Ant Focused Crawling Algorithm*

Piotr Dziwiński and Danuta Rutkowska

Department of Computer Engineering
Czestochowa University of Technology, Poland
dziwinski@kik.pcz.czest.pl,
drutko@kik.pcz.czest.pl

Abstract. This paper presents a new algorithm for hypertext graph crawling. Using an ant as an agent in a hypertext graph significantly limits its amount of irrelevant hypertext documents which must be downloaded in order to download a given number of relevant documents. Moreover, during all time of the crawling, artificial ants do not need a queue to central control crawling process. The proposed algorithm, called the Focused Ant Crawling Algorithm, for hypertext graph crawling, is better than the Shark-Search crawling algorithm and the algorithm with best-first search strategy utilizing a queue for the central control of the crawling process.

1 Introduction

Enormous growth of the Internet and easy access to the network enable huge amount of users to access WWW resources through search engines. Changeability and large amount of WWW pages is a big challenge for modern crawlers and search engines. They should reflect WWW resources as accurately as possible and also hold information about the resources as fresh as possible. Complete crawling entire Web is impossible in reasonable time, no matter which technology is available at the site where the search engines operate. An ideal crawler should be able to recognize relevance and importance of Web pages. The crawlers can order new links extracted from downloaded WWW pages by use of different methods. Some of them are measurements of similarity between pages and a current query, amount of links to point out WWW pages or the most popular Page Rank.

Most crawler algorithms use a queue that globally control the process of crawling. The first crawler algorithm was the Simple-Crawler method mentioned in [1,2]. In the Simple-Crawler algorithm, a crawler extracts URL addresses from documents and includes them at the end of the queue without ordering. Another type of the crawlers is called selective crawlers [1]. The selective crawlers select a next download page with respect to some criterions (relevance or importance of the page). Relevance of the documents can be calculated using a

* This work was partly supported by the Foundation for Polish Science (Professorial Grant 2005-2008) and the Polish State Committee for Scientific Research (Grant N516 020 31/1977), Special Research Project 2006-2009, Polish-Singapore Research Project 2008-2010, Research Project 2008-2010.

L. Rutkowski et al. (Eds.): ICAISC 2008, LNAI 5097, pp. 1018–1028, 2008.

classifier. In order to classify WWW documents, different methods can be employed, such as: k-nearest neighbors algorithm, Naive Bayes, support vector machines, decision trees, neural networks, fuzzy rules or neuro-fuzzy systems; see e.g. [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

If a crawler inserts a new URL in the queue, in the order depending on the relevance, we obtain the best-first search strategy [13, 14, 15]. In many cases, crawling on entire web is not required. Instead, it can perform crawling only a relevant part of the web. In this way, the focused crawling algorithm [16, 17] has been obtained. Relevant areas of the web are small enough for the focused crawler to operate on such areas in finite time. Locality of the subject in the web are studied by Davison [18]. Rungsawang et al. [19] proposed the consecutive crawling to take advantage of experience from earlier crawling processes. They built a knowledge-base employed to produce better results for the next crawling. The first crawling algorithm inspired by the nature – behavior of some animals or insects – is the Fish-Search algorithm [20]. There is one of the first dynamic search heuristics based on intuition that relevant documents often have relevant neighbors [18]. The best version of the Fish-Search algorithm is the Shark-Search algorithm introduced in [21]. The Shark-Search algorithm bases on the same intuition but introduces a real measure of relevance of anchors extracted from new documents. The relevance of the anchors is based on relevance of a current document, content and context of the anchor. However in this case, the proposed Shark-Search algorithm exploits a queue for central control of the crawling process.

How we could dispose the queue for the central control of crawling is a question for which the answer we get from the nature – from real ants that solved this problem long time ago. There is a lot of algorithms developed based on behavior of real ants. Most of them are known as ant colony optimization (ACO). The ACO has been applied to the Traveling Salesman Problem (TSP) [22, 23, 24], graph coloring [25, 26], dynamic shortest path problems arising in telecommunication networks, dynamic cleaning problem [27]. Wagner I. A. et al. [28] adopt artificial ants to consider the problem of deciding whether graph $G(V, E)$, V– set of vertices, E – set of edges, is Hamiltonian. This problem is a special case of the TSP. Moving rules of the artificial ants were employed to control robots in multi agent systems to solve the distributed covering problem [28]. In [29], artificial ants are used to network covering in the Vertex Ant Walk (VAW) algorithm.

The goal of this article is to introduce the Ant Focused Crawling Algorithm which does not require the queue and decreases the number of downloaded irrelevant documents. Artificial ants leave pheromone trails on the ground in order to mark some favorable paths that should be followed by other members of the colony. The ants, using the pheromone trails, share information about the problem to be solved. In this way, all members of the colony have access to the information about relevance of the vertices that can be visited in next steps in the hypertext graph while progressing the crawling process. This crawling process performed by artificial ants can be running simultaneously by many artificial ants (crawling robots) without central control by the queue.

Section 1 provides background information on the selective crawling, focused crawling, ant colony optimization, and the proposed algorithm. Section 2 surveys the Ant Focused Crawling Algorithm. Section 3 highlights experimental results for the first crawling process compared with the Shark-Search algorithm and the best-first search strategy. Moreover, the crawling process performed by the proposed algorithm is analyzed with regard to the number of repetitions. Section 4 concludes the article.

2 Ant Focused Crawling Algorithm

Effective crawling the Internet by crawlers is a main issue concerning search engines. There are two types of crawling: passive crawling and active crawling. In the active crawling, crawlers are controlled by utilize a queue to central control of the crawling process. The passive crawling consists of crawling without any central control. In this article, new Ant Focused Crawling Algorithm which does not use the central control in the form of the queue, is proposed. This algorithm saves system and memory resources of the hardware.

The ant colony optimization (ACO) takes inspiration from the foraging behavior of some ant species. Ants moves from their nests to food and leave pheromone trails on the ground, in order to mark favorable path that should be followed by other members of the colony. In this way, individual ants discover the shortest path between the nest and the source of food using undirected communication in the form of the pheromone trail. In the similar way, we adopt the ant behavior with regard to the focused crawling hypertext graph $\mathbf{G}$. Hypertext graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ is a directed graph in which vertice $v \in V$ and edge $e \in E$ correspond to WWW documents and links in documents, respectively. Artificial ants move in the hypertext graph. The pheromone in vertices contain information about relevance, number of visits, time of visits. Artificial ants selects next vertice v according to the rule as follows [30]

$$v = \begin{cases} \arg\max_{\omega \in J(u)} \left\{ [\widetilde{\tau}(\omega)]^{\alpha} \cdot [\eta(\omega)]^{\beta} \right\} & \text{if} \quad q \leq q_0 \\ p_{u \to v} & \text{if} \quad q > q_0 \end{cases} \tag{1}$$

where

$$p_{u \to v} = \begin{cases} \frac{[\widetilde{\tau}(v)]^{\alpha} \cdot [\eta(v)]^{\beta}}{\sum\limits_{\omega \in J(u)} [\widetilde{\tau}(\omega)]^{\alpha} \cdot [\eta(\omega)]^{\beta}} & \text{if} \quad v \in J(u) \\ 0 & \text{if} \quad v \notin J(u) \end{cases} \tag{2}$$

$J(u)$ – set of vertices connected with vertice u by use of edges $e \in \mathbf{E}$ in the hypertext graph $\mathbf{G}$,
q_0 – parameter, $q_0 \in [0, 1]$,
q – random number belonging to $[0, 1]$,
$p_{u \to v}$ – probability of the movement from vertex u to vertex v,
α – importance of the pheromone trail,
β – importance of the heuristic information,
$\widetilde{\tau}(u)$ – value of the pheromone smell perceived by artificial ants.

Equations (1) and (2) are similar to the rule used in the ant colony system [22, 23, 31].

Artificial ants move in the hypertext graph $\mathbf{G}$ from vertice u to vertice v, where $u, v \in \mathbf{G}$, using the set of edges E, and leave pheromone trails $\tau_l(v)$, $\tau_r(v)$, $\tau_{\Delta t}(v)$.

Pheromone $\tau_l(v)$ guarantees that the searching process is similar to the Hamiltonian cycle in a graph [32]. Pheromone $\tau_r(v)$ directs the ants to move in the relevant area of the hypertext graph. Pheromone $\tau_{\Delta t}(v)$ causes refreshment of the visited vertices after specific time which depends on the frequency change of the documents in the hypertext graph or the WWW network.

It is proposed that values of pheromone smell perceived by artificial ants are calculated as follows

$$\widetilde{\tau}(v) = [\tau_r(v)]^{\alpha_r} \cdot \left[\frac{1}{1+\tau_l(v))}\right]^{\alpha_l} \cdot [\tau_{\Delta t}(v)]^{\alpha_{\Delta t}} \tag{3}$$

where:
$\alpha_r, \alpha_l, \alpha_{\Delta t}$ – parameters which weight the relative importance of the pheromones $\tau_r(v)$, $\tau_l(v)$, $\tau_{\Delta t}(v)$, respectively.

Pheromone $\tau_r(v)$ is refreshed by all ants in each step from vertice u to vertice v in the hypertext graph G, similarly as in the ant system presented in [23], according to the equation

$$\tau_r(u) \leftarrow (1-\varphi_r) \cdot \tau_r(u) + \varphi_r \cdot \Delta\tau_r^k(u) \tag{4}$$

where:
ϕ_r – evaporation rate of pheromone $\tau_r(v)$,
$\Delta\tau_r^k(u)$ – quality of the pheromone $\tau_r(v)$, dependent on the relevance of the vertices available from vertice v and the next ones – $ch[v]$ (children of v) through ant k.

Every ant has a small memory that contains a specified number of visited vertices. This memory is used for local control of the crawling process (but is functioning differently as the queue), and avoids cycles that occur frequently in the hypertext graph. Quality of the pheromone $\Delta\tau_r^k(u)$ is calculated based on memory of the ants. It is dependent on relevance at further vertices. This is done with a specified delay and is associated with reinforcement learning. In this way, each ant contains small path $\mathbf{Q}^k$ which includes relevance of the visited vertices. Each ant uses path $\mathbf{Q}^k$, and leaves pheromone $\Delta\tau_r^k(u)$ calculated by the equation

$$\Delta\tau_r^k(u) = \sum_{i=1}^{L} \delta^i q_i^k \tag{5}$$

where:
δ – coefficient reducing influence of the relevance of the following vertices; $\delta \in [0, 1]$,
q_i^k – relevance of vertice i remembered for ant k; $q_i^k \in \mathbf{Q}^k$,
L – length of path $\mathbf{Q}^k$.

Equation (5) is effective only in the first crawling process performed by the Ant Focused Crawling Algorithm. We develop another equation for successive crawling process, in the form:

$$\Delta\tau_r^k(u) = \frac{\sum_{i=1}^{L} \delta^i \cdot q_i^k}{\sum_{j=1}^{L} \delta^j} \tag{6}$$

Pheromone $\tau_l(v)$ is increased by all ants during visiting each vertice. This strategy warrants convergence of the algorithm to the path approximate to the Hamiltonian path, like in the VAW algorithm [29].

Pheromone $\tau_{\Delta t}(u)$ is calculated as follows

$$\tau_{\Delta t}(u) = \frac{t_a(u) - t(u)}{t_{od}(u)} \tag{7}$$

where:
$t_a(u)$ – current time in vertice u,
$t(u)$ – visit time in vertice u by an artificial ant,
$t_{od}(u)$ – reference time (referenced to speed of the change in vertice u).

During the crawling process performed by artificial ants, members of the colony select the next vertice according to the value of the pheromone in the vertice and heuristic information $\eta(u)$, see Equations (1),(2). The heuristic information in ant algorithms evaluates a local solution of the problem. In the case of the TSP, the heuristic information is calculated as follows:

$$\eta(u, v) = \frac{1}{d(u, v)} \tag{8}$$

where $d(u, v)$ – distance measure between cites.

In the hypertext graph, the heuristic information should be related to potential relevance $r_p(v)$ of the following vertices or relevance $R(v)$ of the document with the context. It is proposed that the heuristic information is calculated as follows:

$$\eta(v) = \begin{cases} R(v) & \text{if} \quad d(v) \neq \varnothing \\ r_p(v) & \text{if} \quad d(v) = \varnothing \end{cases} \tag{9}$$

where:
$r_p(v)$ – potential relevance of the next vertice [20]; see Equation (11),
$d(v)$ – document for vertice v.

The relevance of a downloaded document with a context of the hypertext graph depends on the relevance of the document for vertice v and the arithmetic average of the heuristic information about the descendant vertices $ch[v]$, and is calculated as follows:

$$R(v) = r(d(v)) + \frac{1}{|ch[v]|} \sum_{\omega \in ch[v]} \eta(\omega) \tag{10}$$

where:
$ch[v]$ – set of descendant vertices of vertice v (children of v),
$|ch[v]|$ – number of descendant vertices of vertice v.

Equation (10) is similar to that presented by Mark et al. [1,33]; this involves the context of the graph.

The potential relevance of vertice v depends on the inherited relevance of the document $r_d(v)$ for vertice v, relevance of the content of the anchor $r_a(u,v)$ and relevance of the neighborhood of the anchor $r_{ac}(u,v)$. Potential relevance $r_p(v)$ is calculated as follows [20]:

$$r_p(v) = \gamma \cdot r_d(v) + (1-\gamma) \cdot r_n(v) \tag{11}$$

where:
γ – coefficient of the inherited relevance of the document $r_d(v)$ for vertice v; see Equation (13),
$r_n(v)$ – anchor relevance, given by Equation (15).

Inherited relevance $r_d(v)$ for vertice v depends on relevance of vertice u, which contains document $d(u)$, $u \in pa[v]$, is calculated as follows:

$$r_d(v) = \begin{cases} r(d(u)) \cdot \delta & \text{if} \quad r(d(u)) > \epsilon;\ u \in pa[v] \\ r_d(u) \cdot \delta & \text{if} \quad r(d(u)) \leq \epsilon;\ u \in pa[v] \end{cases} \tag{12}$$

where:
ϵ – threshold relevance value, like that in the Shark-Search algorithm [21],
δ – coefficient of reduction of the relevance for the inherited relevance of parent documents $pa[u]$.

If we have more than one vertice $u \in pa[v]$, the inherited relevance is given by:

$$r_d(v) = \frac{1}{|pa[v]|} \sum_{u \in pa[v]} \begin{cases} r(d(u)) \cdot \delta & \text{if} \quad r(d(u)) > \epsilon \\ r_d(u) \cdot \delta & \text{if} \quad r(d(u)) \leq \epsilon \end{cases} \tag{13}$$

where $|pa[v]|$ – number of parent vertices of vertice v.

The relevance of the anchor in vertice v consists of relevance of the anchor text $r_a(u,v)$ of document in vertice u and relevance of the context of the anchor $r_{ac}(u,v)$ of document in vertice u, and is calculated similarly as in [20]:

$$r_n(v) = \beta \cdot r_a(u,v) + (1-\beta) \cdot r_{ac}(u,v) \tag{14}$$

where β – coefficient of the influence of the content relevance of the anchor $r_a(u,v)$ and the context of the anchor $r_{ac}(u,v)$.

If we have more than one parent vertice $u \in pa[v]$, the relevance of the anchor is given by:

$$r_n(v) = \beta \cdot \frac{1}{|pa[v]|} \sum_{u \in pa[v]} r_a(u,v) + (1-\beta) \cdot \frac{1}{|pa[v]|} \sum_{u \in pa[v]} r_{ac}(u,v) \tag{15}$$

The relevance of the anchor text $r_a(u,v)$ is defined as [20]:

$$r_a(u,v) = \mathrm{sim}(q, \mathrm{anchor}(u,v)) \tag{16}$$

The relevance of the anchor context is calculated by [20]:

$$r_{ac}(u,v) = \begin{cases} \text{sim}(q, \text{ConAn}(u,v)) & \text{if} \quad r_a(u,v) \leq 0 \\ 1 & \text{if} \quad r_a(u,v) > 0 \end{cases} \tag{17}$$

where:
anchor(u,v) – anchor text for edge $u \rightarrow v$,
$\text{ConAn}(u,v)$ – text in the specified neighborhood of the anchor, for the edge $u \rightarrow v$.

3 Experiments and Results

The experiments were performed for JAVA documentation [34], initially parsed, and indexed. The parsed documentation was saved as a compressed XML file, which after reading and decompressing becomes the indexed hypertext graph $\mathbf{G}$. All experiments were performed for the same simple query q and for the same starting address. The experiments for a specified parameter are repeated some number of times.

In the experiments, we obtained satisfied results which overcome comparable algorithms. Figure 1a shows the number of relevant documents as a function of the number of all downloaded documents during entire crawling process. Figure 1b illustrates total amount of relevant information as a function of the number of all downloaded documents. The total amount of relevant information is calculated as follows [20, 21]:

$$Sum\_Inf(\mathbf{D^r}, q) = \sum_{d(u) \in \mathbf{D^r}} d(d(u), q) \tag{18}$$

where $\mathbf{D}^r$ – set of downloaded relevant documents.

The proposed new Ant Focused Crawling Algorithm, for hypertext graph crawling, overcomes compared algorithm – achieves results 33% better those obtained from the Schark-Search algorithm. Moreover, it skip an irrelevant area of the graph $\mathbf{G}$ and is focused only on relevant areas. In addition better stabilization over period of the crawling process is observed. The proposed algorithm does not use the queue to central control of the crawling process. Effectiveness of the algorithm [1] is calculated as follows:

$$e = \frac{r_t}{t} \tag{19}$$

where:
t – number of downloaded documents in specific period of time;
r_t – number of relevant downloaded documents for which the relevance is better than ϵ, where ϵ – threshold of the relevance.

An ideal crawling algorithm should obtain effectiveness equal to 1. The proposed algorithm achieves effectiveness better than two compared algorithms. The efficiency of co-operation of the artificial ants in the process of crawling

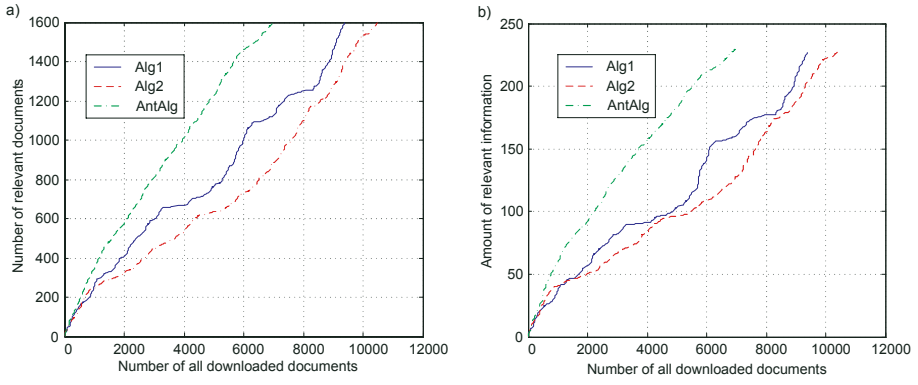

Fig. 1. Compared crawling algorithm with respect to (a) – number of relevant documents as a function of the number of all downloaded documents; (b) – amount of relevant information as a function of the number of all downloaded documents; Alg1 – focused crawling algorithm with the best-first search strategy; Alg2 – Shark-Search crawling algorithm; AntAlg – Ant Focused Crawling Algorithm

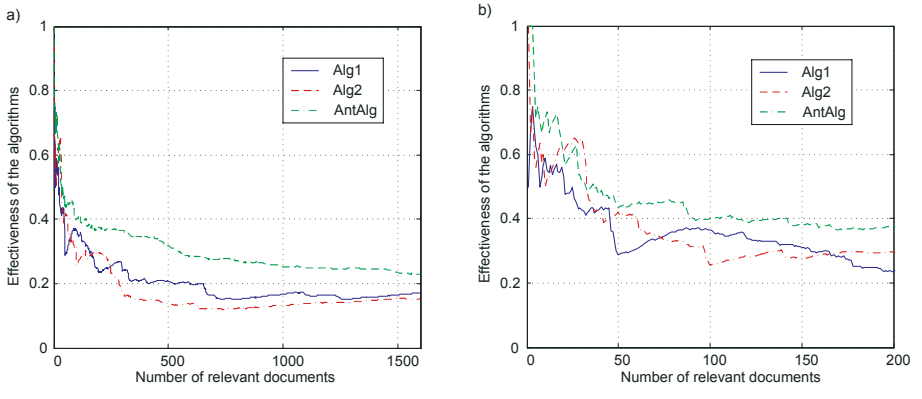

Fig. 2. Compared crawling algorithm with respect to effectiveness (a) – for number of relevant documents; (b) – for first 200 relevant documents; Alg1 – focused crawling algorithm with the best-first search strategy; Alg2 – Shark-Search crawling algorithm; AntAlg – Ant Focused Crawling Algorithm

is shown in Fig. 3. For more ants, the proposed algorithm obtains the same or better results. This feature of the proposed algorithm can be useful for crawling the web by many agents without using the queue for central control.

For successive crawling, an ant colony possesses experience about location of relevant information in the hypertext graph, saved in the form of pheromone. In successive crawling process, individual ants use information from earlier processes, and the Ant Focused Crawling Algorithm obtains better results, what is shown in Fig. 4. The proposed algorithm is useful in order to maintain freshness

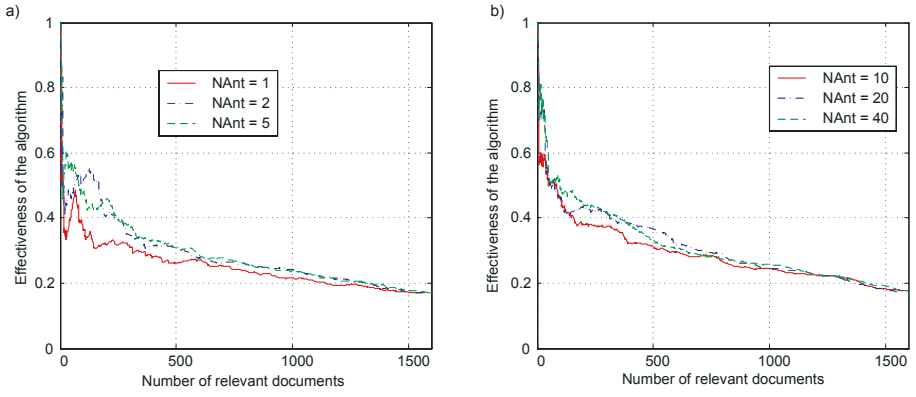

Fig. 3. Effectiveness of the Ant Focused Crawling Algorithm as a function of the number of relevant documents, depending on the number of ants; (a) – for the number of ants: 1,3,5; (b) – for the number of ants: 10,20,40

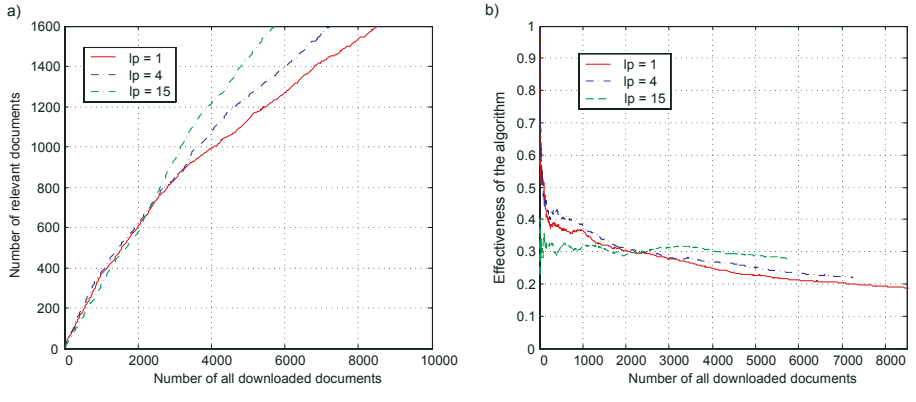

Fig. 4. Evaluation of the Ant Focused Crawling Algorithm depending on the number of repetitions: $lp = 1, 4, 15$ of the crawling process ; (a) – number of relevant documents as a function of the number of all downloaded documents ; (b) – effectiveness as a function of the number of all downloaded documents

of information in a local database about a part of the web. Moreover, the use of pheromone $\tau_{\Delta t}(v)$ makes possible focusing the ants in the changeable area of the hypertext graph.

4 Conclusion

The proposed new Ant Focused Crawling Algorithm, by applying indirected communication similar to that observed in some ant species, enables a crawling process performed by many agents (artificial ants) without any central control.

In this way, the proposed algorithm saves memory and hardware requirements. Moreover, it produces better results than two other compared algorithms.

References

1. Baldi, P., Frasconi, P., Smyth, P.: Modeling the Internet and the Web, Probabilistic Methods and Algorithms. Wiley, Chichester (2003)
2. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press, Cambridge (2001)
3. Cortez, C., Vapnik, V.N.: The hybrid application of an inductive learning method and a neural network for intelligent information retrieval. Machine Learning 20, 1–25 (1995)
4. Kłopotek, A.M.: Intelligent Search Engines. EXIT (in polish) (2001)
5. Duch, W., Adamczak, R., Diercksen, G.H.F.: Classification, association and pattern completion using neural similarity based methods. International Journal of Applied Mathematic and Computer Science 10(4), 101–120 (2000)
6. Bilski, J.: The UD RLS algorithm for training feedforward neural networks. International Journal of Applied Mathematic and Computer Science 15(1), 115–123 (2005)
7. Łęski, J., Henzel, N.: A neuro-fuzzy system based on logical interpretation of if-then rules. International Journal of Applied Mathematic and Computer Science 10(4), 703–722 (2000)
8. Łęski, J.: A fuzzy if-then rule-based nonlinear classifier. International Journal of Applied Mathematic and Computer Science 13(2), 215–223 (2003)
9. Piegat, A.: Fuzzy Modeling and Control. Physica-Verlag (2001)
10. Rutkowska, D., Nowicki, R.: Implication-based neuro-fuzzy architectures. International Journal of Applied Mathematic and Computer Science 10(4), 675–701 (2000)
11. Dziwiński, P., Rutkowska, D.: Algorithm for generating fuzzy rules for WWW document classification. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 1111–1119. Springer, Heidelberg (2006)
12. Dziwiński, P., Rutkowska, D.: Hybrid algorithm for constructing DR-FIS to classification www documents. In: Some Aspects of Computer Science, EXIT Academic Publishing House, Warsaw (2007)
13. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs (1995)
14. Cho, J., Garcia-Molina, H., Page, L.: Efficient crawling through URL ordering. Computer Networks and ISDN Systems 30, 161–172 (1998)
15. Baeza-Yates, R., Castillo, C., Marin, M., Rodriguez, A.: Crawling a country: Better strategies than breadth-first for web page ordering. In: International Word Wide Web Conference (2005)
16. Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific web resource discovery. Computer Networks (31), 1623–1640 (1999)
17. Diligenti, M., Coetzee, F.M., Lawrence, S., Giles, C.L., Gori, M.: Focused crawling using context graphs. In: 26th International Conference on Very Large Data Bases, pp. 527–534 (2000)
18. Davison, B.D.: Topical locality in the web. In: 23rd Ann. Int. ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 272–279 (2000)

19. Rungsawang, A., Angkawattanawit, N.: Learnable topic-specific web crawler. Computer Applications 28, 97–114 (2005)
20. Hersovici, M., Jacovi, M., Maarek, Y., Pelleg, D., Shtalhaim, M., Ur, S.: The shark-search algorithm – an application: tailored web site mapping. In: 7th International World-Wide-Web Conference on Computer Networks, pp. 317–326 (1998)
21. De Bra, P., Post, R.: Information retrieval in the world wide web: making client-based searching feasible. Computer Networks and ISDN Systems 27(2), 183–192 (1994)
22. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
23. Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization, artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine, 28–39 (November 2006)
24. Pintea, C.M., Pop, P.C., Dumitrescu, D.: An ant-based technique for the dynamic generalized traveling salesman problem. In: 7th WSEAS International Conference on Systems Theory and Scientific Computation, vol. 7 (2007)
25. Vesel, A., Zerovnik, J.: How good can ants color graphs? Journal of Computing and Information Technology - CIT 8, 131–136 (2000)
26. Dowsland, K.A., Thompson, J.M.: An improved ant colony optimisation heuristic for graph coloring, vol. 156, pp. 313–324. Elsevier Science Publishers B. V (2008)
27. Altshuler, Y., Bruckstein, A., Wagner, I.: Swarm robotics for a dynamic cleaning problem. In: Swarm Intelligence Symposium, SIS 2005, pp. 209–216 (2005)
28. Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Distributed covering by ant-robots using evaporating traces. IEEE Transactions on Robotics and Automation 15(5) (1999)
29. Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Efficiently searching a graph by a smell-oriented vertex process. Annals of Mathematics and Artificial Intelligence 24, 211–223 (1998)
30. Birattari, M., Pellegrini, P., Dorigo, M.: On the invariance of ant colony optimization. IEEE Transactions on Evolutionary Computation 11(6) (2007)
31. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B 26(1), 29–41 (1996)
32. Yanowski, V., Wagner, I.A., Lindenbaum, M., Bruckstein, A.: A distributed ant algorithm for efficiently patrolling a network. Algorithmica 37, 165–186 (2003)
33. Mark, E.: Searching for information in a hypertext medical handbook. Communications of the ACM (31), 880–886 (1988)
34. Documentation for the Java Platform, Standard Edition (2008), `http://java.sun.com/javase/reference/index.jsp`