes

The basic GSO algorithm has been modified by several researchers subsequent to its first appearance. Some of these works have suggested modifications in the basic algorithm and some have combined it with other swarm intelligence algorithms to get improved performance. A good summary of these modifications and hybridizations has been described by Singh and Deep [197]. Some of these will be discussed in Chap. 8 with more details.

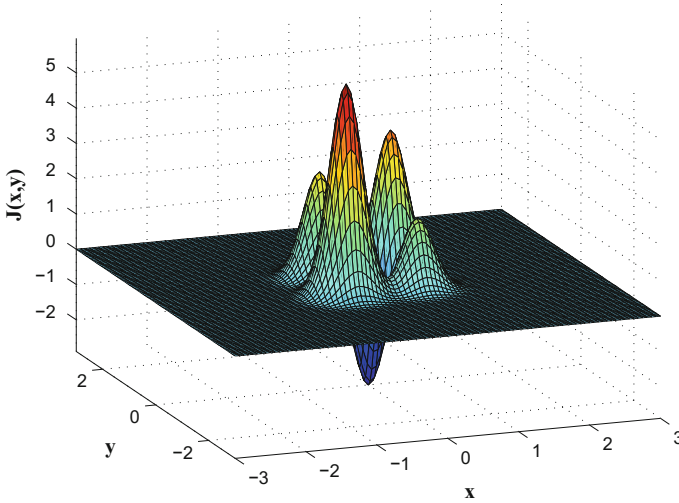


Fig. 2.15 The four-peaks function

A few years after the GSO algorithm was introduced, the firefly algorithm (FA) was proposed by Yang [222], which essentially follows a similar logic as GSO with some variations. The first important similarity is that the attractiveness of each firefly is proportional to its brightness. In particular, each firefly gets attracted and moves toward a relatively brighter firefly. The second similarity lies in the fact that the brightness of the glow of each firefly is determined by the landscape of the objective function. The FA algorithm has two variations. In GSO, the attractiveness of each glowworm is constant within a fixed sensing range r_s and zero beyond the range. However, the attractiveness of each agent in FA exponentially decays with distance. The second variation is the addition of a small randomization into the movement update of each firefly.

The key property (adaptive neighborhood) of GSO that enables an explicit splitting behavior of the swarm is not present in FA. The effect of the exponentially decaying property of agents' brightness in FA is that, although farther neighbors have less influence on the movement-decisions of each agent, it is still fully connected with all the remaining agents in the swarm, thereby leading to a centralized system. This makes FA more geared toward global optimization.

Work on GSO that appeared in the recent literature can be primarily classified into three categories. Some researchers proposed modifications of GSO [3, 4, 50, 75–77, 165, 171, 203, 219, 243, 244], most of which (with the exception of [3, 4]) were geared toward modifying GSO for global optimization problems. Some used basic GSO in different applications [27, 33, 90, 100, 135, 214], while others modified GSO and used them in some applications [3, 89, 116, 144, 147, 148, 213, 226, 235, 242].

Although GSO was designed for multimodal optimization, it was used for global optimization by Chetty and Adewumi [27]. The authors showed that its performance is comparable to other swarm intelligence algorithms that are specifically designed for global optimization. They used the effectiveness of employing these algorithms on an annual crop planning problem as a metric for comparison purposes.

Ouyang et al. [165] proposed BFGS-GSO, a hybrid algorithm of Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, which is one of the well-known classical gradient based algorithms, and GSO for global optimization problems. The authors showed that BFGS-GSO improved on the performance of basic GSO on a set of eight standard benchmark test functions.

Zhou et al. [243] proposed a GSO variant, in which the swarm was divided into two sub-populations that co-evolved toward global optima. Agents in one sub-population executed the basic GSO, while those in the other sub-population executed an algorithm based on Lévy flights. Lévy flights refer to a class of generalized random walks in which the step lengths during the walk are described by a ‘heavy-tailed’ probability distribution [12].

Du et al. [50] modified GSO for global optimization by omitting the adaptive neighborhood and probability based neighbor-selection phases and adding an element of elitism. The authors showed that the simplified-GSO performed better than the basic GSO and other swarm intelligence algorithms on a set of nine benchmark multimodal functions.

Zhouab et al. [244] incorporated principles of artificial fish swarm algorithm (AFSA) and differential evolution into GSO and applied it to constrained global optimization problems. A local search strategy based on simulated annealing was also applied in order to overcome premature convergence. The authors used tests on several benchmark functions to show that these modifications improved the convergence efficiency and computational precision of GSO.

He et al. [76] proposed a GSO variant by incorporating a two-layer hierarchical structure into the basic GSO algorithm. The bottom layer consisted of multiple subswarms of glowworms that separately searched different partitions of the search space. The optima captured in multiple partitions were used to initialize a new swarm of glowworms that formed the top layer of the hierarchy. The operators of selection and crossover were incorporated into the top-level swarm which led to enhancement in the diversity of the swarm. The authors reported simulation results on several benchmark functions to show that these modifications improved the speed and accuracy of GSO.

Aljarah and Ludwig [3, 5] parallelized GSO by using MapReduce, a popular programming model developed by Google for processing and generating large data sets with a parallel, distributed algorithm on a cluster [38]. The authors showed significant speedup and scalability, while maintaining the optimization quality on several large scale multimodal functions. The clustering-GSO was also parallelized by using MapReduce by Al-madi et al. [2] and similar improvements in speed and scalability were reported.

Huang and Zhou [84] introduced chaotic search strategies in GSO to obtain a more well-distributed initial solution. Zhou et al. [239] modified GSO by the cloud model

of optimization to improve its convergence property. Gu and Wen [69] attempted to improve the convergence property of GSO by using quantum models which increases the diversity of the swarms. Singh and Deep [196] proposed several variants of the original GSO on the basis of step size variations and tested them against standard benchmark problems. Similarly, Zhang et al. [230] presented an adaptive step size glowworm swarm optimization algorithm, which is claimed to improve convergence of the original GSO.

Basic GSO was used by Manimaran and Selladurai [135] to solve nonlinear fixed charge transportation problem in a single stage supply chain network. The objective function consists of a fixed cost that is incurred for every route and a variable cost that is proportional to the amount shipped. Difficulty in solving the problem arises due to nonlinearities in variable costs and discontinuities caused due to fixed costs. GSO was formulated to compute a least cost transportation plan that minimizes the total variable and fixed costs while satisfying the supply and demand requirements of each plant and customer. The authors showed that GSO performs better than spanning tree-based genetic algorithm in terms of total distribution cost.

Couceiro et al. [33] conducted swarm robotic experiments to benchmark five state-of-the-art algorithms for cooperative exploration tasks: (1) Robotic Darwinian Particle Swarm Optimization (RDPSO) [31, 32], (2) Extended Particle Swarm Optimization (EPSO) [178, 179], (3) Physically-embedded Particle Swarm Optimization (PPSO) [78], (4) GSO, and (5) Aggregations of Foraging Swarm (AFS) [60, 61]. All the five algorithms were described as belonging to a class of algorithms originally designed to solve optimization problems and later adapted to embrace the principles associated with real robotic tasks. Features like obstacle avoidance, initial deployment, communication mechanism, parametrization, handling of multiple/dynamic sources, computational complexity, memory complexity, and communication complexity were used to evaluate the theoretical advantages and disadvantages of the algorithms. The authors noted that only RDPSO and GSO were suited to handle multiple and dynamic sources. The authors initially compared the algorithms in a multi-robot simulation where agents collectively explored a large basement garage with a large density of obstacles. Metrics like exploration ratio and the associated area under the curve were used to evaluate each algorithm. The effectiveness of the top three performers found in simulations (RDPSO, GSO, and AFS) was further explored in real experiments using a swarm of fourteen e-puck robots [154]. The task consisted of collectively finding two victims emulated by e-pucks located at diagonally opposite corners of the workspace. The time taken to rescue each victim was used as a metric to evaluate each algorithm. The authors showed that RDPSO performed best and that GSO's performance closely followed RDPSO. The authors argued that this was a crucial result as GSO presents itself as a "low cost" alternative to the RDPSO in terms of computational and memory requirements. They also noted that although RDPSO performed better than GSO, a similar final outcome would be achieved by GSO if a larger mission time were available.

Senanayake [191] compared various swarm intelligence algorithms, including GSO, for the problem of search and tracking by a swarm of robots. This was a fairly exhaustive comparison.

Zhou et al. [241] used a modified GSO with a random operator to solve the problem of scheduling the dispatch of public transport vehicles.

Jiang and Tan [90] applied basic GSO to optimization of polarimetric multiple-input multiple-output (MIMO) radar systems. The problem involves a distributed array of transmitters and receivers used to detect a target. The goal is to select the parameters of the polarization waveforms of the transmitter array that maximize the probability of detection of the target. The authors formulated the optimization problem in the framework of GSO by using the formula for probability of detection in the luciferin update function and allowing the glowworms to search in the space of polarization parameters. The authors showed that the proposed GSO based algorithm outperformed other transmit waveform polarization schemes.

Liang et al. [119] applied GSO to solve a combined system identification and adaptive control problem for a mechanical servo system.

Xing [220] explored the possibility of using various computational intelligence methods, including GSO, to various robot related activities, such as location identification, manipulation, communication, vision, learning, and docking capabilities, in the context of assisted living for elderly people. The same author also tested computational intelligence algorithms (including GSO) for their suitability for data mining problems in the context of assisted living [221].

Kavipriya [96] applied GSO to solve code allocation problems in communication channels.

Kang et al. [94] applied GSO to solve a problem in X-ray based navigation. The bispectral feature points of the standard pulsar integrated pulse profile is extracted by the GSO algorithm and stored in the spacecraft's database. This information is now used to compute the X-ray pulsar time delay. The method shows a very good improvement in terms of computation time and suppression of Gaussian noise.

Pushpalatha and Ananthanarayana [180] proposed a GSO based document clustering algorithm that helps to cluster documents according to topics. The algorithm was tested through cluster based retrieval of multimedia documents.

Reddy and Rathnam [183] addressed the problem of optimizing power flow by minimizing the generation cost while keeping the power outputs of generators, bus voltages, bus shunt reactors/capacitors and transformer tap settings within acceptable limits. The problem is formulated as a multi-objective optimization problem keeping the minimization of emission for environmental reasons in mind. Both PSO and GSO is used to obtain useful results. Mageshvaran and Jayabarathi [129] used GSO to minimize the amount of load shedding in power systems in order to contain the deleterious effect of cascaded tripping and blackout. Wang et al. [214] used GSO for optimizing load allocation between hydropower units. They formulated the problem as a multi-objective optimization model in accordance with the characteristics and particularity of each station, with the minimum water rate of the station as the optimal objective.

Khan and Sahai [100] applied GSO for adaptive usability evaluation of B2C eCommerce web services.

Tang and Zhou [202] modified GSO by combining it with some aspects of PSO and applied it to path planning of uninhabited combat air vehicles. They showed the

effectiveness of GSO by comparing it with ten other population-based optimization methods.

Zhao et al. [236] incorporated the Dijkstra's shortest path algorithm and genetic operators from GA into GSO to solve shortest path problems. Don et al. [43] used an adaptive discrete GSO algorithm and compared its performance by extensive simulations on the Travelling Sales Problem (TSP) with ACO and PSO.

Aljarah and Ludwig [4] modified GSO for data clustering applications. Specifically, the fitness function used to evaluate the goodness of the glowworms was adjusted to locate multiple optimal centroids. The fitness function was set to maximize similarity of glowworms within a cluster (by minimizing intra-agent distance) and minimize similarity of glowworms lying in different clusters (by maximizing inter-agent distance). The authors used the entropy and purity metrics to evaluate algorithmic performance. On most of data sets (Iris, Ecoli, Glass, Balance, Seed, Mouse, and VaryDensity) selected from the UCI database repository,⁴ the authors showed that their clustering-GSO outperformed other well-known clustering algorithms like K-Means clustering [128], average linkage agglomerative Hierarchical Clustering (HC) [237], Furthest First (FF) [81], and Learning Vector Quantization (LVQ) [102].

In the work carried out by Marinaki and Marinakis [137], GSO has been used in conjunction with other meta-heuristic algorithms to solve a vehicle routing problem with random demands. This is a case where GSO has been used to solve a combinatorial optimization problem. An innovative method, to express the solution as a combination of two vectors, one in the continuous solution space and the other in the discrete solution space, is proposed. The former is updated using GSO while the latter uses a combinatorial neighbourhood topology technique.

Yepes et al. [225] addressed the problem of optimizing cost and CO₂ emissions while designing pre-cast, pre-stressed, concrete road bridges. GSO has been used in a hybrid scheme with simulated annealing (SA), where the local searches are carried out by SA and the global one by GSO. The problem is defined in a 40 dimensional decision space.

Cui et al. [35] proposed a modification to GSO in terms of making the weight on the location update rule vary, followed by a hybrid scheme that uses differential evolution in conjunction with GSO for solving the problem of time series prediction using single multiplicative neuron to which feedback and feedforward links are incorporated. The author claims that this modification improves the performance of GSO by avoiding locally optimal traps and overcomes the deficiency of not using memory of the search history.

Li and Huang [118] solved the problem of blind signal separation using a modified version of GSO based on step-size adjustment. Pubo and Yu [177] also dealt with blind source separation problem using GSO and its modification by baffle effect.

Jayakumar and Venkatesh [89] proposed two modifications of the original GSO to solve a multi-objective optimization problem related to environmental economic dispatch. In particular, the multiple objectives included minimization of fuel costs and

⁴<http://archive.ics.uci.edu/ml/index.html>.

minimization of emissions, respectively. The first modification consisted of using TOPSIS (Technique for Order Preference Similar to an Ideal Solution), a multi-criterion decision making method to compute the fitness of each glowworm. For this purpose, each glowworm is evaluated with respect to all the objective functions and ranked using the TOPSIS method, which works by assigning a higher rank to agents that simultaneously have shorter geometric distance from the positive ideal solution and farther geometric distance from the negative ideal solution. The second modification was a time varying step-size as opposed to a fixed step-size in the original GSO algorithm. The authors showed that the proposed GSO-TOPSIS approach produced comparable results over other state-of-the-art algorithms, and better results in some cases.

Meher et al. [144] modified GSO by incorporating stochastic updating of glowworm positions and applied it to an optimization problem in power systems. The objective of the economic load consignment problem was to obtain the most favorable active power outputs that minimize the generating cost while gratifying the running constraints. The authors evaluated the algorithm on a set of fourteen power generation models and showed it provided better results than those using differential evolution (DE) found in the literature [133]. Later, the same authors solved the dynamic load dispatch problem of thermal generating units using GSO [145] where the objective was to schedule optimal power generation of dedicated thermal units over a specific time band.

Unit commitment (UC) is a problem in power systems whose objective is to determine the on/off status of each power unit and the economic dispatch of power demand in a scheduling period so as to minimize the total system production cost subject to constraints of the units and the power system. Li et al. [116] developed a binary version of GSO and applied it to the UC problem. Each glowworm represented a matrix of on/off status of all the units at every hour, over the entire scheduling period. The Euclidean distance metric in the original GSO was replaced by the Hamming distance. The algorithm was shown to be very competitive in solving the UC problem in comparison to previously reported results by algorithms like quantum-inspired evolutionary algorithm, improved binary particle swarm optimization, and mixed integer programming.

Zhou et al. [238] addressed the problem of optimal sensor placement for structural health monitoring systems. The problem is posed as a multi-objective optimization of information entropy indices, which provide uncertainty metrics for the identified structural parameters. The basic GSO with some modifications using binary coding system is proposed. The application area is structural health monitoring of a long-span suspension bridge.

Wang et al. [213] modified GSO by incorporating a congestion factor and used it to optimize the parameters of an echo-state network (ESN)⁵ based soft-sensor model of flotation processes.

Yu and Yang [226] used GSO with an adaptive step-size for a scheduling problem. They considered the whole-set orders problem: different customers place orders,

⁵ESN is a recurrent neural network with sparsely connected neurons.

where each order consists of multiple workpieces with different processing times and a different overall completion deadline. The goal is to maximize the number of weighted wholeset-orders. The authors reported that GSO performed better than genetic algorithms on this problem.

Menon and Ghose [147] modified GSO to address the problem of localizing the sources of contaminants spread in an environment, and mapping the boundary of the affected region. The authors used two types of agents, called the source localization agents (or S-agents) and boundary mapping agents (or B-agents) for this purpose. They defined new behaviour patterns for the agents based on their terminal performance as well as interactions between them that help the swarm to split into subgroups easily and identify contaminant sources as well as spread along the boundary to map its full length. Later the authors extended this GSO model to boundary mapping of 3-dimensional regions [148].

Zhou et al. [242] proposed a K-means image clustering algorithm based on GSO and showed its effectiveness in performing image classification on several benchmark images. The authors discuss the drawbacks of classical K-means clustering (sensitivity to initial conditions and getting trapped in local optima) and how GSO can do well in global searching, by searching for optimal clusters in parallel, thereby avoiding the impact of initial conditions. The authors reported that GSO performed better than K-means and fuzzy C-means (FCM) on the chosen benchmark images.

Zhao et al. [235] used a modified version of GSO for optimizing parameters of a Kaplan turbine (a propeller-type turbine with adjustable blades) with the goal of decreasing hydraulic losses and maximizing operating efficiency. The authors used a combination of a Relevance Vector Machine model and GSO to approximate the relationship between the wicket gate opening and runner blade angle.

Raman and Subramani [181] considered the problem of prioritization and minimization of the number of test cases for software testing. The authors proposed a modified GSO to solve this problem by conceptualizing a definite updating search field in the movement rule of the original GSO. The objectives were to maximize the path coverage and fault coverage to obtain optimal prioritized test cases. The authors claimed that the resulting solution guaranteed an optimal ordering of test cases and compared the performance of their modified GSO with PSO and artificial bee colony optimization (BCO).

Mo et al. [152] presented an example from economics and finance where GSO is used to determine the parameters of the Black–Scholes option pricing model.



<http://www.springer.com/978-3-319-51594-6>

Glowworm Swarm Optimization
Theory, Algorithms, and Applications
Kaipa, K.N.; Ghose, D.
2017, XXVI, 248 p. 122 illus., Hardcover
ISBN: 978-3-319-51594-6