Makespan Minimization on Parallel Processors: An Immune-Based Approach

A. M. Costa, P. A. Vargas, F. J. Von Zuben, P. M. França
School of Electrical and Computer Engineering State University of Campinas - UNICAMP
C.P. 6101 13083-270 Campinas - SP - Brazil
e-mail: alysson@densis.fee.unicamp.br, pvargas@dca.fee.unicamp.br,
vonzuben@dca.fee.unicamp.br, franca@densis.fee.unicamp.br

Abstract - This work deals with the problem of scheduling jobs to identical parallel processors with the goal of minimizing the completion time of the last processor to finish its execution (makespan). This problem is known to be NP-Hard. The algorithm proposed here is inspired by the immune systems of vertebrate animals. The advantage of combinatorial optimization algorithms based on artificial immune systems is the inherent ability to preserve a diverse set of near-optimal solutions along the search. The results produced by the method are compared with results of classical heuristics.

I. INTRODUCTION

The problem of minimizing the makespan while scheduling jobs to identical parallel processors is a classical combinatorial optimization problem. Following the 3-field classification scheme proposed by Graham *et al.* [1], this problem is denoted by $P||C_{max}$. The goal is to distribute the jobs amongst the processors, in a way that the makespan, i.e., the ending time of the last job in the most loaded processor, is as low as possible. In 1974, Bruno *et al.* [2] showed that this problem is NP-Hard and, therefore, there is no polynomial-time algorithm to obtain the optimal solution.

Several heuristic methods have been proposed for this problem, including the Longest Processing Time (LPT) [3] and the Multifit [4], both classified as constructive methods, i.e., heuristics that obtain solutions through additive steps, trying to improve the current solution gain at each step. Lee and Massey [5] hybridized these two heuristics and obtained results always better than the ones obtained by each heuristic individually.

Besides constructive heuristics, improvement heuristics are also abundant in the literature. These heuristics are designed for improving the results, via modifications in an initial solution [6]–[8]. Ho and Wong [6] used methods based on lexicographic search, able to find the optimal schedule in the two-processors case as an improvement algorithm for the general case. França *et al.* [8] proposed a 3-phase algorithm that uses a new job classification scheme which avoids, in the constructive and improvement phases, the use of sorting. Computational tests showed that 3-phase found better results than LPT and Multifit.

Classical heuristics strategies, like Tabu Search, have also been adopted to search for near optimal solutions to the problem. Piersma and Dijk [9], for instance, studied the problem focusing on the effect of the tabu neighborhood.

More recently, as for other combinatorial problems, there has been a tendency toward using evolutionary algorithms. Min and Cheng [10] proposed a genetic algorithm for this problem. Cheng and Gen [11] proposed a memetic algorithm to the extended problem of considering the jobs' execution times sequence-dependent.

In this work, we present a new search algorithm for the problem, based on the way the vertebrate immune system works. Artificial Immune Systems (AIS) [12], [13] are used in several domains of engineering, in particular in scheduling problems [14]–[16]. In this work, we extend the ideas proposed by De Castro and Von Zuben [17] to adapt the AIS mechanisms to our scheduling problems.

The results obtained are compared with the results of the following heuristics: LPT, Multifit (both followed by a local search heuristic) and Simulated Annealing.

In the next section, we formally define the problem. In section 3, we briefly present the heuristics to be compared. In section 4, the proposed method is detailed. The instance set and the computational results are presented in sections 5 and 6, respectively. Conclusions are outlined in section 7.

II. PROBLEM DEFINITION

The problem of scheduling jobs to identical parallel processors consists of executing n jobs in m identical parallel processors. All the jobs must be run once and the completion time of the last processor to finish execution

0-7803-7282-4/02/$10.00 ©2002 IEEE

must be the lowest possible.

In mathematical terms, given a set of n numbers, we must group them into m subsets in a way that each number is in one and only one subset, and the sum of the elements of the biggest subset is as low as possible.

A mathematical formulation is given below:

$$Min_X \quad M$$

$$s.t.$$
$$M - \sum_{i=1}^{n} p_i x_{ij} \geq 0 \quad 1 \leq j \leq m$$
$$\sum_{j=1}^{m} x_{ij} = 1 \quad 1 \leq i \leq n$$

It is clearly a mixed-integer optimization problem where:

M is the makespan,

X is a $n \times n$ matrix with binary elements,

$$x_{ij} = \begin{cases} 1, & \text{if job } i \text{ is placed on processor } j \\ 0, & \text{otherwise} \end{cases}$$

p_i is the i^{th} job size .

The first restriction assures that M is the makespan of the problem while the second assures that all jobs are executed only once.

III. THE HEURISTICS TO BE COMPARED

In this section, we briefly describe the heuristics used for comparisons with the method to be proposed. The constructive heuristics are presented first, followed by the local search adopted to improve their performance. Finally, the Simulated Annealing method is outlined.

A. Longest Processing Time (LPT)

The constructive heuristic LPT consists in sorting the jobs, by size, in a descending order and, then, scheduling the jobs, one by one, to the processor with the current least load.

B. Multifit

The Multifit heuristic, as the LPT, sort the jobs by size in a descending order. However, instead of scheduling each job to the least loaded processor, it tries to schedule the jobs to the same processor, until it is not possible to put jobs in the processor without exceeding its capacity C (optimization parameter). Only then, it goes to the next processor. The process continues until there is no more processors or jobs, i.e.:

1 - no more jobs to be placed: all the jobs have been placed in one of the processors. It means the makespan of the problem is less or equal to the actual capacity C. C is reduced and the process restarts, trying to find a smaller makespan.

2 - no more processors and still jobs to be placed: it was not possible to execute all jobs considering that the processors had a capacity C. C is increased and the process restarts.

After N iterations the process terminates and the best solution is chosen.

C. Local Search (LS)

A local search heuristic is used to improve the solutions obtained by the constructive heuristics.

The neighborhood generation mechanism is defined as all pairs of jobs, laying in processors with different charges, one of which must be the most loaded one.

The local search consists in visiting the pairs of jobs and accepting a change if it reduces the difference in charge of the two processors. Once a change has been made, the neighborhood is reconstructed and the search is restarted.

The search is interrupted when the whole neighborhood has been visited and no improvement was made (local optimum).

D. Simulated Annealing (SA)

Search strategies based on SA are widely used as combinatorial problem solvers [18]–[20]. It consists of a search strategy that, in a few words, allows the escape of local optima by accepting changes that degrade the solution if these changes satisfy one acceptation fuction.

The SA algorithm used in this work is presented bellow:

Simulated Annealing

Chose an initial Solution $S = S_0$;
$T = T_0$, $T_0 \geq 0$; $t = 0$;
Repeat:
  n=0;
  Repeat:
    Take the next element, S_v, in $N(S)$;
    $\delta = f(S_v) - f(S)$;
    if $\delta < 0$ then $S = S_v$
    else, do $S = S_v$ with a probability $e^{\frac{-\delta}{T}}$;
    $n = n + 1$;
  while $n < N$.
  $t = t + 1$;
  $T = T(t)$;
while stopping criteria is not satisfied.

The neighborhood $N(S)$ is the same one adopted in the local search and the initial solution used was given by the LPT rule.

IV. ARTIFICIAL IMMUNE SYSTEMS

The Vertebrate Immune System acts defending the organism against invaders (antigens) and has several desired features for optimization purposes, like robustness, flexibility, learning ability and memory. These characteristics are frequently useful in scheduling problems. Some examples of applications of AIS to scheduling can be found in [14]–[16].

Mori *et al.* [14] used an AIS for controlling a semiconductor production line. In their work, the control of the production line was done by a set of agents (named detector, mediator, inhibitor and restoration agents). Each agent interacted with the production line and with the other agents. The way this interaction occurred was based on the vertebrate immune system.

Hart *et al.* [15] worked with the job-shop problem, with the goal of minimizing the maximum tardiness. Each solution (a complete schedule) was an antibody. The proposed algorithm constructed these antibodies from a set of libraries. These libraries were previously evolved using a genetic algorithm. Once the libraries were defined, 1000 individuals were evaluated and 1000 clones of the best individual found were generated. The clones were mutated and the best clone found was selected as the solution of the problem.

Russ *et al.* [16] created an AIS model for task allocation in computer systems. The goal was to design a system capable of adapting to a changing environment. This was done in a way similar to the one proposed in [14]. Agents interacted with the system and amongst themselves as the B-cells and T-cells do in the natural immune system.

In this work, as in [15], we map the scheduling possibilities in a string of integers, and use these strings as antibodies of our AIS. The evolution of these antibodies in the AIS follows, basically, two principles: Clonal Selection (CS) and Affinity Maturation (AM). The CS principle dictates that the best defense cells (antibodies) should be selected to be cloned. The newly generated cells undergo hypermutation (a mutation with high probability) and receptor editing [21], [22], guiding to a process of Affinity Maturation. This principle is so called because the processes of mutation with high rates, together with selection, allow the immune system cells to improve their affinities with the recognized antigens. The number of clones is proportional to the antibody affinity (level of antigen matching), while the rate of mutation is inversely proportional to the affinity of the parent cell with the recognized antigen.

The CS and the AM principles are processes that occur simultaneously: during the CS process, each newly generated cell goes through a blind variation process, whereas during the AM process, the ones that best match the invaders (antigens) are selected.

Optimization operators may be modeled based on these principles. In this paper, we used an adaptation of one of the proposals of the literature, known as CLONALG [17]. In this algorithm, candidate solutions for a problem are coded. After that, selection and mutation operators are applied. The simplified algorithm is described below :

```
Algorithm for Clonal Selection and Affinity Maturation
Create a population of k antibodies (feasible solutions to
 the problem);
For each generation, do:
    For each antibody, do:
      decode the antibody;
      determine the antibody affinity;
    determine the number of clones of each antibody;
    determine the number of mutations;
    do cloning and mutation;
    For each clone, do:
      decode the clone;
      determine the clone affinity;
      if afin(clone) > afin(antibody) ⇒ antibody =
clone;
while stopping criterion = false.
```

For the $P||C_{max}$ problem, each feasible solution, i.e., a complete schedule, was coded in a string of fixed size n (n = number of processes). Each position on the string is associated with a process. The value of each position i indicates the machine where the process is allocated.

Each antibody (solution) of the population has an affinity. This affinity, as illustrated in the equation bellow, reflects the quality of the solution.

$$\text{Affinity}(k) = \frac{LB}{(1 + M(k) - LB)}$$

where

$M(k)$ is the makespan of the solution represented by the antibody k and LB is a lower bound of the problem.

In this paper, we have used the solution to the problem with preemption (i.e., allowing the split of jobs into two or more processors) as lower bound. This lower bound is calculated by the sum of all job processing times divided by the number of processors.

The denominator in the affinity expression states that for solutions where M(k) is closer to LB - i.e., the solution

improves - the affinity is higher. The denominator is just a multiplication factor to make the affinity expression more independent on the values of the jobs being treated.

Good mutation operators must allow the algorithm to implement a fine compromise between exploration and exploitation. In this work, we have used 5 types of mutation. The algorithm randomly chooses one of them any time a mutation is required in the evolutionary process.

Mutation 1 - choose one process randomly and associates it with a randomly chosen machine.

Mutation 2 - switch two randomly chosen processes of two randomly chosen machines.

Mutation 3 - switch a process from the most loaded machine with a process from the least loaded machine.

Mutation 4 - switch a process from the most loaded machine with a process from a randomly chosen machine.

Mutation 5 - (for instances with two processors) switch the machine of each job in one part of the antibody k (jobs belonging to machine 0 now belong to machine 1 and vice versa).

The initial population was composed of one previous evolved member (the outcome of the local search applied to the LPT) and other 29 members randomly generated.

The number of mutations for the antibody k was evaluated submitting the difference between the antibody k affinity and the best antibody of the population to a step function, as depicted in Table I.

Difference between the affinity of antibody k and the affinity of the current best antibody.	Numb. of mutations per antibody
≥ 0.003	9
0.002 - 0.003	8
0.001 - 0.002	7
0.0005 - 0.001	6
0.0004 - 0.0005	5
0.0003 - 0.0004	4
0.00001 - 0.0003	3
≤ 0.00001	2

TABLE I
NUMBER OF MUTATIONS PER ANTIBODY

The number of clones was given by the following equation:

$$NC(k) = 5 \cdot (N_{max} - NM(k)) + 1$$

Where $NC(k)$ is the number of clones of antibody k, N_{max} is the maximum number of mutations that can be applied to a cell (9, in this paper), and $NM(k)$ is the number of mutations of each cell generated from antibody k.

This expression was chosen in order to give the less evolved individuals, i.e., with higher number of mutations (see Table I), a few clones and vice versa. The numbers were selected in a way that the individuals with the maximum mutation rate had just one clone and individuals with the minimum mutation rate had 36 clones. Intermediary individuals had a number of clones between these two values.

The use of the antibodies affinity to estimate the number of newly generated cells and the mutation rate to be adopted for each cell is an inherent aspect of an immune-based approach.

The stopping criterion was a given number of generations with no improvement on the best solution. Another criterion used was the end of the available time for each instance, as explained in section VI.

V. INSTANCE SET GENERATION

For the computational experiments, two groups of instances were used.

In the first group, for each combination of $i \epsilon I$, $j \epsilon J$, $k \epsilon K$, with $i < j$, we generated 10 instances as described below:

- the instance has i processors;
- the instance has j jobs;
- the processing time of the jobs obeys an uniform distribution in the interval $[1, k]$.

In this paper, we worked with the sets $I = \{5, 10, 25\}$, $J = \{10, 50, 100, 500, 1000\}$ and $K = \{100, 1000, 10000\}$. We generated 10 instances for each possible combination, yielding a total of 390 instances. Optimal solutions for these instances were obtained by solving a sequence of bin packing problems within a bisection search scheme (see França *et al.* [8] for details). We also worked with another five instances with a stepwise distribution, proposed by Graham [3]. This distribution generates worst-case instances for the LPT algorithm.

The second group of instances has been borrowed from a similar problem, the number partitioning problem, in which there are only two processors. The set of instances was especially generated in a way that the best local minima - and consequently the optima - are located in very deep valleys.

These instances are expected to be very difficult to solve for many algorithms, especially because of the values of the jobs' processing times (in the order of 10 digits long) [23]. We used 25 instances, all with two processors and 15, 35, 55, 75 or 95 jobs.

VI. RESULTS

The maximum computational time for the optimization of each instance was given by $13 \cdot log(13) \cdot log(n^2)$, where n is the number of jobs of the instance. This time only limits the execution of the Artificial Immune Systems. All the other heuristics had execution times lower than the maximun time given.

The algorithms were coded in JAVA and executed in a Sun workstation Ultra 1.

Table II shows the results (% of optimal solutions found) for the first group of instances. The number of optimal solutions found by each method appears in parenthesis. The table also shows the total time used for the optimization of all instances in this group.

Method	% optima	Total time (s)
LPT	22 % (88)	4
Multifit	19 % (77)	43
LPT + LS	63 % (247)	18
Multifit + LS	35 % (136)	80
Sim. Annealing	87 % (345)	1986
AIS	88 % (346)	10402

TABLE II

RESULTS FOR THE FIRST GROUP OF 395 INSTANCES.

Amongst the constructive heuristics, the LPT has shown to be better than the Multifit. The local search (LPT + LS) improved the LPT result from 22% to 63% while for the Multifit the improvement (Multifit + LS) raised from 19% to 35%.

The table also shows that the proposed heuristic, AIS, for this group of instances, obtained results that are similar to the results of the Simulated Annealing, but with a much larger computational time.

Table III presents the results obtained by the best two heuristics[1] when applied to the second group of instances. In this table, we show the final load difference between the two processors. The table shows the statistical analysis (mean and standard deviation) for the SA and the AIS, after five runs[2].

In the table, the bold values indicate the best values found. The last column shows the optimal solutions found by Korf's exact algorithm [24], used here as a benchmark, given that it is very time-consuming and not comparable to heuristics.

The first conclusion that can be taken is that this group of instances is much harder than the first one. For this group of instances, the proposed heuristic was much more

[1]The other heuristics were not able to find any reasonable result.

[2]As a matter of fact, the SA did not use all the time allowed, converging to a solution before the end of the simulation time. In order to make a fairer comparison, the SA algorithm was allowed to run again and again until the time was over.

Inst.	Simulated Annealing			Art. Immune Systema			Opt.
	Mean	Std	Best	Mean	Std	Best	
15-1	666.58	579.91	**12.57**	12.57	0	**12.57**	12.57
15-2	488.84	316.98	**31.43**	31.43	0	**31.43**	31.43
15-3	407.09	529.44	**0.50**	0.50	0	**0.50**	0.50
15-4	622.99	716.04	**7.58**	7.58	0	**7.58**	7.58
15-5	490.80	418.93	**0.36**	0.36	0	**0.36**	0.36
35-1	168.31	184.40	0.10	0.10	0.09	**0.00800**	0.00002
35-2	115.77	99.36	1.27402	0.11	0.09	**0.00692**	0.00002
35-3	179.41	153.90	2.84846	0.10	0.08	**0.00388**	0.00008
35-4	109.36	113.86	0.15620	0.06	0.04	**0.02652**	0.00002
35-5	14.38	19.89	0.28209	0.09	0.11	**0.00281**	0.00005
55-1	53.23	83.95	0.09288	0.04	0.05	**0.00338**	0
55-2	44.55	46.34	0.77062	0.04	0.04	**0.00016**	0
55-3	53.00	50.02	0.04610	0.04	0.02	**0.01912**	0
55-4	46.61	51.38	0.43982	0.03	0.02	**0.00562**	0
55-5	43.72	38.31	0.38583	0.03	0.01	**0.02089**	0.00001
75-1	35.07	33.40	0.08234	0.07	0.09	**0.00168**	0
75-2	50.69	58.52	0.05917	0.04	0.02	**0.01355**	0.00001
75-3	31.56	31.26	0.08501	0.03	0.02	**0.01225**	0.00001
75-4	28.85	30.13	0.12869	0.02	0.01	**0.00147**	0.00001
75-5	36.55	39.91	0.36649	0.05	0.04	**0.00125**	0.00001
95-1	23.29	23.12	0.02212	0.03	0.02	**0.00706**	0
95-2	32.87	26.22	0.02031	0.03	0.03	**0.00123**	0.00001
95-3	29.92	30.21	0.04804	0.02	0.02	**0.00216**	0
95-4	19.74	22.24	0.06084	0.02	0.01	**0.00088**	0
95-5	23.68	33.29	0.01141	0.02	0.02	**0.00245**	0.00001

TABLE III

RESULTS FOR THE SECOND GROUP OF 25 INSTANCES, PRESENTED IN TERMS OF THE LOAD DIFFERENCE BETWEEN THE TWO PROCESSORS (ALL VALUES HAVE BEEN DIVIDED BY 10^5).

efficient than the Simulated Annealing. This can be seen not only by the best results obtained (see columns *Best*) but also by the robustness of the method, showed by the columns *Mean* and *Std*. The explanation for this different behavior lies on one inherent characteristic of combinatorial optimization algorithms based on AIS: the maintenance of the diversity of the population.

For the first group, the diversity was not essential, once there is a large number of global optima for each instance, scattered across the search space. Every time a job, belonging to a processor, is equal to the sum of some jobs belonging to another processor, we can exchange the jobs obtaining equivalent solutions. The number of these equivalent solutions is very high for the first group of instances, since we have an uniform distribution. However, for the second group, one can hardly find two equivalent solutions in different points of the search space. Diversity is, therefore, fundamental for an effective exploration of the search space. Additionally, it is well known the difficulty of SA to solve instances with local optima located in deep valleys, or said in other words, with rugged landscape.

VII. CONCLUSION

In this work, one algorithm based on the way the vertebrate immune system works was proposed to the problem of makespan minimization, when scheduling jobs to identical parallel processors. The results produced by the proposed algorithm, when compared with alterna-

tive strategies, indicate that immune-based algorithms applied to makespan minimization are effective in dealing with instances characterized by the presence of jobs with long processing times and a small number of machines. This kind of instances present a small set of good solutions through the search space, leading to a poor performance when single-solution strategies are considered.

The proposed strategy is based on a population of candidate solutions at each iteration, and the maintenance of diversity is considered the distinctive aspect to explain the better performance.

ACKNOWLEDGEMENTS

Fernando J. Von Zuben (proc. number 300910/96-7) and Paulo M. França acknowledge CNPq, and Alysson M. Costa and Patricia A. Vargas acknowledge CAPES for their financial support.

References

[1] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey", *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[2] J. Bruno, E.G. Coffman, and R. Sethi, "Scheduling independent tasks to reduce mean finishing time", *Communications of the ACM*, vol. 17(7), pp. 382–387, 1974.

[3] R. L. Graham, "Bounds on multiprocessing timing anomalies", *SIAM J.Applied Mathematics*, vol. 17, pp. 416–429, 1969.

[4] E. G. Coffman, M. R. Garey, and D.S. Johnson, "An application of bin-packing to multiprocessor scheduling", *SIAM J. Comput.*, vol. 7, pp. 1–17, 1978.

[5] C. Lee and J. D. Massey, "Multiprocessor scheduling combining LPT and multifit", *Discrete Applied Mathematics*, vol. 20, pp. 233–242, 1988.

[6] J.C.Ho and J.S.Wong, "Makespan minimization for m parallel identical processors", *Naval Research Logistics*, vol. 42, pp. 935–948, 1995.

[7] J.H. Blackstone Jr., "An improved heuristic for minimizing makespan among m identical parallel processors", *Comp. and Indust. Engn*, vol. 5, pp. 279–287, 1996.

[8] P.M. França, M. Gendreau, G. Laporte, and F.M. Muller, "A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective", *Computers & Operations Research*, vol. 21(2), pp. 205–210, 1994.

[9] N. Piersma and W. V. Dijk, "A local search heuristics for unrelated parallel machine scheduling with efficient neighborhood search", *Mathematic Comput. Modelling*, vol. 24, pp. 11–19, 1996.

[10] L. Min and W. Cheng, "A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines", *Artificial Intelligence in Engeneering*, vol. 13, pp. 399–403, 1999.

[11] R. Cheng and M. Gen, "Parallel machine scheduling problems using memetic algorithms", *Comp. and Indust. Engn*, vol. 33, pp. 761–764, 1997.

[12] D. Dasgupta (ed), *Artificial Immune Systems and their Applications*, Springer Verlag, 1998.

[13] S. A. Hofmeyer and S. Forrest, "Architecture for an artificial immune system", *Evolutionary Computation*, vol. 7(1), pp. 45–68, 2000.

[14] M. Mori, M. Tsukiyama, and T. Fukuda, "Artificial immunity based management system for a semiconductor production line", *Proc. of the IEEE Systems, Man and Cybernetics Conference*, pp. 851–855, 1997.

[15] E. Hart, P. Ross, and J. Nelson, "Producing robust schedules via an artificial immune system", *ICEC*, pp. 464–469, 1998.

[16] S. H. Russ, A. Lambert, R. King, R. Rajan, and D. Reese, "An artificial immune system model for task allocation", *Proc. of the Symposium on High Performance Distributed Computing*, 1999.

[17] L. N. De Castro and F. J. Von Zuben, "The clonal selection algorithm with engineering applications", *GECCO 2000 - Workshop proceedings*, pp. 36–37, 2000.

[18] K. C. Tan and R Narasimhan, "Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach", *Omega, Int. Journal Mgmt. Science*, vol. 25 (6), pp. 619–634, 1997.

[19] K. K. Lai and J. W. M. Chan, "Developing a simulated annealing algorithm for the cutting stock problem", *Comp. and Indust. Engn*, vol. 32(1), pp. 115–127, 1997.

[20] Y. Li and D. Wang, "A semi-infinite programming model for earliness/tardiness production planning with simulated annealing", *Mathematical and Computer Modelling*, vol. 37(1-2), pp. 277–280, 1997.

[21] S. Tonegawa, "Somatic generation of antibody diversity", *Nature*, vol. 302, pp. 575–581, 1983.

[22] S. Tonegawa, "The molecules of the immune system", *Scientific American*, vol. 253(4), pp. 104–113, 1985.

[23] R. Berreta and P. Moscato, *The number partitioning problem: an open challenge for evolutionary computation ?, Chapter 17 of* New ideas in optimization, D. Corne, F. Glover, and M. Dorigo, (eds.), MacGraw Hill, 1999.

[24] R Korf, "A complete anytime algorithm for number partitioning", *Artificial Intelligence*, vol. 106(2), pp. 181–203, 1998.