S. SOFIANOPOULOU

A queueing network application to a telecommunications distributed system

RAIRO. Recherche opérationnelle, tome 26, nº 4 (1992), p. 409-420

<http://www.numdam.org/item?id=RO_1992__26_4_409_0>

Recherche opérationnelle/Operations Research
(vol. 26, n° 4, 1992, p. 409 à 420)

A QUEUEING NETWORK APPLICATION TO A TELECOMMUNICATIONS DISTRIBUTED SYSTEM (*)

by S. SOFIANOPOULOU ([1])

Communicated by R. E. BURKARD

Abstract. — *The purpose of this paper is to present and solve a particular class of a telecommunications related Process Allocation Problem. The problem deals with the allocation of processes to a network of processors with the aim to minimize a "trade-off" objective function composed of (a) the queueing delays overhead which is formed in the underlying queueing network and (b) the communication costs incurred between processes residing on different processors. Various application constraints are also taken into account. A cost function is first constructed to reflect the queueing delay "felt" by a subscriber and a simulated annealing algorithm is then used to minimize the trade-off objective function.*

Keywords : Queueing networks; distributed processing; process allocation.

Résumé. — *L'objet de cet article est de présenter et résoudre une classe particulière de problème d'affectation continue de processus (entendu ici comme modules, programmes spéciaux, etc.) relatifs à un système de télécommunication. Le problème traite de l'affectation de processus un réseau de processeurs en vue de minimiser une fonction-objectif de substituabilité composée de : (a) les charges dues aux retards dans le réseau de files d'attente sous-jacent; (b) les coûts dûs aux communications entre les processus résidant dans les différents processeurs. Nous prenons aussi en compte diverses contraintes. Une fonction de coût est tout d'abord construite pour refléter le retard « ressenti » par un abonné, puis nous utilisons un algorithme de recuit simulé pour minimiser la fonction de substituabilité.*

Mots clés : Réseaux de files d'attente; traitement distribué; affectation de processus.

1. INTRODUCTION

Of primary importance in modern computer and communication technology is the concept of distributed systems. Distributed systems are considered to be networks of loosely coupled processors on which jobs are executed. A particular job in the system is partitioned into several small self-contained tasks which are performed by different independent modules, special pro-

(*) Received May 1990.
([1]) The Economic University of Athens, 104 34 Athens, Greece.

grams, known as processes. In a telecommunications environment, which is of interest here, the processors system controlling a telephone exchange is considered as a distributed processing network where a variety of jobs, such as fault interrupts, administrative programs, call processing, routine tests and diagnostics are performed by the processors.

As the job is progressed, every task is processed by a particular process, while several other tasks, belonging to either the same or a different job, are due to be executed by the same process. This gives rise to delays, since the chain of tasks which form a job must wait until the appropriate process becomes available and ready to execute them. The queueing delays overhead is a significant cost in the problem which needs to be minimized in order to improve the network's performance.

Another "cost" which is also associated with the distributed network is the message passing overhead related to the exchange of messages among processes residing on different processors. The problem of allocating processes to processors with the aim to minimize the message passing overhead is known as the Process Allocation Problem (PAP) [11]. In telecommunications applications, however, one is faced with the problem of allocating processes to processors, so that both the queueing delay and communication costs are sufficiently low. In addition, certain resource requirements -constraints- should be met.

The two above mentioned performance criteria are obviously conflicting. It is evident that in order to keep both costs at a reasonable (or desirable) level, some sort of trade-off between the queueing delay and the message passing cost, associated with a particular network configuration, should be adopted.

The purpose of this paper is to introduce the queueing delay overhead to the PAP as a performance criterion via a trade-off approach. The resulting minimization problem is tackled with an efficient simulated annealing algorithm. Computational results of a set of random problems which have similar characteristics to a "real-world" application in telecommunications are reported. A brief discussion of the application of queueing network theory to the PAP is also included.

2. PROBLEM STATEMENT

Consider a number N of processes not necessarily all distinct from one another. Replicate processes can be included in the problem with the restriction not to co-exist in the same processor. Let C denote the message passing matrix whose elements c_{ij}, $i=1, \ldots, N-1$, $j=i+1, \ldots, N$, represent the amount of messages exchanged between processes i and j residing on different processors (groups of processes). It is assumed that the communication cost is zero when processes i and j reside on the same processor. The sum of messages received by process i is denoted by m_i, whilst r_i is the resource requirement (in memory, occupancy etc.) of process i and R is the resource availability (in memory, occupancy etc.) of each one of the identical processors.

The minimal combined cost (of communication and queueing delay) problem is formulated as a 0-1 programming problem using the 0,1 variables X_{ij}, $i=1, \ldots, N-1$, $j=i+1, \ldots, N$. X_{ij} is 1 if processes i and j are co-located, allocated to the same processor, and 0 if they reside on different processors. The formulation of the problem is

$$\text{minimize } f = w f_c + (1-w) f_d \tag{1}$$

subject to

$$\sum_{i=1}^{k-1} r_i X_{ik} + \sum_{j=k+1}^{N} r_j X_{kj} \leqq R - r_k \text{ (memory/capacity constraints)}$$

$$k=1, \ldots, N \tag{1.1}$$

$$\left.\begin{array}{l} X_{ij}+X_{ik}-X_{jk} \leqq 1 \\ X_{ij}-X_{ik}+X_{jk} \leqq 1 \\ -X_{ij}+X_{ik}+X_{jk} \leqq 1 \end{array}\right\} \begin{array}{l} \text{(triangular constraints)} \\ i=1, \ldots, N-2, \\ j=i+1, \ldots, N-1,\ k=j+1, \ldots, N \end{array} \tag{1.2}$$

$$X_{ij} = \begin{cases} 1, & \text{if processes } i \text{ and } j \text{ are allocated to the same processor (co-located)} \\ 0, & \text{otherwise,} \quad i=1, \ldots, N-1,\ j=i+1, \ldots, N \end{cases} \tag{1.3}$$

$$X_{ij}=0, \quad \text{if } i \text{ and } j \text{ are replicate processes} \tag{1.4}$$

where f_c and f_d are the two performance criteria, *i.e.* the communication and queueing delays costs respectively, appropriately scaled to be rendered in the same order of magnitude. These two costs are discussed in detail in the next section.

w is a weighting factor ranging from 0 to 1 indicating the direction of "interest" between the two costs. At the two extreme values of w, *i.e.* at $w=0$ and $w=1$, we seek to minimize either the amount of queueing delay (which is trivial, since this will produce allocations which use as many processors as there are processes) or the inter-processor communication overhead. At intermediate values of w, *i.e.* as w gradually increases from 0 to 1, we obtain solutions with the weight moving from solutions with near-to-minimum amount of queueing delays to those with near-to-minimum amount of communication cost.

It is interesting to note that problem formulation (1) is a mathematical programming formulation which does not only produce solutions with the minimal combined cost, but it also provides automatically the number of groups (processors) to which the best allocation corresponds [12].

3. PERFORMANCE CRITERIA

The Queueing Network Model

A queueing network can be defined as a collection of service centres, *i.e.* a multiple resource system, where the customers proceed from one to another in order to fulfill their service requirements [5]. When queueing networks are related to the PAP, in any particular solution/allocation the service centres represent the processes (and hence the processors which execute them) and the customers represent the various tasks to which the call processing is divided. Each service centre has its own queue in which customers wait to receive service. There is a certain service rate and a queueing discipline associated with each one of these centres.

In the present analysis Jackson's model is employed. Jackson [7] introduced the analysis of open queueing networks. A network is considered open if jobs are permitted to enter or leave the system at any time. He proved that in equilibrium each service centre behaves as if it were an individual M/M/1 queueing system, *i.e.* a system with Poisson arrival rates, exponential service rates, FCFS queue discipline and a single server. This model has the property of having a stationary solution in "product form" for the joint probabilities of the lengths of the queues, *i.e.*

$$P(n_1, n_2, \ldots, n_Q) = p_1(n_1)\, p_2(n_2) \ldots p_Q(n_Q)$$

where $p_i(n_i)$ represents the marginal probability that there are n_i customers waiting or in service at centre i. This result, known as Jackson's decomposition theorem, proves that the joint distribution is decomposable into the product of Q marginal distributions. Thus an open queueing network of service centres can be decomposed into several individual centres, and its performance measures can be calculated using the traditional queueing theory for M/M/1 queueing systems (e.g. *see* [4]).

Application to PAP

In calculating the queueing delays in the PAP Jackson's model has been adopted. It is assumed that the flows of messages into and out of each processor are equal. Although this is not always true in a real system (some processes expire at certain stages and hence do not forward any more messages) it does not seriously affect the results [9].

Each service centre is considered as an independent M/M/1 queueing system with one server serving at a time. Although each service centre (processor) has several servers (processes) allocated to it, it is still considered as an M/M/1 queueing system since only one process can be executed at a time by each processor. The service rate μ_i of each process i run on any processor depends on the load of the processor on which it resides (group of co-located processes) and is given by

$$\mu_i = \lambda_i / \rho_i \tag{2}$$

where λ_i is the sum of messages received by process i as well as any other process co-located to it, and ρ_i is the sum of their occupancies, *i.e.* proportion of time for which they are running. Thus

$$\lambda_i = m_i + \sum_{j=1}^{i-1} m_j X_{ji} + \sum_{k=i+1}^{N} m_k X_{ik} \tag{3}$$

$$\rho_i = r_i + \sum_{j=1}^{i-1} r_j X_{ji} + \sum_{k=i+1}^{N} r_k X_{ik} \tag{4}$$

Then the delay "seen" by a task requiring service by process i is given by

$$d_i = 1/(\mu_i - \lambda_i) \tag{5}$$

This delay includes both service and queueing time and can be written as

$$d_i = \rho_i / \lambda_i (1 - \rho_i) \tag{6}$$

The overall delay of a job submitted to a network of processors consists of the sum of all delays experienced by its chain of tasks as the latter are sequentially executed by different processes allocated to the various processors. Our objective here, however, is not to minimize this overall delay but only the delay which is "felt" by the network user, in our case the telephone system user-subscriber. During the call setup procedure (job), for instance, the subscriber "notices" the delay associated with the process of analyzing the first few (routing) digits dialed, but he does not "notice" any delay involved with the traffic-recording process.

What is of importance therefore, is the sum of the delays on certain time-critical paths of the job. These critical paths, however, are not all equally important because some of them are used by the call setup procedure more frequently than others. Since it is most unusual to minimize the delays on all critical paths simultaneously, what we are interested in, is to minimize a single "effective" delay composed of the weighted delays on each time-critical path, the weighting factors being proportional to the percentage of jobs, call setup procedures, using each path.

Assume for example that we have s time-critical paths denoted by P_k, $k=1, \ldots, s$, each composed by a sequence of a number of processes $l(k)$ $k_1, k_2, \ldots, k_q, \ldots, k_{l(k)}$, where k_q could be any number between 1 and N. Let dP_1, $dP_2, \ldots, dP_s$ be the delays on each path respectively and let a_k, $k=1, \ldots, s$ be the percentage (frequency) of the call setups using time-critical path k ($\Sigma a_k=1$). Then the "effective" delay sought to be minimized would be

$$f_d=a_1\, dP_1+a_2\, P d_2+\ldots+a_k\, dP_k+\ldots+a_s\, dP_s \tag{7}$$

where $dP_k=d_{k_1}+d_{k_2}+\ldots+d_{k_q}+\ldots+d_{k_{l(k)}}$

Equation (7) becomes

$$f_d=\sum_{k=1}^{s} a_k \sum_{q=1}^{l(k)} \frac{\rho_{k_q}}{\lambda_{k_q}(1-\rho_{k_q})}$$

or in terms of X_{ij}

$$f_d=\sum_{k=1}^{s} a_k \times \sum_{q=1}^{l(k)} \frac{r_{k_q}+\sum_{j=1}^{k_q-1} r_j X_{j k_q}+\sum_{i=k_q+1}^{N} r_i X_{k_q i}}{\left(m_{k_q}+\sum_{j=1}^{k_q-1} m_j X_{j k_q}+\sum_{i=k_q+1}^{N} m_i X_{k_q i}\right) \times\left(1-r_{k_q}-\sum_{j=1}^{k_q-1} r_j X_{j k_q}-\sum_{i=k_q+1}^{N} r_i X_{k_q i}\right)} \tag{8}$$

The fractional type of objective function (8) is typical of queueing delays costs [2].

Communication cost

The other aspect of the PAP, which is mentioned in the first section, is the message passing cost associated with the volume of messages exchanged between processes assigned to different processors. Processes that communicate with one another incur a communication cost c_{ij}, $i \neq j$, and the objective would be to minimize the message passing cost between processes residing in different processors, *i. e.*

$$f_c = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij}(1 - X_{ij}) \tag{9}$$

The sum of equations (8) and (9) form the optimization criterion of the PAP considered in this work.

4. THE ALGORITHM

In formulation (1) the communication cost part of the objective function as well as the constraints are linear in the X_{ij} variables. On the other hand, the queueing delays overhead part of the objective function is fractional. Even if only the communication cost is taken into account, the PAP is more general than the NP-hard graph partitioning problem. It seems therefore appropriate to tackle the PAP using a heuristic algorithm.

The heuristic algorithm employed to solve problem (1) is simulated annealing [10], [14], [15]. This algorithm is a heuristic optimization method based on iterative improvement. The method has been successfully applied to combinatorial problems in computer systems design [8], to the solution of the travelling salesman [6], [8], and the quadratic assignment problems [3], and to the minimization of message passing cost in the PAP [13].

The basic idea is to generate random displacements from any current feasible solution, and accept as new current solution not only solutions which improve the objective function, but also some, which do not improve it; the latter ones are accepted with probability $\exp(-\Delta f/T)$, depending on the amount of deterioration Δf of the objective function and a tunable parameter T (the temperature).

The two ingredients required in the implementation of simulated annealing are:

1. A perturbation scheme for generating random displacements, *i. e.* a neighborhood generation scheme.

2. An annealing schedule. This consists of determining

(*a*) A sequence of the control parameter temperature T; *i.e.* initial and final values and a rate of decrease.

(*b*) The number of solutions L_t attempted at each temperature T_t as the temperature decreases.

The algorithm iterates until the stopping temperature value is reached or no feasible solution has been found during L_t attempts.

In the present implementation, the neighborhood generation scheme consists of randomly selecting two processes p and q which are not interconnected ($X_{pq} \neq 0$), and making the corresponding variable X_{pq} equal to 1. Of course this move has some implications which involve disconnecting process p from all processes to which it was previously connected and connecting it to all processes to which q is already connected [triangular constraints (1.2)].

The initial (and final) value of T is determined using a small number of pilot runs before the actual annealing process begins, at an appropriately high (low) value so that almost all candidate solutions, which deteriorate the objective function, are accepted as current solution with a probability of 0.95 or more (0.05 or less). The cooling schedule, *i.e.* the rate of decrease of temperature T, is very crucial for the successful application of the algorithm. If the rate of temperature decrease is too high then the algorithm leads to local optimum solutions, while if it is too low CPU time is wasted. In this work the cooling schedule suggested in [1] is adopted where T_t is updated by

$$T_{t+1} = \frac{T_t}{1 + (T_t \ln(1+\delta)/3\,\sigma_t)}$$

where $\delta = 0.1$ and σ_t is the standard deviation of the objective function values of the solutions examined at T_t.

The number of solutions L_t attempted at each temperature T_t is set to $N(N-1)/2$. This parameter setting is adopted taking into account that all possible moves that can lead a current solution to a neighboring one (*i.e.* making a variable X_{pq}, which is currently set to 0, equal to 1) are at most $N(N-1)/2$ (actually they are much fewer since capacity and triangular constraints should be satisfied).

Having defined the above ingredients of the simulated annealing and an initial feasible solution to the PAP, one can apply the algorithm. Obviously, the number of iterations to be performed depends on the size of the problem. For the sake of completeness it should be mentioned that a starting feasible solution to the PAP is formed by randomly choosing processes and grouping them together until a capacity constraint is violated. If this is the case, the next randomly chosen process starts forming a second group and this procedure is continued until all processes are grouped.

5. COMPUTATIONAL RESULTS

Computations were carried out on a VAX 8810 computer. The 24 randomly generated data sets used to test the method are very similar in structure to a 12-processes sample instance of the problem which was provided to us by a telecommunications laboratory. Data input to the model includes the number of processes to be allocated (N), the occupancy, code- and data-storage requirements (r_i) of each process i, and the resource availabilities on each processor (R), the matrix $C=[c_{ij}]$ of messages exchanged between processes and the time-critical paths P_k, with their usage frequencies a_k. The random data sets include 12 problems with 12 processes each, while replicate processes (identical to one another which are not allowed to run on the same processor) are present in the model. Different groups of the above test problems were generated, having 5 sets with three (12-3GRP1 to 12-3GRP5), 2 sets with one (12-1GRP1 and 12-1GRP2), 5 sets with zero (12-OGRP1 to 12-OGRP5) group(s) of three replicate processes each. Larger test problems, compared to our original instance of the problem, including 5 problems with 15 processes each (15-OGRP1 to 15-OGRP5), 5 problems with 20 processes each (20-OGRP1 to 20-OGRP5) and finally 2 problems, each with 25 processes (25-OGRP1 and 25-OGRP2) were also tried. For each one of the identical processors used in the model the occupancy figure did not exceed 70%, while the amount of code- and data-storage did not exceed 300 and 350 units respectively. The time-critical paths used to evaluate the queueing delays in each test problem were constructed by randomly choosing sequences of processes.

Tables 1 and 2 present the results obtained with the weight w of the combined objective function equal to 0.8 and 0.2 respectively. The two tables are divided in three parts. The annealing heuristic was run with (*a*) $L_t=N(N-1)/2$, (*b*) $L_t=N(N-1)/4$ and (*c*) $L_t=N(N-1)/8$ and the message passing, the amount of queueing delay and the CPU seconds consumed in each case are recorded in each one of the three parts of the tables. It is reasonable to assume that the annealing schedule with $L_t=N(N-1)/2$ would provide solutions closer to the optimum, which is of course not known, since the algorithm spends more time in each temperature attempting more solutions. Comparing now the results for $L_t=N(N-1)/2$ and $L_t=N(N-1)/4$

TABLE 1

$w=0$ 8 DATA SET	$L_t=N(N-1)/2$			$L_t=N(N-1)/4$			$L_t=N(N-1)/8$		
	Mes Pas (units/s)	DELAY (s × 100)	CPU (s)	Mes Pas (units/s)	DELAY (s × 100)	CPU (s)	Mes Pas (units/s)	DELAY (s × 100)	CPU (s)
12-3GRP1	395	2 78	39	395	2 78	19	395	2 78	7
12-3GRP2	594	4 54	49	594	4 54	23	594	4 54	9
12-3GRP3	471	3 95	33	471	3 95	9	471	3 95	12
12-3GRP4	468	4 00	19	468	4 01	8	468	4 02	4
12-3GRP5	636	3 89	122	636	3 91	7	636	3 89	5
12-1GRP1	490	2 64	52	490	2 64	9	494	2 60	3
12-1GRP2	463	2 39	17	463	2 40	8	463	2 39	3
12-0GRP1	459	3 77	32	459	3 77	4	459	3 77	3
12-0GRP2	499	3 87	35	499	3 87	9	499	3 87	6
12-0GRP3	533	3 03	28	537	3 06	3	533	3 03	5
12-0GRP4	514	3 91	36	514	3 91	10	501	4 44	3
12-0GRP5	546	2 99	18	546	2 99	14	538	3 21	5
15-0GRP1	685	5 96	77	685	5 96	51	690	5 99	12
15-0GRP2	808	4 90	198	808	4 90	35	808	4 90	16
15-0GRP3	707	3 50	49	707	3 50	21	707	3 50	12
15-0GRP4	827	3 92	176	827	3 92	59	820	4 22	10
15-0GRP5	787	5 47	64	787	5 47	80	787	5 47	39
20-0GRP1	1 561	6 44	508	1 561	6 44	313	1 561	6 44	95
20-0GRP2	1 468	7 11	1 140	1 468	7 11	407	1 468	7 11	96
20-0GRP3	1 539	8 26	595	1 539	8 26	286	1 539	8 26	101
20-0GRP4	1 425	5 29	823	1 425	5 29	438	1 425	5 29	90
20-0GRP5	2 284	7 67	1 783	2 270	8 08	1 454	2 284	7 75	169
25-0GRP1	2 422	7 61	2 046	2 422	7 61	1 178	2 422	7 61	380
25-0GRP2	2 297	8 09	2 968	2 297	8 09	945	2 312	7 81	773

it can be easily seen that in almost all cases the $L_t=N(N-1)/4$ schedule performs equally well, in terms of solutions obtained, while the CPU time consumed is much less than half of the time of the $L_t=N(N-1)/2$ schedule. Obviously, "reasonably good" and CPU time-efficient solutions of the PAP are obtained using the $L_t=N(N-1)/8$ schedule.

In tables 3 and 4 the message passing and queueing delay costs for two test problems with w varying from 0.2 to 1 are reported. The corresponding number of processors used is also included. As expected, as w increases, the number of processors used and the amount of message passing decreases, while the amount of queueing delay increases.

TABLE 2

$w=0\ 2$ DATA SET	$L_t=N(N-1)/2$			$L_t=N(N-1)/4$			$L_t=N(N-1)/8$		
	Mes Pas (units/s)	DELAY (s × 100)	CPU (s)	Mes Pas (units/s)	DELAY (s × 100)	CPU (s)	Mes Pas (units/s)	DELAY (s × 100)	CPU (s)
12-3GRP1	431	2 12	99	431	2 12	17	431	2 12	6
12-3GRP2	656	3 36	185	656	3 36	17	656	3 36	7
12-3GRP3	509	3 13	62	509	3 13	10	509	3 13	8
12-3GRP4	511	3 05	90	511	3 05	18	511	3 05	5
12-3GRP5	713	3 01	64	713	3 01	22	713	3 01	7
12-1GRP1	569	1 84	31	552	1 92	5	555	1 90	4
12-1GRP2	565	1 48	72	551	1 51	17	569	1 48	7
12-0GRP1	523	2 52	32	523	2 52	9	520	2 59	3
12-0GRP2	576	2 85	15	576	2 65	8	576	2 65	5
12-0GRP3	619	2 51	59	619	2 51	19	619	2 51	5
12-0GRP4	599	2 72	46	599	2 72	17	590	2 76	4
12-0GRP5	612	2 06	56	612	2 06	9	612	2 06	5
15-0GRP1	771	4 38	243	771	4 38	49	771	4 38	20
15-0GRP2	853	4 20	2 676	853	4 20	259	853	4 20	23
15-0GRP3	762	3 05	98	762	3 05	89	762	3 05	14
15-0GRP4	849	3 65	93	849	3 65	94	858	3 65	13
15-0GRP5	840	4 60	370	840	4 60	140	825	4 65	13
20-0GRP1	1 626	5 40	1 496	1 625	5 40	2 324	1 625	5 40	100
20-0GRP2	1 515	6 16	4 223	1 515	6 16	182	1 531	6 14	55
20-0GRP3	1 620	7 08	3 777	1 620	7 08	270	1 620	7 08	108
20-0GRP4	1 500	4 56	887	1 500	4 56	173	1 507	4 56	37
20-0GRP5	2 443	6 04	5 532	2 443	6 04	653	2 457	6 08	119
25-0GRP1	2 490	6 75	1 841	2 490	6 75	1 390	2 491	6 75	160
25-0GRP2	2 402	6 72	3 770	2 402	6 72	1 066	2 417	6 72	237

TABLE 3

w	$L_t=N(N-1)/2$		
	Mes Pas (units/s)	DELAY (s × 100)	N° Proc
0 2	565	1 48	9
0 4	540	1 54	7
0 6	529	1 60	7
0 8	463	2 39	4
1 0	429	4 33	4

DATA SET 12-1GRP2

TABLE 4

w	$L_t=N(N-1)/2$		
	Mes Pas (units/s)	DELAY (s × 100)	N° Proc
0 2	1 620	7 08	16
0 4	1 603	7 15	15
0 6	1 582	7 35	14
0 8	1 539	8 26	13
1 0	1 491	12 56	10

DATA SET 20-0GRP3

6. CONCLUSION

In this paper the Process Allocation Problem is examined with the aim to minimize a combined objective function composed of queueing delay and communication costs and at the same time satisfy memory and occupancy constraints imposed on the network of identical processors. The construction of the queueing network, through the presentation of the congestion components in a telecommunications environment, and its solution, using Jackson's model, are discussed. A simulated annealing algorithm was employed to tackle the problem. Computational results with a set of test problems, which have similar structure to a real world telecommunications problem, are reported indicating that the algorithm performs very well within reasonable amount of CPU time.

REFERENCES

1. E. Aarts and J. Korst, Simulated Annealing and Boltzmann Machines, *J. Wiley*, Chichester, 1990.
2. S. R. Agnihothri, S. Narasimhan and H. Pirkul, An Assignment Problem with Queueing Time Cost, *Naval Res. Logist. Quart.*, 1990, *37*, p. 231-244.
3. R. E. Burkard and F. Rendl, A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems, *E. J. Oper. Res.*, 1984, *17*, p. 169-174.
4. R. B. Cooper, Introduction to Queueing Theory, *Elsevier North Holland*, New York, 1981.
5. E. Gelenbe and G. Pujolle. Introduction to Queueing Networks, *J. Wiley*, Chichester, 1987.
6. B. L. Golden and C. C. Skiscim, Using Simulated Annealing to Solve Routing and Location Problems, *Naval Res. Logist. Quart.*, 1986, *33*, p. 261-279.
7. J. R. Jackson, Networks of Waiting Queues, *Oper. Res.*, 1957, *5*, p. 518-521.
8. F. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi, Optimization by Simulated Annealing, *Science*, 1983, *220*, p. 671-680.
9. N. W. Macfadyen, *Private communication*, 1984.
10. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth and A. H. Teller, Equation of State Calculation by Fast Computing Machines, *J. Chem. Phys.*, 1953, *21*, p. 1087-1092.
11. T. Muntean and El-G. Talbi, Methodes de placement statique des processus sur architectures parallèles, *Techniques Sci. Inform.*, 1991, *10*, p. 355-373.
12. S. Sofianopoulou, Optimum Allocation of Processes in a Distributed Processing Environment: A Process-to-Process Approach, *J. Oper. Res. Soc.*, 1990, *41*, p. 329-337.
13. S. Sofianopoulou, Simulated Annealing Applied to the Process Allocation Problem, *E. J. Oper. Res.*, 1992, *60*, p. 327-334.
14. P. J. M. van Laarhoven, Theoretical and Computational Aspects of Simulated Annealing, *Centre for Mathematics and Computer Science*, CWI Tract 51, Amsterdam, 1988.
15. P. J. M. van Laarhoven, and E. H. L. Aarts, Simulated Annealing Theory and Applications, *D. Reidel Publ. Comp.*, Dordrecht, Holland, 1987.