An Artificial Bee Colony with Random Key for Resource-Constrained Project Scheduling

Yan-jun Shi[1], Fu-Zhen Qu[1], Wang Chen[2], and Bo Li[2]

School of Mechanical Engineering,
[1]Dalian University of Technology, Dalian, P.R. China 116024
[2]China North Vehicle Research Institute Beijing, P.R. China 100072
syj@dlut.edu.cn

Abstract. This paper proposes an artificial bee colony (ABC for short) algorithm with random key for resource-constrained project scheduling (RCPSP for short) in real time. Aim at resource saving by the activities, the RCPSP problem attempts to obtain a feasible schedule minimizing the makespan. We modified the artificial bee colony algorithm (named by ABC-RK) for this problem, where the problem representation was based on random key, and a heuristic priority rule to assign activities was also employed. The preliminary experimental results showed the effectiveness of the ABC-RK algorithm.

Keywords: resource saving, artificial bee colony, project scheduling, random key.

1 Introduction

As a class of evolutionary computation or artificial evolution, the artificial bee colony (ABC for short) algorithm is a meta-heuristic intelligence optimization algorithm that motivated by the intelligent behavior of honey bees, and based on bee foraging model proposed by Karaboga in 2005[1]. Karaboga and Basturk had used ABC algorithm[2] to tackle the constrained optimization problems, and compared its performance with GA, PSO, and etc. They have extended ABC algorithm for training neural networks, medical pattern classification and clustering problems[3]. In previous studies, the ABC algorithm had showed the effectiveness and efficiency on algorithm performance, and used only common control parameters such as colony size and maximum cycle number. However, there is little report in the available literature on employing the ABC algorithm to tackle RCPSP problem. This study attempts to modify ABC algorithm to deal with such problems by employing random key technique.

The RCPSP problem continues to be an active area of research and has attracted increasing interest from researchers and practitioners searching for better solution methods[4]. This problem involves finding a feasible schedule minimizing the makespan, that is, the total duration, of a project consisting of a set of activities with known deterministic durations. And for resource saving, this problem is also subject to precedence constraints between activities and accumulative constraints related to resource availability and resource consumption by the activities. The solution

K. Li et al. (Eds.): LSMS/ICSEE 2010, Part II, LNCS 6329, pp. 148–157, 2010.

methods included exact and heuristics approaches. To solve the practical larger scale problem instances with up to 60 activities in a satisfactory manner, only heuristics approaches can remain the feasible way. Thus, heuristic solution strategies appear to be preferred when dealing with larger problem instances. Many researchers have proposed various heuristics and meta-heuristics to solve the underlying RCPSP in real-time[4]. Several previous studies provided detailed descriptions of different heuristic methodologies employed to solve the RCPSP. These methodologies may be classified into several categories: (1) priority-rule-based X-pass methods, (2) classical meta-heuristics, (3) local-search-oriented approaches, and (4) population-based approaches[5]. In population-based approaches, ant colony optimization (ACO), genetic algorithm, and PSO had successfully applied to several NP-hard combinatorial optimization problems. Also as a population-based approach, the ABC algorithm was not reported for solving this problem to the best of our knowledge. This study attempted to report the preliminary experimental results from the ABC-RK algorithm.

2 The Resource-Constrained Project Scheduling Problem

This study describes the classical RCPSP problem as follows[6]. A project consists of a set of activities $V=\{0,1,...,n,n+1\}$ and K renewable resource types. We denote the duration of an activity j in V as d_j. The availability of each resource type $k \in K$ in each time period is R_k and each activity j requires r_{jk} units of resource k during each period of its duration. The dummy activities 0 and $n+1$ represent the beginning and the end of the project, where $d_0 = d_{n+1} = 0$ and $r_{0k} = r_{n+1k} = 0$. Parameters d_j, r_{jk}, and R_k are assumed to be non-negative integer values. The activities are interrelated by two types of constraints: (1) precedence constraints prevent activity j from being started before all its predecessor activities P_j have been completed, and (2) the sum of the resource requirements for resource k in any time period cannot exceed R_k. The optimized objective of the RCPSP is to find precedence and resource feasible completion times for all activities minimizing makespan of the project. Figure 1 shows an example of a project comprising $n=6$ activities that need to be scheduled subject to $K=1$ renewable resource type with a capacity of 6 units.

Let the completion time of activity j be denoted as F_j, then the completion times of schedule S are denoted as $(F_1, F_2, ..., F_n)$. The conceptual model of the RCPSP can be described as follows[7].

$$\min \ F_{n+1} \tag{1}$$

$$s.t. \quad F_h \le F_j - d_j \qquad j=1,...,n+1; h \in P_j, \tag{2}$$

$$\sum_{j \in A(t)} r_{j,k} \le R_k \qquad k \in K; t \ge 0 \quad , \tag{3}$$

$$F_j \ge 0 \qquad j=1,...,n+1 \quad . \tag{4}$$

The objective function given by Eq. (1) minimizes the makespan of the schedule. Eq. (2) enforces the precedence constraints between activities, while Eq.(3) enforces the resource limitation constraint, where $A(t) = \{ j \in V \mid F_j - d_j \le t < F_j \}$, i.e., the set of activities being processed (active) at time t. Finally, Eq. (4) describes the constraint of the decision variables

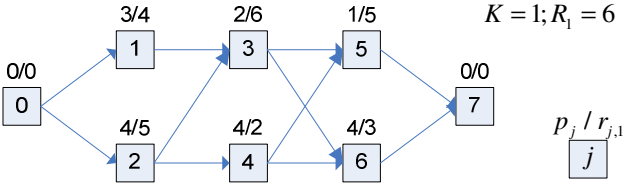

Fig. 1. A project example[6]

3 The Proposed ABC-RK Algorithm

The standard ABC algorithm uses a colony of artificial bees[1], which are divided into three groups which are employed bees, onlooker bees, and scout bees. We extend the standard ABC algorithm with random key for the RCPSP problem, which is called by ABC-RK algorithm and is detailed below.

3.1 A Brief Introduction for ABC Algorithm

Like GA, the ABC algorithm also provides a population-based search procedure[3], where individuals are called foods positions and modified by the artificial bees with time. The artificial bees fly around in a multidimensional search space. Some bees, i.e., employed and onlooker bees, choose food sources depending on their experience or nest mates, and adjust their positions. Some bees, i.e., scouts bees, choose the food sources randomly without using experience. If the nectar amount of a new source is higher than that of the previous one in their memory, they memorize the new position and forget the previous one. Thus, the ABC algorithm takes the employed and onlooker bees to make local search, and takes onlookers and scouts to make global search. In this way, the ABC algorithm makes a robust search process, and carries out exploration and exploitation together.

3.2 Representation and Decoding Procedure for RCPSP

The representation and decoding procedure for RCPSP play an important role in algorithm performance. Ref.[8] listed some representations used within meta-heuristic approaches: activity-list representation, random-key representation, priority rule representation, shift vector representation, schedule scheme representation, etc. The

random–key (RK) representation and the activity-list (AL) representation are most usually been used in literatures. Hartmann and Kolish[9] compared the experimental results of current state-of-the-art heuristics, and reported that in general, procedures employing AL representation will leads to better results than other representations for the RCPSP. Debels et al.[10] considered the main reason for the inferior performance of the RK representation lied in the fact that one single schedule can have many different representations. Although AL representation also suffers from this case, the frequency of this case occurs more higher when using RK representation. However, the RK representation had the advantage[10] that each solution corresponds to a point in Euclidian space, so that geometric operations can be performed on its components. Previous studies in the ABC algorithm usually focused on continuous search space. Therefore, we herein employ random-key representation. That is, a solution is represented by an array $r = (r_1, r_2, ..., r_n)$, which assigns a real-valued number $r_j (0 < r_j < 1)$ to each activity j .

In random key representation, schedule generation schemes (SGS) are the core decoding procedures for the RCPSP[8]. There are two different SGS available: (1) the so-called serial SGS performs activity-incrementation, and (2) the so-called parallel SGS performs time-incrementation. The difference between the serial and the parallel SGS is that the serial SGS constructs so-called active schedules while the parallel one constructs so-called non-delay schedules[11]. The set of the non-delay schedules is a subset of the set of the active schedules. While the set of the active schedules always contains an optimal schedule, this does not hold for the set of the non-delay schedules. Consequently, the parallel SGS may miss an optimal schedule. On the other hand, it produces schedules of good average quality because it tends to utilize the resources as early as possible, leading to compact schedules. This difference leads to the different behaviors of the SGS in computational experiments[9][12].

Considering the pros and cons of the serial SGS and the parallel SGS, we construct a schedule by employing a random real number $r_{SGS} (0 \le r_{SGS} \le 1)$ to select SGS, which is detailed as

$$\text{SGS} = \begin{cases} serial-\text{SGS}, & if \quad r_{SGS} \le 0.5 \\ parallel-\text{SGS}, & if \quad r_{SGS} > 0.5 \end{cases}, \tag{5}$$

That is, for a given individual, we judge a random r_{SGS} to decide a schedule of SGS, where the fitness of the individual is defined as the makespan of the schedule.

For any random key representation, both the serial and parallel SGS can be directly used to derive a schedule from ρ . On each stage g of SGS, we must select an activity j from the decision set D_g following some rules. Since the random keys play the role of priority values, what we need to do is just to select an efficient priority rule. The latest finish time (LFT)[9]heuristic is used in this study because it is reported to be an outstanding heuristic for evolutionary algorithm. Therefore, on the stage g , we will select activity j with the minimum random key $r_j = \min\{r_i \mid i \in D_g\}$ from the decision set D_g .

3.3 The ABC-RK Algorithm for RCPSP

Using the mentioned-above solution representation, the ABC-RK algorithm for RCPSP can be seen in Fig. 2. Firstly, the ABC-RK generates a randomly distributed initial population of *SN* solutions, i.e., food source positions. Each solution x_i ($i \in [1, SN]$) is a *D*-dimensional vector. Then, we evaluate the initial population after random key based solution encoding is done. Moreover, an iterative process with max generation *N* is searched through the employed bees, the onlooker bees and scout bees:

(1) An employed bee produces a modification on the position (solution) in her memory depending on the local information (visual information) and tests the nectar amount (fitness value) of the new source (new solution).

(2) An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with a probability related to its nectar amount.

(3) A scout bee investigates the food source if the nectar is abandoned. Then, we decide that the bee is replaced with a new food source by the scouts or not.

Furthermore, when the *Best* solution is the ideal solution or we have run out of generation *N*, the iterative process is terminated.

```
i ← number of food source positions
j ← number of optimization parameters
Initialize the population of solutions x_{i,j}
Apply random key to decode the solution for RCPSP
Evaluate the population for each solutions for RCPSP
Initialize best solution Best
repeat
    Produce new solutions v_{i,j} for the employed bees ①
    Apply random key to decode v_{i,j} for RCPSP
    Evaluate employed bees for each new solution
    Apply the greedy selection process
    Calculate the probability values p_{i,j} for the solutions x_{i,j} ②
    Select the onlookers from the solutions x_{i,j} depending on p_{i,j}
    Produce the new solutions v_{i,j} for the selected onlookers
    Apply the greedy selection process
    if the abandoned solution for the scout is existed then
        replace it with a new randomly produced solution
    end if
    if Fitness(v_{i,j}) > Fitness(Best) then
        Best ← v_{i,j}
    end if
until Best is the ideal solution or we have run out of generation
Output Best
```

Fig. 2. The proposed ABC-RK algorithm

In the above interactive process, the ABC-RK algorithm produce new solutions $v_{i,j}$ for the employed bees using (see Fig. 2 symbol ①)

$$v_{ij} = x_{ij} + \psi_{ij}(x_{ij} - x_{kj}), \tag{6}$$

where $k \in [1, SN]$, $j \in [1, D]$ are randomly chosen indexes, and $\psi_{ij} \in [-1,1]$ is a random number that controls the production of neighbor food sources around x_{ij} and represents the comparison of two food positions visually by a bee. Note if a parameter value produced by this operation exceeds its predetermined limit, the parameter can be set to an acceptable value.

The probability for an onlooker bee choosing a food source can be calculated by

$$p_i = fitness_i / \sum_{n=1}^{SN} fitness_n , \tag{7}$$

(See Fig. 2 symbol ②) where $fitness_i$ is the function fitness value of the solution i.

Additionally, the value of predetermined number of generations (cycles) is an important control parameter of the ABC algorithm, which is called "limit" for abandonment.

4 Experimental Study

We applied ABC-RK to the standard instance sets $J30$ (480 instances each with 30 activities), $J60$ (480 instances each with 60 activities), and $J120$ (600 instances each with 120 activities) generated by the problem generator ProGen devised by Kolisch and Sprecher[13]. These sets are available in the well known PSPLIB (http://129.187.106.231/psplib/) along with the optimum, or best known values that have been obtained by various authors over the years. Only in $J30$ are the optimal makespans for all instances known. In $J60$ and $J120$ the optimal makespans for some instances are not known and only upper bounds (current best solutions) and lower bounds are provided. The lower bounds are determined by the length of a critical path in the resource relaxation of the problem using a critical path method. For a fair machine-independent comparison, we limited the number of generated and evaluated schedules in ABC-RK to 1000, 5000, and 50000, respectively. This restriction allowed us to compare our results with those reported in the evaluation study of Kolisch and Hartmann[12] for several RCPSP heuristics proposed in the literature. ABC-RK was implemented in JAVA 1.6 and all the experiments were performed on a PC with 1.86GHz CPU and 1G RAM running the Windows XP operating system.

Tables 1-3 give the performances of ABC-RK and several RCPSP heuristics from the literature with 1000, 5000, and 50000 evaluated schedules. This study compared ABC-RK with priority-rule based sampling methods[14], genetic algorithms[4][15]-[19], scatter search[10], ant colony algorithms[20 ,tabu search[21] , etc.

Table 1 gives the average percentage deviation from the optimal makespan for $J30$. ABC-RK ranks 9th for 1000 schedules and 7th and 7th for 5000 and 50000

schedules, respectively. Table 2 showed the lower bound results for instances with 60 activities. ABC-RK ranks 10th and 7th for 1000 and 5000 schedules, respectively, and 7th for 50000 schedules. Table 3 gives the results for $J120$, where ABC-RK ranks 8th for 1000 schedules, and 7th and 8th for 5000 and 50000 schedules, respectively.

To highlight the actual performance, Table 4 lists the results of the heuristics together with the average and maximum computation times as well as the clock-cycle of the computer used. The performances of the reported methods were obtained from surveys[12] and the original papers. The performance of ABC-RK as shown in the table was limited to 5000 evaluated schedules. From Table 4, we can see that ABC-RK requires a shorter average computation time for $J30$, $J60$, and $J120$.

Table 1. Average deviation (%) from optimal makespan-ProGen set $J = 30$

Algorithm	References	Schedules		
		1000	5000	50000
GA, TS-path relinking	Kochetov and Stolyar[18]	0.10	0.04	0.00
GAPS	Mendes et al.[22]	0.06	0.02	0.01
ANGEL	Tseng et al.[20]	0.22	0.09	-
Scatter Search-FBI	Debels et al.[10]	0.27	0.11	0.01
GA-DBH	Debels et al.[15]	0.15	0.04	0.02
GA-hybrid, FBI	Valls et al.[19]	0.27	0.06	0.02
GA-FBI	Valls et al.[4]	0.34	0.20	0.02
ABC-RK	**This study**	**0.35**	**0.12**	**0.04**
Sampling-LFT, FBI	Tormos and Lova[14]	0.25	0.13	0.05
TS-activity-list	Nonobe and Ibaraki[21]	0.46	0.16	0.05

Table 2. Average deviation (%) from critical path lower bound-ProGen Set $J = 60$

Algorithm	References	Schedules		
		1000	5000	50000
GAPS	Mendes et al. [22]	11.72	11.04	10.67
GA-DBH	Debels et al.[15]	11.45	10.95	10.68
Scatter Search-FBI	Debels et al.[10]	11.73	11.10	10.71
GA-hybrid, FBI	Valls et al. [19]	11.56	11.10	10.73
GA, TS-path relinking	Kochetov and Stolyar[18]	11.71	11.17	10.74
ANGEL	Tseng et al.[20]	11.94	11.27	-
GA-FBI	Valls et al.[4]	12.21	11.27	10.74
ABC-RK	**This study**	**12.75**	**11.48**	**11.18**
GA-self-adapting	Hartmann[17]	12.21	11.70	11.21
GA-activity list	Hartmann[16]	12.68	11.89	11.23

Table 3. Average deviation (%) from critical path lower bound-ProGen Set $J = 120$

Algorithm	References	Schedules		
		1000	5000	50000
GA-DBH	Debels et al.[15]	34.19	32.34	30.82
GA-hybrid, FBI	Valls et al.[19]	34.07	32.54	31.24
GAPS	Mendes et al. [22]	35.87	33.03	31.44
Scatter Search-FBI	Debels et al.[10]	35.22	33.10	31.57
GA-FBI	Valls et al.[4]	35.39	33.24	31.58
GA, TS-path relinking	Kochetov and Stolyar[18]	34.74	33.36	32.06
GA-self-adapting	Hartmann[17]	37.19	35.39	33.21
ABC-RK	**This study**	**36.29**	**34.18**	**33.69**
Sampling-LFT, FBI	Tormos and Lova[14]	35.01	34.41	33.71
ANGEL	Tseng et al. [20]	36.39	34.49	-

Table 4. Average and maximum computation times for $J30$, $J60$ and $J120$

Algorithm	Reference	Result	CPU-time(seconds)		Computer
			Average	Max.	
(a) $J = 30$					
Decompos. & local opt.	[23]	0.00	10.26	123.0	2.3 GHz
Local search-critical	[24]	0.06	1.61	6.2	400 MHz
ABC-RK	**This study**	**0.12**	**1.15**	**4.17**	**1.86 GHz**
ANGEL	[20]	0.09	0.11	-	1 GHz
Population-based	[5]	0.10	1.16	5.5	400 MHz
(b) $J = 60$					
Decompos. & local opt.	[23]	10.81	38.8	223.0	2.3 GHz
Population-based	[5]	10.89	3.7	22.6	400 MHz
ABC-RK	**This study**	**11.18**	**2.12**	**13.5**	**1.86 GHz**
ANGEL	[20]	11.27	0.76	-	1 GHz
Local search-critical	[24]	11.45	2.8	14.6	400 MHz
Tabu search	[25]	12.05	3.2	-	450 MHz
(c) $J = 120$					
Population-based	[5]	31.58	59.4	264.0	400 MHz
Decompos. & local opt.	[23]	32.41	207.9	501.0	2.3 GHz
ABC-RK	**This study**	**33.18**	**30.8**	**239.8**	**1.86 GHz**
ANGEL	[20]	34.49	4.79	-	1 GHz
Local search-critical	[24]	34.53	17.0	43.9	400 MHz
Tabu search	[25]	36.16	67.0	-	450 MHz

These results show that ABC-RK is capable of providing near-optimal solutions for the instance set $J30$, and can produce good solutions for the large scale problems $J60$ and $J120$. The reason for the showed performance is that ABC-RK algorithm

combines local search methods, carried out by employed and onlooker bees, with global search methods, managed by onlookers and scouts, and then balance exploration and exploitation process to provide a good solution.

5 Conclusion

In this study, we proposed an artificial bee colony (ABC for short) algorithm with random key for resource-constrained project scheduling. The computational results of ABC-RK were compared with state-of-the-art heuristics on the standard instance sets $J30$, $J60$ and $J120$ from the PSPLIB. The preliminary experimental results indicated that, with a limited number of generated schedules, ABC-RK is capable of providing near-optimal solutions for a small scale RCPSP and large scale problems. Further studies will focus on enhancing the applicability and efficiency of the proposed ABC-RK algorithm for large-scale instances of RCPSP.

Acknowledgements

This work was supported by National Natural Science Foundation of P R China (Grant No. 50975039) and National Defense Basic Scientific Research Project of PR China (Grant No. B0920060901).

References

1. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Computer Engineering Department, Erciyes University, Turkey (2005)
2. Karaboga, D., Akay, B.: A comparative study of Artificial Bee Colony algorithm. Applied Mathematics and Computation 214, 108–132 (2009)
3. Karaboga, D., Akay, B.: Artificial Bee Colony (ABC) Algorithm on training artificial neural networks. In: IEEE 15th Signal Processing and Communications Applications, Inst. of Elec. and Elec. Eng. Computer Society, Eskisehir, Turkey (2007)
4. Valls, V., Ballestín, F., Quintanilla, S.: Justification and RCPSP: A technique that pays. European Journal of Operational Research 165, 375–386 (2005)
5. Valls, V., Ballestín, F., Quintanilla, S.: A Population-Based Approach to the Resource-Constrained Project Scheduling Problem. Annals of Operations Research 131, 305–324 (2004)
6. Chen, W., Shi, Y.-j., Teng, H.-f., Lan, X.-p., Hu, L.-c.: An efficient hybrid algorithm for resource-constrained project scheduling. Information Sciences 180, 1031–1039 (2010)
7. Christofides, N., Alvarez-Valdes, R., Tamarit, J.M.: Project scheduling with resource constraints: A branch and bound approach. European Journal of Operational Research 29, 262–273 (1987)
8. Kolisch, R., Hartmann, S.: Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. In: Weglarz, J. (ed.) Project Scheduling: Recent Models, Algorithms and Applications, pp. 147–178. Kluwer Academic Publishers, Berlin (1999)

9. Hartmann, S., Kolisch, R.: Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Research 127, 394–407 (2000)
10. Debels, D., De Reyck, B., Leus, R., Vanhoucke, M.: A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. European Journal of Operational Research 169, 638–653 (2006)
11. Sprecher, A., Kolisch, R., Drexl, A.: Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. European Journal of Operational Research 80, 94–102 (1995)
12. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174, 23–37 (2006)
13. Kolisch, R., Sprecher, A.: PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. European Journal of Operational Research 96, 205–216 (1997)
14. Tormos, P., Lova, A.: Integrating heuristics for resource constrained project scheduling: One step forward. Technicalreport, Department of Statistics and Operations Research, Universidad Politecnica de Valencia (2003)
15. Debels, D., Vanhoucke, M.: A Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem. Operations Research 55, 457–469 (2007)
16. Hartmann, S.: A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics 45, 733–750 (1998)
17. Hartmann, S.: A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics 49, 433–448 (2002)
18. Kochetov, Y., Stolyar, A.: Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In: Proceedings of the 3rd International Workshop of Computer Science and Information Technologies, Russia (2003)
19. Valls, V., Ballestin, F., Quintanilla, M.S.: A hybrid genetic algorithm for the RCPSP. Department of Statistics and Operations Research, University of Valencia (2003)
20. Tseng, L.-Y., Chen, S.-C.: A hybrid metaheuristic for the resource-constrained project scheduling problem. European Journal of Operational Research 175, 707–721 (2006)
21. Nonobe, K., Ibaraki, T.: Formulation and tabu search algorithm for the resource constrained project scheduling problem. In: Hansen, P. (ed.) Essays and Surveys in Metaheuristics, pp. 557–588. Kluwer Academic Publishers, Dordrecht (2001)
22. Mendes, J.J.M., Goncalves, J.F., Resende, M.G.C.: A random key based genetic algorithm for the resource constrained project scheduling problem. Computers & Operations Research 36, 92–109 (2009)
23. Palpant, M., Artigues, C., Michelon, P.: LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighbourhood Search. Annals of Operations Research 131, 237–257 (2004)
24. Valls, V., Quintanilla, S., Ballestín, F.: Resource-constrained project scheduling: A critical activity reordering heuristic. European Journal of Operational Research 149, 282–301 (2003)
25. Artigues, C., Michelon, P., Reusser, S.: Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research 149, 249–267 (2003)