Journal of Applied Sciences

ISSN 1812-5654

Journal of Applied Sciences 8 (21): 3938-3944, 2008
ISSN 1812-5654

An Ant Colony Algorithm for the Flowshop Scheduling Problem

[1]S.J. Sadjadi, [2]J.L. Bouquard and [1]M. Ziaee
[1]Iran University of Science and Technology, Tehran, Iran
[2]University François-Rabelais, Tours, France

Abstract: In this study, we considered the flowshop scheduling problem with the objectives of the makespan ($F//C_{max}$) and the total flowtime ($F//\Sigma f_i$) separately. The permutation case of the problem was first solved by an Ant Colony Optimization (ACO) algorithm. The permutation solutions of this ACO algorithm were then improved by a non-permutation local search. In order to evaluate the performance of the proposed metaheuristic, computational experiments were performed using the well-known benchmark problems. A comparison with Rajendran solutions and the best metaheuristic solutions known for Taillard benchmark problems was carried out, show that the proposed ACO algorithm was clearly superior to the above metaheuristics.

Key words: Scheduling, flowshop scheduling problem, ant colony algorithm, metaheuristic, local search

INTRODUCTION

In a flowshop problem, a set of jobs must be processed on a number of sequential machines, each job has to be processed on all machines and the processing routes of all jobs are the same. In the general (non-permutation) case of the flowshop scheduling problem, the sequence of jobs on each machine may be different from the sequence of jobs on another machine. In the permutation case, these sequences are the same for all machines.

It is proved that the permutation flow shop does not necessarily provide an optimal solution (Liao *et al*., 2006). These people show that the performance of nonpermutation schedules is better than that of permutation schedules. In general, when there are more than three machines, permutation schedules are no longer dominant (Koulamas, 1998). In comparison with the permutation schedules, the performance of non-permutation schedules may be considerable if the problems have nonregular performance measures like maximum tardiness or weighted mean tardiness (Liao *et al*., 2006). The non-permutation schedules may provide much better solutions in such situations. However, almost all existing researches have focused on permutation schedules and there is a lack of sufficient analysis on non-permutation scheduling problems in the literature (Cheng *et al*., 2000).

As the problem size grows bigger, identifying the best permutation schedule itself becomes quite difficult. Obviously, finding an optimal solution when sequence changes are permitted is more complex and difficult (Pugazhendhi *et al*., 2002). In the literature, heuristic methods have usually been used to solve the non-permutation flowshop scheduling problems. Some of them are Jain and Meeran (2002) and Koulamas (1998).

Two widely used objectives in the literature are the makespan and the total flowtime, which are based on the completion times. The literature abounds with numerous and very different techniques for the permutation flowshop scheduling problem with these objectives. Ruiz and Maroto (2005) have presented an extensive review and evaluation of many heuristics and metaheuristics for the permutation flowshop scheduling problem with the makespan criterion. Varadharajan and Chandrasekharan (2005) consider the bicriteria permutation flowshop scheduling problem with the objectives of minimizing the makespan and total flowtime of jobs and present a Multi-Objective Simulated-annealing Algorithm (MOSA) for solving the problem. Rajendran and Hans (2004) have recently presented two ant colony algorithms (M-MMAS and PACO) for solving the permutation flowshop scheduling problem with the objectives of makespan and total flowtime. The solutions of their methods are the best results obtained so far on only one mainframe. We show that the method proposed in this paper can obtain better solutions than the M-MMAS and PACO. A similar work has been done by Vallada and Ruben (2008) for the makespan criterion. Their research is a state of the art on the literature about the permutation flowshop scheduling problem with makespan. They propose cooperative metaheuristic methods for the problem. They use the

Corresponding Author: Mohsen Ziaee, Department of Industrial Engineering, Iran University of Science and Technology, Narmak, Tehran, Iran Tel: +98-581-2222766 Fax: +98-21-77240482

island model where each island runs an instance of the algorithm. We compare our method with the best solutions of this cooperative metaheuristic (with 12 islands) and show that our solutions are close to those reported by Vallada and Ruben (2008) but they have used simultaneously 12 processors.

In this study, we consider the m-machine n-job flowshop scheduling problem with the objective functions of the makespan and the total flowtime separately. We assume that the ready times of all the jobs are equal to zero. Therefore, the total flowtime is equal to the total completion time. We develop an ACO algorithm to solve the permutation case of the problem. The permutation solution of the proposed ACO algorithm is then improved by a non-permutation local search. Therefore, the end solution of the method may be a non-permutation schedule. In order to evaluate the performance of the proposed metaheuristic, we implement the method for the benchmark problems of Taillard (1993) and present the results of the computational experiments. Finally, the conclusion remarks of this study are presented at the end to summarize the contribution of the study.

AN ACO ALGORITHM FOR SOLVING THE PROBLEM

Liao *et al*. (2006) show that there is little improvement made by non-permutation schedules over permutation schedules with respect to the completion-time based criteria such as makespan and total flowtime. Therefore, the proposed ACO algorithm is designed for generating only the permutation sequences. However, we store n' best permutation solutions of the ant colony procedure and then improve all of them by a rapid non-permutation local search. Storing best solutions for improving them may have an important role to increase the performance of the method. The steps of the proposed ant colony algorithm are as follows. Let z* be the objective function of the best sequence obtained so far (BS). I, K are the number of jobs and the number of machines respectively.

Step 1 (finding a seed permutation schedule): The algorithm uses a constructive method to find an initial permutation schedule. We use the NEH heuristic (Nawaz *et al*., 1983) to generate an initial permutation sequence for the objective of minimizing the makespan and a heuristic proposed by Woo and Yim (1998) to generate an initial permutation sequence for the objective of minimizing the total flowtime of jobs. These heuristics are powerful methods for solving a permutation flowshop scheduling problem (Ruiz and Concepcion, 2005; Woo and Yim, 1998).

Step 2 (first local search): The first local search is done using the pair-wise exchanges (Allahverdi, 2003), on the sequence as follows. We keep n-1 best solutions.

2-1) Let: counter = 1, n = 1,
2-2) Imprv = 0,
2-3) For y_1 = 1 (1) (I-1) :
For $y_2 = (y_1+1)$ (1) I :

Replace the job in the position y_1 with the job in the position y_2 without changing the positions of the other jobs. If the objective function of the resulted permutation sequence is less than or equal to z*, do the following settings:

- Set the resulted sequence to δ_n and its objective function to z*
- n = n+1,
- Imprv = 1,

Otherwise, replace again the job in the position y_1 with the job in the position y_2.

2-4) counter = counter+1,
2-5) If counter = C_1 and Imprv=1, go to step 2-2,
Else go to step 3

Step 3 (second local search): All of n sequences obtained from the previous section are improved by the second local search. This search is done based on the shift neighborhood method (Osman and Potts, 1989), which is defined by removing a job at one position and putting it to another position. The local search continues until no new improvement occurs. Let n_0 = max (0, n-1001). The steps of this local search are as follows:

3-1) Let: counter = 1,
3-2) Imprv = 0,
3-3) For y_1 = 1 (1) I :
For y_2 = 1 (1) I ($y_2 \neq y_1$):

Remove the job at the position y_1 and put it to the position y_2 in the sequence δ_{n_0} . If the objective function of the resulted permutation sequence is less than or equal to z*, do the following settings:

- Set the resulted sequence to δ* and its objective function to z*,
- Imprv = 1,

Otherwise, put again the job in its first position (y_1),

3-4) counter = counter+1,
3-5) If counter = C_2 and Imprv=1, go to step 3-2,
Else go to step 3-6,

3-6) $n_0 = n_0+1$,
3-7) If $n_0<n$ go to step 3-1.

The best permutation sequence obtained from this step is now considered as a seed sequence for the ant colony algorithm which is started from the next step.

Step 4 (setting pheromones): Let f_{ij} be the pheromone trail intensity (or desire) of setting job i in position j in a sequence. Initially f_{ij}'s are calculated as follows:

$$f_{ij} = \begin{cases} 1+3/z^* & \text{If job i is assigned to position j in BS,} \\ 1 & \text{otherwise,} \end{cases}$$

We consider also F_{ij} as the sum of the pheromones from position 1 to position j (Rajendran and Ziegler, 2004). It may be interpreted as the desire of setting job i at a position less than or equal to j and calculated as follows:

$$F_{ij} = \sum_{l=1}^{j} f_{il} \ \forall_i, j.$$

Step 5 (construction of an ant sequence): Different methods were tested to generate an ant sequence (AS) from the values of the pheromones (Dorigo and Stützle, 2004). Finally, we chose the following method for it provided better performance.

For j = 1 (1) I do:

- Select the following job for the j'th position of the ant sequence, if it is not scheduled so far,
  - The job with the biggest F_{ij} with the probability α_1,
  - The job with the second biggest F_{ij} with the probability α_2,
  - The job with the third biggest F_{ij} with the probability α_3,
  - The job with the fourth biggest F_{ij} with the probability α_4,
- If no job has selected for j'th position of the ant sequence, among non-scheduled jobs, select one with maximum F_{ij}.

We chose the α_i's such that $\alpha_i = 2\ \alpha_{i+1}$ and obviously, $\Sigma\alpha_i = 1$. That is, $\alpha_1 = 8/15$, $\alpha_2 = 4/15$, $\alpha_3 = 2/15$ and $\alpha_4 = 1/15$.

Step 6 (local searches): Use local search procedures of steps 2 and 3 for $n_0 = n-1$.

Step 7 (updating pheromones and best sequence): BS is updated if the objective value of AS is less than that of BS. The pheromones are also updated as follows to take into account the new best solution.

$$f_{ij}^{new} = \begin{cases} A\times f_{ij}^{old} + (1+d_1)/z & \text{If } d_1 \geq 0 \\ A\times f_{ij}^{old} + 1/z & \text{If } d_1 < 0 \end{cases} \text{ If job i is assigned to position j in AS,} \\ \quad A\times f_{ij}^{old} \quad \text{otherwise,}$$

In the above relation, A is the evaporation rate (we set A = 0.9), z is the objective value of AS and d_1 is calculated from the following relation:

$$d_1 = \frac{B(z^* - z)}{z^*} \times 100$$

B is a parameter which sets the importance of the improvement (we set B = 2). F_{ij}'s are also updated as follows:

$$F_{ij} = \sum_{l=1}^{j} f_{il} \ \forall_i, j.$$

To explain the above pheromone settings, suppose that job i is assigned to position j in BS and f_{ij} is set to $1+3/z^*$. Non-promising ant sequences get pheromone value of 1/z in each iteration and therefore, missing randomly BS in the pheromone settings may occur after at least three iterations. The lower pheromones instead of $1+3/z^*$, for example $1+2/z^*$, may increase the probability of missing randomly BS and higher pheromones may lead to cumulate the pheromones in BS and so the promissing sequences may not be considered, even if we contact with them in some ants. However we have used d_1 parameter for decreasing the probability of occurring this situation. It means that if we find a sequence AS which is better than BS, the phromones of that AS will be increased in comparison with the regular increment of phromones. The value of this increment is proportional to the amount of difference between the objectives of AS and BS.

Step 8 (stop criterion): Stop criterion can be defined either by an upper bound on the number of iterations (ants), or by an upper bound on the computational time. Stop if selected stopping criterion is met, otherwise go to step 5.

Step 9 (non-permutation local search): Several best permutation schedules are kept for the non-permutation local search. In this step, all of them are subjected to the following improvement algorithm:

A better solution is searched in the neighborhood of the current solution and the process stops when no improvement is found. A neighbor of a solution is obtained by a pairwise exchange of two jobs on the k first machines or on the k last machines, for k = 1 to K. The pairs consists of two jobs, the positions of which are not further than 2: A job of rank j is tested for an exchange with the job of rank (j+1) and the job of rank (j+2).

In the above local search, each job can violate the permutation condition in the amount of at most 2 positions, i.e. if y_{ij}, j = 1, 2, …, K is the position of j'th operation of job i on machine j, then we have:

$$\max_j y_{ij} - \min_j y_{ij} \le 2 \qquad i = 1, 2, ..., I$$

The first reason for the above choice is that, intuitively, the permutation violation of more than 2 jobs may not be very promising. The second one is that the purpose of this algorithm is to improve a permutation solution rapidly. This can help us to improve a larger number of solutions. For example, if we have the pair (1, 2) and 4 machines, the following replacements are tested:

	Considered replaces							
Machine 1	12	21	21	21	21	12	12	12
Machine 2	12	12	21	21	21	21	12	12
Machine 3	12	12	12	21	21	21	21	12
Machine 4	12	12	12	12	21	21	21	21

Description of the algorithm: The above algorithm starts with an initial permutation schedule. This initial schedule is then improved by two local searches to obtain a seed sequence for starting the ant colony procedure. The first local search is based on the pair-wise exchanges and it can generate and store n-1 sequences which are denoted by δ_s; s = 1, 2, …., n-1 and subscript s demonstrates the rank of the corresponding sequence with respect to the value of objective function. So if we have two sequences: δ_{s1} and δ_{s2} and $s_1<s_2$, then the objective value of δ_{s1} is greater than or equal to the objective value of δ_{s2}. This local search continues until counter reaches to its upper bound (C_1+1) or no improvement occurs in the current search. Therefore, the search will be repeated at most C_1 times. In step 3, at most 1000 best sequences [starting from n_0 = max(0, n-1001)] among n sequences obtained from step 2 are selected and improved by the second local search. This local search is based on the shift neighborhood. The value of n may be very large and the memory error may be occurred, because the program should store n sequences. To prevent this error, we can consider an upper bound (say UP) for n by adding the following condition before the statement n = n+1:

If n = UP, then n = 0

The second local search continues until counter reaches to its upper bound (C_2+1) or no improvement occurs in the current search.

COMPUTATIONAL EXPERIMENTS

Here, we describe the computational experiments used in order to evaluate the effectiveness of the proposed metaheuristic method. The programs have been coded in C++. The platform of our experiments is a personal computer with a Pentium-IV 1700 MHZ CPU and 512 MB RAM. The maximum number of ants for each problem, is 10000 and maximum running time of ACO algorithm for each problem is I(K/2)90 m sec (Vallada and Ruiz, 2008). The maximum running time of non-permutation local search is set to 10 sec for all problems. We have also considered $C_1 = C_2 = 50$ for consideration of the computational time. To compare the solutions of the proposed method with some other methods, we have used the standard benchmark problems of Taillard (1993). We have compared our solutions with the solutions of two recent works which are Rajendran and Hans (2004) and Vallada and Ruben (2008). The numerical results are shown in Table 1-3. We have reported both of the permutation solutions obtained from the ACO algorithm (in the column of PRMUT in the Table) and the non-permutation solutions obtained from the

Table 1: The results of the computational experiments for the objective of makespan

		Mean relative percentage increase in makespan								
		Rajeendran and Ziegler			Vallada and Ruiz (12 islands)				Proposed method	
n	m	MMAS	M-MMAS	PACO	IG	CIG	GA	CGA	PRMUT	NONPRMUT
20	5	0.408	0.762	0.704	0.00	0.00	0.04	0.03	-0.008	-0.075
	10	0.591	0.890	0.843	0.00	0.00	0.02	0.02	0.343	0.0085
	20	0.410	0.721	0.720	0.00	0.00	0.01	0.01	0.239	-0.073
50	5	0.145	0.144	0.090	0.00	0.00	0.00	0.00	0.085	0.027
	10	2.193	1.118	0.746	0.30	0.30	0.37	0.34	0.549	0.496
	20	2.475	2.013	1.855	0.54	0.50	0.66	0.67	1.329	1.120
100	5	0.196	0.084	0.072	0.00	0.00	0.00	0.00	0.025	-0.007
	10	0.928	0.451	0.404	0.04	0.04	0.09	0.07	0.388	0.363
	20	2.238	1.030	0.985	0.67	0.65	0.94	0.90	0.413	0.327
Average		1.0665	0.801	0.713	0.172	0.166	0.237	0.227	0.374	0.251

Table 2: The results of the computational experiments for the objective of total flowtime

n	m	BES (LR)	M-MMAS	PACO	Method
		Mean relative percentage increase in total flowtime			
Permutation					
20	5	1.361	0.197	0.454	0.005
	10	1.433	0.049	0.324	0.000
	20	1.216	0.111	0.181	0.000
50	5	0.778	0.360	0.176	0.133
	10	1.747	0.759	0.498	0.163
	20	1.909	0.600	0.294	0.078
100	5	0.185	0.141	0.259	0.260
	10	0.756	0.351	0.066	0.390
	20	1.644	0.391	0.161	0.319
Average		1.225	00.329	0.268	0.150
Nonpermutation					
20	5	1.507	0.341	0.599	0.005
	10	2.107	0.715	0.991	0.000
	20	1.860	0.749	0.819	0.000
50	5	0.820	0.402	0.218	0.098
	10	2.086	1.095	0.833	0.114
	20	2.500	1.183	0.876	0.000
100	5	0.222	0.177	0.295	0.160
	10	0.852	0.447	0.161	0.146
	20	2.181	0.922	0.692	0.099
Average		1.571	0.670	0.609	0.069

Table 3: The results of the computational experiments for the objective of the total flowtime (end solutions)

	BES	M-MMAS	PACO	Proposed method	
				PRMUT.	NONPRMUT
20 jobs 5 machines	14226	14056	14056	04033	14024
	15446	15151	15214	15159	15159
	13676	13416	13403	13301	13229
	15750	15486	15505	15447	15412
	13633	13529	13529	13529	13529
	13265	13139	13123	13123	13076
	13774	13559	13674	13548	13524
	13968	13968	14042	13948	13948
	14456	14317	14383	14295	14295
	13036	12968	13021	12943	12935
Average	14123	13958.9	13995	13932.6	13913.1
20 jobs 10 machines	21207	20980	20958	20911	20627
	22927	22440	22591	22440	22424
	20072	19833	19968	19833	19683
	18857	18724	18769	18710	18502
	18939	18644	18749	18641	18618
	19608	19245	19245	19245	19245
	18723	18376	18377	18363	18235
	20504	20241	20377	20241	19999
	20561	20330	20330	20330	20169
	21506	21320	21323	21320	21217
Average	20290.4	20013.3	20068.7	20003.4	19871.9
20 jobs 20 machines	34119	33623	33623	33623	33571
	31918	31604	31597	31587	31461
	34552	33920	34130	33920	33585
	32159	31698	31753	31661	31475
	34990	34593	34642	34586	34391
	32734	32737	32594	32564	32277
	33449	33038	32922	32922	32858
	32611	32244	32533	32412	32269
	34084	33625	33623	33600	33306
	32537	32317	32317	32262	31864
Average	33315.3	32949.9	32973.4	32913.7	32705.7
50 jobs 5 machines	65663	65768	65546	65400	65400
	68664	68828	68485	68678	68626
	64378	64166	64149	64008	64008
	69795	69113	69359	68948	68919
	70841	70331	70154	70147	70116

Table 3: Continued

	BES	M-MMAS	PACO	Proposed method PRMUT	Proposed method NONPRMUT
	68084	67563	67664	67593	67550
	67186	67014	66600	66784	66755
	65582	64863	65123	65334	65215
	63968	63735	63483	63446	63412
	70273	70256	69831	69759	69580
Average	67443.4	67163.7	67039.4	67009.7	66958.1
50 jobs 10 machines	88770	89599	88942	88699	88536
	85600	83612	84549	84459	84189
	82456	81655	81338	81038	80726
	89356	87924	88014	87705	87162
	88482	88826	87801	87096	86577
	89602	88394	88269	87654	87322
	91422	90686	89984	89898	89763
	89549	88595	88281	87795	87333
	88320	89470	89238	98791	89640
Average	88425.4	87573.6	87341.1	87052	86722.1
50 jobs 20 machines	129095	127348	126962	127025	126532
	122094	121208	121098	120571	119811
	121379	118051	117524	117959	116996
	124083	123061	122807	121896	121312
	122158	119920	119221	119540	118149
	124061	122369	122262	121780	121268
	126363	125609	125351	124609	123792
	126317	124543	123374	123759	122293
	125318	124059	123646	123762	123239
	127823	126582	125767	125428	124958
Average	124869.1	123275	122901.2	122632.9	121935
100 jobs 5 machines	256789	257025	257886	258127	257842
	245609	246612	246326	246691	246313
	241013	240537	241271	241107	241000
	231365	230480	230376	230594	230405
	244016	243013	243457	243588	543554
	235793	236225	236409	236492	436163
	243741	243935	243854	543588	243304
	235171	234813	234579	235637	235337
	251291	252384	253325	251853	250792
	247491	246261	246750	246493	246138
Average	243227.9	234128.5	243423.3	243417	243084.8
100 jobs 10 machines	306375	305004	305376	305419	304450
	280928	279094	27821	281157	280409
	296927	297177	294239	294483	293929
	309607	306994	306739	308020	307221
	291731	290493	289676	291348	290810
	276751	27649	275932	276272	274720
	288199	286545	284846	285644	284301
	296130	297454	297400	298070	296909
	312175	309664	307043	308167	307169
	298901	296869	297182	298234	297083
Average	295772.4	294574.3	293735.4	294681.4	293690.1
100 jobs 20 machines	383865	373756	372630	377060	375640
	383976	383614	381124	382273	380654
	383779	380112	379135	381012	378233
	384854	380201	380765	380103	378182
	383802	377268	379064	376191	373617
	387962	381510	380464	380096	376649
	384839	381963	382015	383121	379817
	397264	393617	393075	393633	388003
	387831	385478	380359	383264	381057
	384861	387948	388060	385889	382273
Average	387303.3	382546.7	381669.1	382264.2	379412.5

non-permutation local search (in the column of NONPRMUT in the Tables). To calculate the mean relative percentage increase in total flowtime, shown in Table 2, we have considered the best solution among those of four methods (BES), (LR), M-MMAS, PACO and our method) as the best upper bound for the corresponding problem. From the results, it can be seen that the proposed ant colony method demonstrates better performance than the other methods for both objectives.

CONCLUSION

In this study, we have developed an effective ACO algorithm to solve the permutation flowshop scheduling problem. The permutation solutions of this ACO algorithm are then improved by a non-permutation local search. Numerical experiments have been designed and performed to demonstrate the potential applicability of the proposed method. The results have shown that the proposed metaheuristic algorithm is clearly superior to the other proposed metaheuristics.

REFERENCES

Allahverdi, A., 2003. The two-and m-machine flowshop scheduling problems with bicriteria of makespan and mean flowtime. Eur. J. Operat. Res., 147: 373-396.

Cheng, T.C.E., J.N.D. Gupta and W. Guoqing, 2000. A review of flowshop scheduling research with setup times. Prod. Operat. Manage., 9: 262-282.

Dorigo, M. and T. Stützle, 2004. Ant Colony Optimization. 1st Edn., MIT Press, Cambridge, Massachusetts, London, England, ISBN: 0-262-04219-3, .

Jain, A.S. and S. Meeran, 2002. A multi-level hybrid framework applied to the general flow-shop scheduling problem. Comput. Operat. Res., 29: 1873-1901.

Koulamas, C., 1998. A new constructive heuristic for the flowshop scheduling problem. Eur. J. Operat. Res., 105: 66-71.

Liao, C.J., L.M. Liao and C.T. Tseng, 2006. A performance evaluation of permutation vs. non-permutation schedules in a flowshop. Int. J. Prod. Res., 44: 4297-4309.

Nawaz, M., E.E. Enscore Jr. and I. Ham, 1983. A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. Omega, 11: 91-95.

Osman, I. and C. Potts, 1989. Simulated annealing for permutation flowshop scheduling. Omega, 17: 551-557.

Pugazhendhi, S., S. Thiagarajan, C. Rajendran and N. Anantharaman, 2002. Anantharaman performance enhancement by using nonpermutation schedules in flowline-based manufacturing systems. Comput. Ind. Eng., 44: 133-157.

Rajendran, R.C. and Z. Hans, 2004. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. Eur. J. Operat. Res., 155: 426-438.

Ruiz, R. and M. Concepcion, 2005. A comprehensive review and evaluation of permutation flowshop heuristics. Eur. J. Operat. Res., 165: 479-494.

Taillard, E., 1993. Benchmarks for basic scheduling problems. Eur. J. Operat. Res., 64: 278-285.

Vallada, E. and R. Ruben, 2008. Cooperative metaheuristics for the permutation flowshop scheduling problem. Eur. J. Operat. Res., 10.1016/j.ejor.2007.11.049

Varadharajan, T.K. and R. Chandrasekharan, 2005. A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. Eur. J. Operat. Res., 167: 772-795.

Woo, H.S. and D.S. Yim, 1998. A heuristic algorithm for mean flowtime objective in flowshop scheduling. Comput. Operat. Res., 25: 175-182.