SEEA FOR MULTI-OBJECTIVE OPTIMIZATION:

REINFORCING ELITIST MOEA THROUGH MULTI-PARENT CROSSOVER, STEADY ELIMINATION AND SWARM HILL CLIMBING

Yan Zhenyu [1], Kang Lishan [1], Bob McKay [2], Fu Penghui [1]
[1] State Key Laboratory of Software Engineering Wuhan University, Wuhan 430072, P. R. CHINA
[2] School of Computer Science, UC, UNSW Australian Defence Force Academy Northcott Drive, Canberra, ACT 2600 AUSTRALIA
zyyan@whu.edu.cn

ABSTRACT

Non-elitist Multi-objective Evolutionary Algorithm does not guarantee that desirable solutions, once found, remain in the generational population until MOEA termination because of the EA's stochastic nature. Elitist MOEA handles this drawback with elitism. However the implementation of elitism leaves the user with many undetermined parameters; choosing proper elite-preserving operators becomes a difficult decision. Furthermore, this strategy increases the space and time complexity. This paper presents a new MOEA: Steady Elimination Evolutionary Algorithm. Through the strategies of steady elimination, multi-parent crossover and swarm hill climbing, SEEA can preserve elites effectively and instinctively without increasing computational complexity, and can get good uniform solutions representing the Pareto Front. The simulation experiments show that SEEA has some advantages for solving Multi-Objective Problems.

1. INTRODUCTION

In recent years, researchers always classify MOEA into two categories: no-elitist MOEA and elitist MOEA[6]. Deb holds the view that elitist MOEAs outperform no-elitist MOEAs significantly [6][4]. But there are many questions not yet determined regarding the realization of elitism. Which solutions should be kept, and for how long, in the elite set? When and how are members of the elite set reinserted into the population? How should one choose an appropriate ∂ value [6]? And how should one choose good elitist elite-preserving operators? Furthermore elitism has an inevitable side effect, that the archive and elitist-preserving operators will increase the space and time complexity.

In this study we present a new MOEA: the Steady Elimination Evolutionary Algorithm. By using the steady elimination approach and other strategies, MOEA can preserve elites instinctively, and get good uniform solutions representing the Pareto Front, without using archives or other elite-preserving operators, therefore it doesn't require extra time and space complexity.

The paper is structured as follows: section 2 introduces the concept and definition of elitism. We then discuss its important function in MOEA. In section 3, we present SEEA and discuss its new features and advantages. In section 4, we illustrate the power of SEEA by some experiments and discuss its efficiency. Furthermore, we prove that SEEA realizes elitism instinctively.

2. THE CONCEPT AND FEATURES OF ELITISM

Recently several authors indicated that elitism could improve evolutionary multi-objective search significantly [1][4][5][6][10].

There are various ways to incorporate elitism into MOEA. Most algorithms today make use of a second population of elite individuals. The Strength Pareto EA (SPEA), for example, stores all nondominated solutions separately from the 'normal' population. Individuals are chosen from the population and the nondominated set as recombination partners, while the latter get higher selection probabilities.

Though there are great varieties in the implementation of elitism, we can summarize some important features as follows: First, they require preserving the best solutions in the population, an archive or sub-population. Second, they require consideration of several strategies, including the elitism strategy, or how the elitist population is updated; the re-insertion strategy, or how elite individuals take part in the production of offspring; and the control flow, or when archiving and re-insertion take place. Unfortunately, particular implementations leave the determination of many parameters to the decision makers. They also inevitably increase the space and time complexity. Nevertheless, the evidence shows that elitism is an important and indispensable factor in EMOO.

According to Laumanns [10] and Deb [6], elitism means that 'elite' individuals cannot be expelled from the active gene-pool of the population in favor of worse individuals. This concept can be formalized as follows:

Definition 1(Elitism) *Let P^t denote the population of a given evolutionary algorithm EA after $t \in \mathrm{N}$ iterations(generations). Let $p^t(x)$ denote the probability of individual $x \in P^t$ being selected as an operand for the variation operator in generation t. Then EA is said to be elitistic, if and only if for any preference relation $\prec$ given by a decision problem the following condition holds:*

$$\forall t \in \mathrm{N} : \exists x \in \rho^{*t} : p^{t+1}(x) > 0$$

$$\text{with} \quad \rho^{*t} = \{ x \in \rho^t : \neg\exists x' \in \rho^t : x' \prec x \},$$

$$\rho^t = \bigcup_{\tau \le t} P^\tau$$

In this definition ρ^t refers to the set of all individuals produced so far and ρ^{*t} to all non-dominated individuals out of ρ^t.

3. SEEA

We propose a new approach for Multi-objective Optimization: Steady Elimination Evolutionary Algorithm (SEEA) SEEA uses the steady elimination strategy: to generate only one new individual in each generation. If the new one is better than the worst of the current population, the former will supersede the latter. We use a multi-parent crossover operator and real-valued encoding without a mutation operator.

3.1. The flow of SEEA

The flow of the algorithm is as follows:

Algorithm 1:

Step1: t=0, generate an initial population P^t.

Step2: Calculate rank and nichecount of the individuals in P^t.

Step3: Locate X_{best} and X_{worst} by sorting P^t using ***better***.

Step4: Select ***m*** individuals randomly from P^t and create a new individual X_{son} by multi-parent crossover. Calculate the rank and nichecount of X_{son}

Step5: If ***better*** (X_{son} , X_{worst}), then X_{son} supersede X_{worst}.

Step6: If a stopping criterion is satisfied then terminate, else t=t+1 go to step2.

3.2. *Better*

We compare individuals using ***better*** rather than assigning fitness to them in traditional EAs, so that ***better*** is the basic motivator of the evolution.

The return value of ***better(*** x_1 , x_2 ***)*** is Boolean. The flow of the algorithm is as follows:

Algorithm 2:

Step1: If x_1 dominates x_2, return true, otherwise go to step2.

Step2: If x_1 .rank< x_2 .rank, return true , if x_1 .rank> x_2 .rank, return false, otherwise go to step3。

Step3: If x_1 .nichecount < x_2 .nichecount，return true, otherwise return false。

3.3. Pareto ranking

We use the technique proposed by Fonseca and Fleming (1998)[3] to calculate Pareto rank. A solution *x* at generation *t* has a corresponding objective vector x_u . We let r_u^t signify the number of vectors associated with the current population dominating x_u ; *x*'s rank is then defined by:

$$\text{rank}(x,t)= r_u^t \quad (3.1)$$

This ensures that all solutions with nondominated vectors receive rank zero and that the maximum rank won't be larger than N. By this means, we convert multiple objective values to a ranked metric by using the non-domination principle.

As far as implementation is concerned, we need to consider the computational complexity.

We suppose that the size of population is N and the number of objects is M. We need M*N*(N-1) comparisons for ranking in each generation using equation 3.1. So the computational complexity is O(N^2). Suppose that the number of individuals created in one generation is K(usually K=N). Then on average, a new individual needs M*N*(N-1)/ K comparisons for ranking. But in SEEA, we generate only one new individual in each generation, so that K =1. The times of comparison for each individual will be M*N*(N-1) if we calculate ranks using equation 3.1 in every generation. It is clear SEEA will consume more time if we use the same way of ranking as traditional MOGA.

We use dominate_list, an N*N dimensions matrix, to record the dominative relation of individuals. If individual *j* dominates individual *i*, then dominate_list[*i,j*]=1. At the beginning of SEEA we initialize dominate_list by equation 3.1. In the later iteration of the algorithm, we only update the row and column relevant to the newly updated individual. So we need at most 2N comparisons in each generation for ranking and the computational complexity reduces to O(N).

Experiments show that these improvements enhance the efficiency of the algorithm effectively. It is well known that ranking and niching dominate the time performance of MOEA execution. Here we use a classic time-space tradeoff, significantly improving the time performance while increasing the space complexity to some extent, a tradeoff we consider to be profitable and reasonable.

3.4. Pareto Niching

We use niches to preserve the diversity of the population in order to find a uniform distribution of vectors. Here we use phenotypic space[3]. We define the normalized distance between two solutions *i* and *j* in the same rank as follows:

$$d(i,j)=\left[\sum_{k=1}^{M}\frac{(f_k^{(i)}-f_k^{(j)})^2}{(f_k^{\max}-f_k^{\min})^2}\right]^{\frac{1}{2}} \qquad (3.2)$$

Where $f_k^{\max}$ and $f_k^{\min}$ are the maximum and minimum objective function values of the k-th objective respectively. $f_k^{(i)}$ and $f_k^{(j)}$ are the objective function values of *i* and *j*. For the solution *i*, *d(i,j)* is computed for each solution *j* having the same rank. The sharing function value is computed as follows:

$$Sh(d)=\begin{cases}1-(\frac{d}{\sigma_{share}})^{\alpha}, & if \quad d\le\sigma_{share};\\ 0, & otherwise.\end{cases} \qquad (3.3)$$

σ_{share} is a maximum phenotypic distance allowed between any two individuals to become a member of a niche. Thereafter, the niche count of *i* is calculated by summing the sharing function values:

$$nc_i=\sum_{j=1}^{N}Sh(d_{ij}) \qquad (3.4)$$

Similarly to ranking, we also use niche_list to reduce the computational complexity of niching.

3.5. Realization of multi-parent crossover

Definition 2

$V=\{X \in D \mid X=\sum_{i=1}^{M} a_i x_i\}$ *is a subspace of D formed by* ***m*** *points* x_i *(i=1,2,…..,m).*

Where *D is the searching space*

$x_i \in D$, *i=1,2,…..,m*

$$\sum_{i=1}^{m} a_i = 1, \quad -0.5 \le a_i \le 1.5, \quad 1 \le i \le m$$

We realize multi-parent crossover by selecting one point randomly from V.

4. NEW FEATURES OF SEEA

4.1. Multi-parent crossover

Guo Tao[7] has discussed the positive features of multi-parent crossover. His experiments show that: with a fixed population size, the larger m (the number of parents) is, the greater the probability of convergence to the optimum. Alternatively**,** the larger *m* is, the smaller the size of population required to retain the probability of convergence to the optimum. These characteristics of multi-parent crossing are also exhibited when we use it in SEEA to solve MOP. Note that the weight coefficient a_i ranges from – 0.5 to 1.5. This implies that V is a nonconvex space, which guarantees that the set $V^* = \bigcup_{\tau \le t} V^\tau$ *(t is the iterations)* can cover the whole search space D, when $t \to \infty$. The experiments also show that the convergence speed of SEEA to the Pareto Front is increased when we accrete ***m*** properly.

4.2. Steady elimination strategy

We generate only one new individual and may eliminate at most one individual (the worst) in each generation. The selection pressure is lower than in most alternative algorithms**.**

With MOEA, we want to find a uniform distribution of vectors representing (or approximating) Pareto Front in one simulation run. Some researchers propose Secondary Population [8][9], archive and elitism [4] to solve this problem, such as SPEA, which gets the best performance among the current algorithms. But these strategies will occupy extra space and time to maintain an archive, continually added to and periodically culled of solutions whose associated vectors are dominated.

Our algorithm does not require an archive to preserve elites in SEEA, rather it can preserve elites automatically and effectively, without increasing the space and time complexity of the algorithm, because of its steady elimination strategy. We will discuss this issue further in a later section with some experimental results.

4.3. Swarm hill climbing

If we decrease the population size of Guo's Algorithm[7] or SEEA to one, the algorithm reduces to hill-climbing**.**. It is a truism that hill-climbing algorithm generally converges to local optima**,** rather than finding the global optimum. However, here we use a population approach instead of seeking just one solution in each iteration. We call it swarm hill climbing; it has completely different properties from common hill climbing algorithms.

Classical beam-based hill climbing fosters independent individuals, which probe good solutions in the searching space. If an individual reaches a local peak, it faces the same problem as single-individual hill climbing, with low probability of escaping the local peak. However, swarm hill climbing fosters a group of interacting individuals. All individuals climb the hills in parallel. They compete with each other (by ***better***) and communicate with each other (by multi-parent crossover and ***better***): an individual in a local optimum will be eliminated by a newly created individual, if it is poor enough compared with other individuals and the new one. By this means, each individual has some probability of jumping onto another hill if it has not reached the global optimum. Because of this, swarm hill climbing is not as subject to premature convergence to local optima as the deterministic classical algorithms.

Guo[7] has discussed its effect on single objective optimizations, based on some successful experiments. In SEEA, we take the advantage of swarm hill climbing to solve multi-objective optimization problems. It shows good search abilities in MOP, as in SOP.

4.4. New evaluation and selection methods: *better*

To design the fitness assignment method for MOEA, one must pay attention to two aims:

(1)Preference to nondominated solutions in a population

(2)Maintenance of diversity among nondominated solutions [1].

This is an inherent two-objective problem in MOEA. There are many ways to deal with this task:

MOGA[3] sorts solutions by rank and assigns fitness via linear or exponential interpolation. Then the assigned fitness values are divided by the niche count and scaled. This method transforms the two objectives (preference to nondominated solutions and maintenance of diversity) into one by summing. But it does not ensure that a solution in a poorer rank will always have a worse scaled fitness than every solution in a better rank. The reverse may happen particularly if there exist many crowded solutions with a better rank. The niche count for these solutions would be large and the resulting shared fitness may be small. If this happens, adequate selection pressure may not exist to all solutions in a better rank, thereby leading to a slow convergence or inability to find a good spread in the Pareto-optimal front.

In order to avoid this drawback, NSGA use another method [2], In this method, no solution in the first front has a shared fitness worse than the assigned fitness of any solution in the second front, which ensures that the rank pressure has more priority than the niche pressure.

In SEEA, we only need to sort the individuals in the population and compare X_{son} with X_{worst} . We can make this comparison directly without fitness assignment, in order to save the expenditure of extra time for this task. Therefore, in SEEA we design ***better*** to make the comparison rather than assign fitness to each individual. During the first step of ***better***, we see if x_1 dominates x_2 . Obviously, if x_1 dominates x_2 , x_1 .rank< x_2 .rank. If x_1 does not dominate x_2 , we compare their ranks directly in the second step. If their ranks are equal, we compare their niche count in the third step. By ***better*** we can get the same comparison results as the results we get by using the NSGA's fitness assignment

method. The two methods both make sure that the rank pressure has more priority than the niche pressure, while ***better*** compares the individuals directly, saving the time cost of fitness assignment.

It is reasonable to give the rank pressure more priority. In the initial stage of optimization, we emphasize the extent the solutions approach the Pareto Front, rather than their uniformity. In the late stage of optimization, the solutions are near to Pareto Front, and the ranks of individuals are almost zero so that the rank pressure is rather small. In this stage, we mainly make the solution more uniform through niche pressure.

5 EXPERIMENTS AND DISCUSSION

5.1. Test functions

F_1 is suggested by Deb [6], who used it to compare the main current MOEAs. For all Pareto-optimal solutions, two objective functions are related as $f_2 = 1/f_1$ （$0.1 \le f_1 \le 1$）, thereby constituting the trade-off among the Pareto-optimal solutions.

Minimize $f_1(x) = x_1$

Minimize $f_2(x) = (1+x_2)/x_1$

Subject to $0.1 \le x_1 \le 1,\ 0 \le x_2 \le 5$

Function F_2, F_3 are proposed by Zitzler [4]. Each of the test functions defined below is structured in the same manner and consists itself of three functions f_1, g, h.

Minimize $F(X) = (f_1(x_1),\ f_2(x_2))$

Subject to

$f_2(X) = g(x_2, \ldots\ldots, x_m)\, h(f_1(x_1), g(x_2, \ldots\ldots, x_m))$

Where $X = (x_1, \ldots\ldots, x_m)$

Function F_2:

$f_1(x) = x_1$

$g(x_2, \ldots\ldots, x_m) = 1 + 9 * \sum_{i=2}^{m} x_i/(m-1)$

$h(f_1, g) = 1 - \sqrt{f_1/g}$

Where m=30, and $x_i \in [0,1]$. The Pareto optimal front is formed with g(x)=1.

Function F_3:

$f_1(x) = x_1$

$g(x_2, \ldots\ldots, x_m) = 1 + 9 * \sum_{i=2}^{m} x_i/(m-1)$

$h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$

Where m=30, and $x_i \in [0,1]$. The Pareto optimal front is formed with g(x)=1.

5.2. Simulation results

To compare SEEA with other MOEAs, we use parameters corresponding to the references from which they are derived.

The concept of generation in SEEA is different from traditional MOEAs. Only one new individual is created in each generation in SEEA, therefore, it is more reasonable to regard the number of new individuals generated during the run as the standard for comparison, rather than number of generations. Traditional MOEA will generate 40 individuals in every generation, if its population size is 40. In Deb [6], the generation parameter is 500, independent of the algorithm. So 500*40=20000 individuals will be generated. This would be generated by SEEA in 20000 generations (one individual one generation).

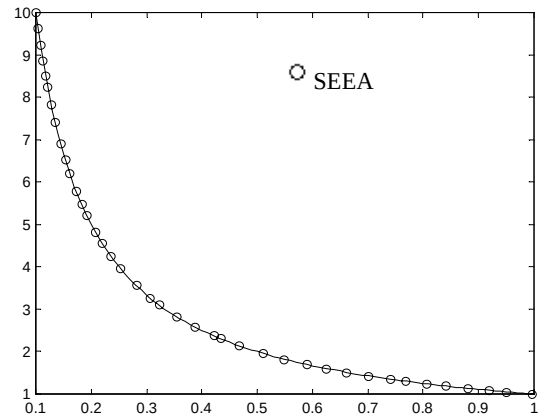

Figure 1: Test function F_1

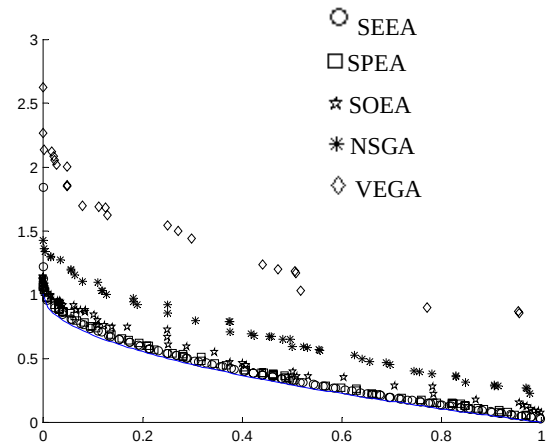

Figure 2: Test function F_2

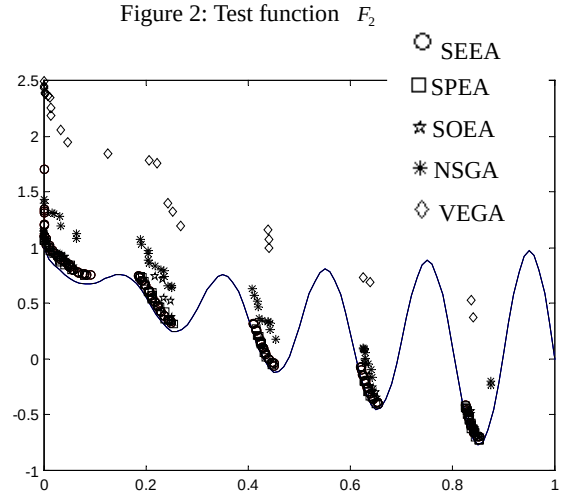

Figure 3: Test function F_3

F_1 , F_2 and F_3 were run using the following parameters:

	Pop size		*Generation Number*	σ_{share}
F_1	40(from [6])	9	20000(equal to 500*40)	0.038
F_2 F_3	100(from [4])	13	25000(equal to 250*100)	0.018

In Figure 1-3, the nondominated fronts achieved by SEEA of the simulation runs are compared to some other algorithms.

According to F_1 , SEEA outperforms all the nonelitist MOEAs listed by Deb [6] (the comparative data could not be printed in Figure 1, please refer to [6]), with respect to both distance to the Pareto-optimal front and distribution of the nondominated solutions. It also outperforms the elitist MOEAs with respect to uniformity.

As Figure2 and Figure3 show, SEEA outperforms SOEA, NSGA and VEGA, and it is as good as, if not better than, SPEA, the elitist MOEA using archive and elitist-preserving operator.

5.3. Analysis and conclusion

It is widely accepted that elitism is an indispensable strategy in MOEAs[1][6].

In SEEA, we do not use a second population to preserving elites intentionally with elite-preserving operator. Nevertheless**,** we get solutions, which are as good as, if not better than, elitist MOEAs. We infer that the steady elimination character of SEEA that preserves the elites automatically and realizes elitism without the extra time and space complexity.

Here we revise definition 1 to a stricter condition:

$$\forall t \in \mathrm{N} : \forall x \in \rho^{*t} : p^{t+1}(x) > 0$$

And we show that SEEA fulfills it.

Conclusion1. *SEEA realize elitism.*

***Proof**:*

In SEEA*(Algorithm1)*, at $\forall t \in \mathrm{N}$ generation, we select ***m*** parents randomly from the current population P^t to generate multi-parent crossover. So it is easy to draw the conclusion that:

$$\forall t \in \mathrm{N} : \forall x \in P^t : p^{t+1}(x) > 0$$

So in order to prove SEEA realize elitism, we only need to prove that:

$\forall t \in \mathrm{N} : \forall x \in \rho^{*t} : x \in P^t$ (based on **definition 1** and our more strictly conditional revision)

The $\prec$ in **definition 1** is instantiated with ***better*** *(Algorithm 2)*, which incorporates the components of rank and niche, in SEEA.

Suppose $\exists t \in \mathrm{N} : \exists x \in \rho^{*t} : x \notin P^t$ and x was eliminated at $t' \in \mathrm{N}$ ($t' \le t$) generation. Then at t' generation, there must have been created an individual $x' : x' \prec x$. And x' was added into the population at t' generation, which is to say that $x' \in P^{t'}$ Its

Since $\rho^t = \bigcup_{\tau \le t} P^\tau$ and $t' \le t$, so $x' \in \rho^t$

So $\exists x' \in \rho^t : x' \prec x$,

So $x \notin \rho^{*t}$, which conflicts with our above supposition.

Thus, we can say that our previous supposition was wrong and it is true that

$$\forall t \in \mathrm{N} : \forall x \in \rho^{*t} : x \in P^t$$

Thus, we can draw the conclusion that SEEA realizes elitism. □

Furthermore, current elitist MOEAs must consume time and space to realize elitism with elite-preserving operators, whereas, SEEA need not. Moreover, multi-parent crossover and swarm hill climbing strategy are incorporated in SEEA, both of which lead to improvements in its search ability. As a conclusion, the proposed Steady Elimination Evolutionary Algorithm shows some advantages in solving Multi-Objective Problems.

ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China (No. 60073043,70071042,60133010)

REFERENCES

[1] David A.VanVeldhuizen and Gary B.Lamont.Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art Evolutionary Computation, 2000, 8(2) 125-147

[2N.Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting In Genetic Algorithms, Evolutionary Computation1995,2(3)221-248

[3]Carlos Fonseca and Peter J.Fleming Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I:A Unified Formulation Research Report , 1995, 564

[4]Zitzler,E.and Thiele, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Compution, 1999,3(4), 257-271

[5]Horn,j and Nafpliotis,N. and Goldberg,D.E. A Niched Pareto Genetic Algorithm for Multiobjective Optimization, CEC 1994, pages 82-87, IEEE Press, Piscataway, New Jersey.

[6] Kalyanmoy Deb. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley&Sons, Ltd Baffins Lane, Chichester, West Sussex, PO19 IUD,England, 2001

[7]Guo Tao and Kang Li-shan A new Evolutionary Algorithm for Function Optimization Wuhan University Journal of Natural Sciences Vol.4 No.4 1999,409-414

[8]Horn ,J. Multicriterion Decision Making. Handbook of EvolutionaryCompution, Oxford University Press, Oxford, England, 1997, Volume 1,Pages F1,9:1-f1.9:15.

[9]Rudolph,G..On a Multi-Objective Evolutionary Algorithm and Convergence to Pareto Set. CEC 98, IEEE Press, Piscateway, New Jersy 1998, Pages 516.

[10]Marco Laumanns and Eckart Zitzler. A Unified Model for Multi-Objective Evolutionary Algorithms with Elitism. 2001, Proceedings of the 2001 IEEE Congress on Evolutionary Computation Seoul, Korea May 27-30, 2001.