A Self-adaptive Weighted Sum Technique for the Joint Optimization of Performance and Power Consumption in Data Centers

Samee Ullah Khan
Department of Electrical and Computer Engineering
North Dakota State University, Fargo, ND 58108
samee.khan@ndsu.edu

Abstract

With data centers, end-users can realize the pervasiveness of services that will be one day the cornerstone of our lives. However, data centers are often classified as computing systems that consume the most amounts of power. To circumvent such a problem, we propose a self-adaptive weighted sum methodology that jointly optimizes the performance and power consumption of any given data center. Compared to traditional methodologies for multi-objective optimization problems, the proposed self-adaptive weighted sum technique does not rely on a systematical change of weights during the optimization procedure. The proposed technique is compared with the greedy and LR heuristics for large-scale problems, and the optimal solution for small-scale problems implemented in LINDO. the experimental results revealed that the proposed self-adaptive weighted sum technique outperforms both of the heuristics and projects a competitive performance compared to the optimal solution.

1 Introduction

Power consumption is one of the most critical design criteria of the modern-day computing system. Because data centers are a collection of multiple computing systems, power consumption is even more critical of an issue. System performance is affected by supply voltage scaling because circuit delay and maximum system clock frequency depend on the supply voltage. Utilizing the available dynamic voltage scaling modules, the supply voltage to a computing system can be altered so that the computing system consumes lesser power. However, to make digital circuits that make up a processing element work correctly, the frequency of the clock must also be altered. This results in an altered performance. As a consequence, the processing element operates on a slower speed, which in turn results in poor system performance. That is to say, if power consumption is reduced for a system, then the performance must degrade. However, in data centers the optimization of power consumption and performance is considered to be equally important. Therefore, we must concurrently optimize power consumption and performance as a multi-objective optimization problem. A traditional method for multi-objective optimization is the weighted sum technique that seeks Pareto optimal solutions one by one by systematically changing the weights between the objective functions. Research has shown that this method often produces poorly distributed solutions along a Pareto front [6]. Moreover the weighted sum methodology cannot find Pareto optimal solutions in convex regions [5]. For this purpose, we propose a self-adaptive weighted sum technique that can circumvent the above mentioned problems.

The proposed self-adaptive weighted sum methodology effectively explores the search regions by changing the weights adaptively compared to the traditional multi-objective optimization methods. As a consequence, the proposed method produces good quality solutions. Moreover, the proposed method finds Pareto optimal solutions in non-convex regions. Furthermore, non-Pareto optimal solutions are negated. The proposed methodology is successfully applied to solve the data center multi-objective power consumption and performance optimization problem. The proposed technique is compared with the greedy and LR heuristics for large-scale problem sizes. The proposed technique also is compared with the optimal solution implemented in LINDO for small-scale problem sizes. The experimental results reveal the competitive performance of the proposed methodology compared to other techniques.

The rest of the paper is organized as follows. In Section 2, we formulate the optimization problem for data centers. Section 3 details the proposed self-adaptive weighted sum methodology. In Section 4, the proposed technique is experimentally compared the greedy and LR heuristics, and the optimal solution implemented in LINDO. The related work and concluding remarks are provided in Sections 5 and 6, respectively.

2 Data Center System Model and Optimization Problem Description

2.1 The System Model

Consider a data center comprising of a set of machines, $M = \{m_1, m_2, \cdots, m_m\}$. Assume that each machine is equipped with a DVS module and is characterized by:

1. The frequency of the CPU, f_j, given in cycles per unit time. With the help of a DVS, f_j can vary from f_j^{min} to f_j^{max}, where $0 < f_j^{min} < f_j^{max}$. From frequency, it is easy to obtain the speed of the CPU S_j that is approximately proportional to the frequency of the machine [12].
2. The specific machine architecture, $A(m_j)$. The architecture would include the type of CPU, bus types, and speeds in GHz, I/O, and memory in bytes.

Consider a metatask, $T = \{t_1, t_2, \cdots, t_n\}$. Each task is characterized by:

1. The computational cycles c_i that it needs to complete. The assumption here is that the c_i is known *a priori*.
2. The specific machine architecture $A(t_i)$ that it needs to complete its execution.
3. The deadline, d_i, before it has to complete its execution. Moreover, we also assume that the metatask, T, also has a deadline D that is met if and only if the deadlines of all its tasks are met.

The number of computational cycles required by t_i to execute on m_j is assumed to be a finite positive number, denoted by c_{ij}. The execution time of t_i under a constant speed S_{ij}, given in cycles per second is $\frac{t_{ij}}{c_{ij}} = S_{ij}$. For the associated data and instructions of a task, we assume that the processor always retrieves it from the level-1 (primary) data cache. A task, t_i, when executed on machine m_j draws, p_{ij} amount of instantaneous power. Lowering the instantaneous power will lower the CPU frequency and consequently will decrease the speed of the CPU and hence cause t_i to possibly miss its deadline.

The architectural requirements of each task are recorded as a tuple with each element bearing a specific requirement. We assume that the mapping of architectural requirements is a Boolean operation. That is, the architectural mapping is only fulfilled when all of the architectural constraints are satisfied.

2.2 Problem Formulating

Find the task to machine mapping, where the cumulative instantaneous power consumed by the data center, M and the makespan of the metatask, T, is minimized.

Mathematically, we can say

$$\text{minimize} \left(\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} x_{ij} \text{ and } \max_{j} \sum_{i=1}^{n} t_{ij} x_{ij} \right) \tag{1}$$

$$\text{subject to } x_{ij} \in \{0, 1\}, \tag{2}$$

$$t_i \rightarrow m_j;\ \text{if} A(t_i) = A(m_j) \text{ then } x_{ij} = 1, \tag{3}$$

$$t_{ij} x_{ij} \leq d_i | x_{ij} = 1, \tag{4}$$

$$(t_{ij} x_{ij} \leq d_i) \in \{0, 1\}, \tag{5}$$

$$\prod_{i=1}^{n} (t_{ij} x_{ij} \leq d_i) = 1 | x_{ij} = 1. \tag{6}$$

Constraint (2) is the mapping constraint. When $x_{ij} = 1$, a task, t_i, is mapped to machine, m_j, and $x_{ij} = 0$ otherwise. Constraint (3) elaborates on this mapping in conjunction to the architectural requirements, and it states that a mapping can only exists if the architecture is mapped. Constraint (4) relates to the fulfillment of the deadline of each task, and constraint (5) tells us about the Boolean relationship between the deadline and the actual time of execution of the tasks. Constraint (6) relates to the deadline constraints of the metatask that will hold if all of the deadlines of the tasks, d_i, are satisfied.

3 The Multi-objective Self-adaptive Weighted Sum Technique

The weighted sum is a traditional technique to achieve optimization for multi-objective problems [6]. However, the traditional weighted sum technique has the following two major drawbacks.

1. The even distribution of weights towards the objective functions do not warrant an even distribution of solutions for each of the objective function.
2. The traditional approach cannot find non-convex solutions. This is often the reason that solutions of multi-objective problems are complex, time consuming, and non-convergent.

The proposed self-adaptive weighted sum technique circumvents the above mentioned problems and can effectively solve multi-objective optimization problems that have non-convex regions for Pareto front. The data center optimization problem described in the previous section is of that nature; hence it can be very effectively solved with the proposed technique. The proposed technique finds its roots in previous works, such as [5] and [6], and is adapted to truly reflect an applicable solution to the joint power and makespan optimization problem of a data center. (Due to the lack of space, we assume that the reader has basic knowledge of multi-objective optimization, and in particular the weighted sum technique. If the reader wishes to refresh knowledge about the weighted sum technique for multi-objective problems, then we strongly recommend reading articles [5] and [6].) Below we detail the steps of the procedure.

Let x^i represent the solution to an objective function ζ^i, and let x^{i*} denote the optimal solution of ζ^i. Moreover, let ζ^u represent the utopia (saddle) point of any number (two in our case) of objective functions.

Step 1: Normalize objective functions as: $\overline{\zeta}^i = \zeta^i - \zeta^u$.

Step 2: Utilizing minute divisions, η_0, perform multi-objective optimization using the standard weighted sum approach (an example of which can be found in [6]). The weights are adjusted relative to η_0, i.e., the step size is determined to be: $\Delta\xi = \frac{1}{\eta_0}$.

Step 3: Once a set of solutions is obtained from Step **2**, the lengths of segments between all of the "neighboring" solutions is computed. This is done to remove any duplicity of overlapping solutions that is the norm of the standard weighted sum technique.

Step 4: This step determines the number of refinements per region. The refinement is obviously directly related to the length of the segment. That is, the longer the segment, the more of refinement is needed. There is no golden rule to predetermine the number of refinements; however, decreasing the step size, $\Delta\xi$, in Step **2** can help in achieving higher resolution of the segmented regions. The refinement numerology that we used took into account the relative correlation of the average length to the length of the segment under question. In other words, we normalized the length of all of the segments relative to the average length of the segments.

Step 5: Once the refinement of Step **4** is achieved, we must in each of the feasible regions (identified through Steps **1**–**4**); (a) force the inequality constraints, and (b) sub-optimize using the standard weighted sum technique. This can be achieved by utilizing the step size, $\Delta\xi$, in Step **2**, and the distance between the optimal solution, x^{i*}, which may be obtained by solving the singleton optimization, and the achievable solution, x^i, when inequalities are forced. The distance, δ^i is understood to be equal to $x^i - x^{i*}$. Hence, we can restate the optimization problem in the following generic form:

$$\min\left(\xi \times \zeta^{\text{power}} + (1-\xi) \times \zeta^{\text{makespan}}\right) \tag{7}$$

$$\text{subject to } x^{1*} \le x^1 + \delta^1, \tag{8}$$

$$x^{2*} \le x^2 + \delta^2. \tag{9}$$

Step 6: Similar to Step **4** determine overlapping solutions. If no overlap is found, then output the solution. If an overlap is identified, then refine the solution to the multi-objective (in our case bi-objective) optimization problem by adjusting the lengths of the segments by revisiting Step **4**.

The proposed self-adaptive weighted sum technique ensures that the segments that do not have converged optimum solutions, are removed from the solution space for further refinement. This is the key property to identify accurate, feasible, and effective solutions to otherwise cumbersome multi-objective optimization problems.

4 Simulations, Results, and Discussion

We set forth two major goals for our simulation study: (a) To measure and compare the performance of the proposed technique against the optimal solution, greedy heuristic [12], and linear relaxation (LR) heuristic [12]. (b) To measure the impact of system parameter variations. We choose to compare the proposed technique against the above mentioned two heuristics because they have shown to perform extremely well compared to several other heuristics [12]. Due to space restrictions, we could not include the finer details of the greedy and LR heuristics. However, we strongly encourage the readers to review the referenced articles.

Based on the size of the problems, the simulations were divided in two parts. For small-size problems, we used an Integer Linear Programming toolkit called LINDO [10]. LINDO is useful to obtain optimal solutions, provided that the problem size is relatively small. Hence, for small problem sizes, the relative performance of the proposed technique, greedy, and LR techniques is compared against the LINDO implementation. For large-size problems, it is impractical to compute the optimal solution. Hence, we consider comparisons only against the greedy and LR heuristics. For the workload, we acquired the data from the US Air Force Satellite Control Network (AFSCN). The data is publicly available over the Internet at [11]. The AFSCN is currently responsible for coordinating communications between civilian and military organization and more than 100 USAF managed satellites. Table 1 summarizes the characteristics of the data.

There are two types of task requests that can be distinguished: (a) low-altitude and (b) high-altitude orbits. The low-altitude tasks specify requests for low-altitude satellites; such requests tend to be very short (on the average they are 5-10 minutes in duration) and

Table 1: Problem Characteristics

Date	No. of Requests	High	Low
10/12/92	322	169	153
10/13/92	302	165	137
10/14/92	311	165	146
10/15/92	318	176	142
10/16/92	305	163	142
10/17/92	299	155	144
10/18/92	297	155	142
03/07/02	483	258	225
03/20/02	457	263	194
03/26/03	426	243	183
04/02/03	431	246	185
05/02/03	419	241	178

have a tight visibility window (simply because the relative velocity of low-altitude satellites is very high compared to high-altitude satellites). High-altitude tasks specify requests for high-altitude satellites; the durations for these requests are more varied and usually longer, with large visibility windows. A problem instance of AFSCN consists of n task requests. Each task request $t_i, 1 \leq i \leq n$, specifies a required processing duration (i.e., the time to complete a task) and the associated data file (i.e., the amount of memory required to process a task). Each task request also specifies a number of pairs of the form (t_i, m_j), each identifying a particular alternative resource (i.e., an antenna or, in our context, a machine m_j) and time window, i.e., the deadline for a task d_i. The deadlines for low-altitude tasks were relatively tighter than high altitude tasks. The processing duration of the task is the same for all possible alternative resources, and it needs to be mapped to a resource and completed within the time window.

Finally, to model the DVS modules, we make the generic assumption that the task processing times were given when machines were running at full instantaneous power (or at a level of DVS equal to 100 percent). Because the task processing time, as given in the AFSCN data, is uniform across all machines, for this problem instance, we assume that the machines have a clock speed of two GHz. We also assume that a potential difference of one mV across a CMOS circuit generates a frequency of one MHz. Altering any of these assumptions will be a trivial task and will have no significant impact on the simulation results. For this study, we keep the architectural affinity requirements confined to memory. Adding other requirements, such as I/O and processor type, will bear no affect on our simulation setup. In all of the simulation setups, the storage capacity of the machines was set proportional to the total size of data items (TS). The capacity of a machine was generated using a uniform distribution from $(0.5 \times TS)$ and $(1.5 \times TS)$. For small-size problems, the number of machines was fixed at five, while the number of tasks varied from five to 50. The tasks were chosen on random from each of the twelve days of the AFSCN workload and the simulation plot is an average over the twelve runs. The number of DVS levels per machine was set to four. For large-size problems, the number of machines was fixed at sixteen, while the number of tasks varied from 322 to 4,370. The bound of the number of tasks reflect the sequential aggregation of the tasks in the AFSN workload, i.e., for the first set, we use the data from 10/12/92 having 322 tasks, for the second set, we use the data as the aggregate of the data of 10/12/92 (322 tasks) and 10/13/92 (302 tasks), giving us a total of 624 tasks, and so on. The number of DVS levels per m_j was set to sixteen.

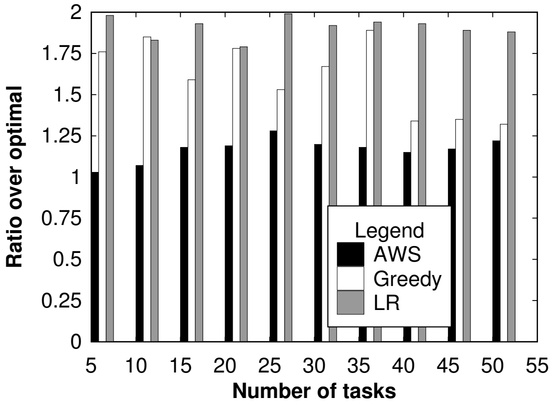

Figure 1: Makespan ratio over the optimal.

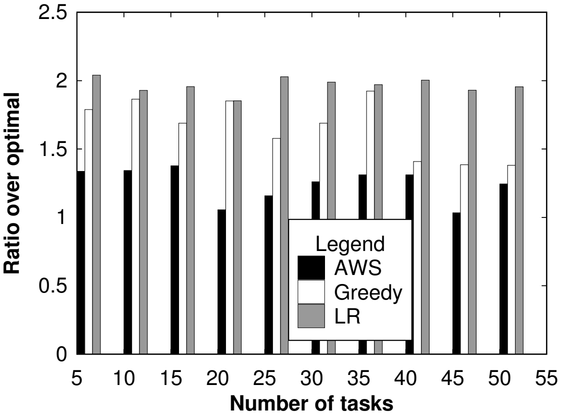

Figure 2: Energy consumption ratio over the optimal.

The simulation results for small size problems with are reported in Figs. 1 and 2. These figures show the ratio of the makespan obtained from the three techniques and the optimal. The plot in Fig. 1 shows that the proposed technique (acronym AWS) performs extremely well by achieving a makespan within ten per-

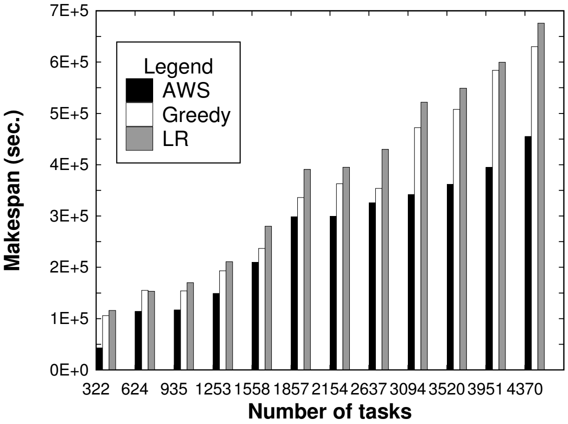

Figure 3: Makespan.

cent of the optimal solution. Next, we compare the overall energy consumption that is calculated as the time interval a task takes to complete on a given machine multiplied by the current instantaneous power of the given machine. (Measuring energy instead of power is meaningful because energy can directly be translated into monetary valuation.) From Fig. 2 we can see that the proposed AWS technique outperforms the other techniques in terms of energy savings compared to the optimal allocation within a range of fifteen percent.

Next, we record the performance of the techniques on large-scale problem instances. Fig. 3 compares the makespan identified by the AWS, greedy, and LR heuristics. The results indicate that proposed technique outperforms the greedy and LR heuristics in identifying a smaller makespan. We can observe that the AWS technique identifies a makespan that is 21.07 percent smaller than greedy and 26.82 percent smaller than the LR heuristic. Second, we compare the energy consumption of the three techniques. Fig. 4 shows the relative performance of the techniques. The proposed AWS heuristic again outperforms the other heuristics by consuming lesser energy when executing the tasks. We observe that the AWS heuristic saves on average 20.63 percent of energy than the greedy and 45.36 percent of energy than the LR heuristic.

Finally, we must analyze the runtime for both small and large problem sizes. For completion, the runtime of the optimal for small problem size is presented for comparisons. The results are depicted in Figs. 5 and 6. The AWS technique terminates many orders faster than the optimal, and the greedy and LR heuristics. This suggests that the AWS technique not only terminates faster but also delivers a superior solution quality than the compared techniques.

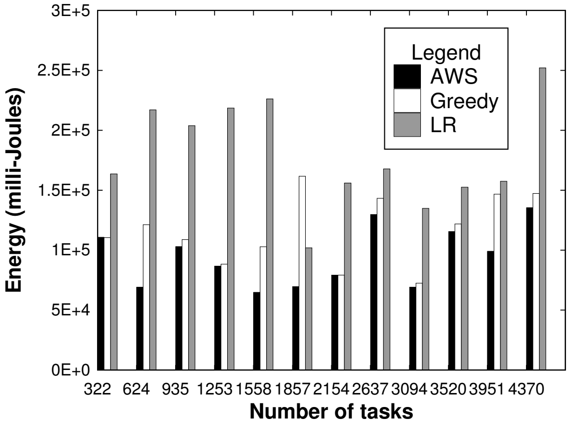

Figure 4: Energy consumption.

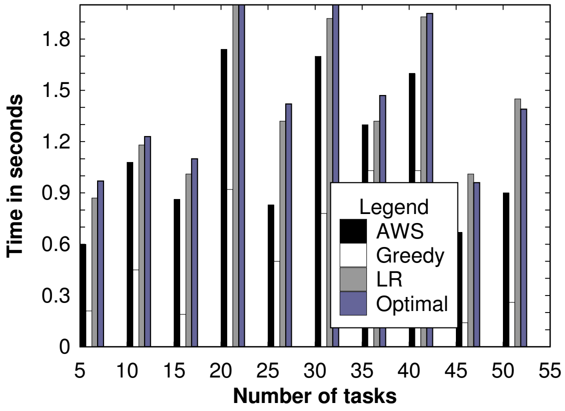

Figure 5: Avg. execution time (small size problems).

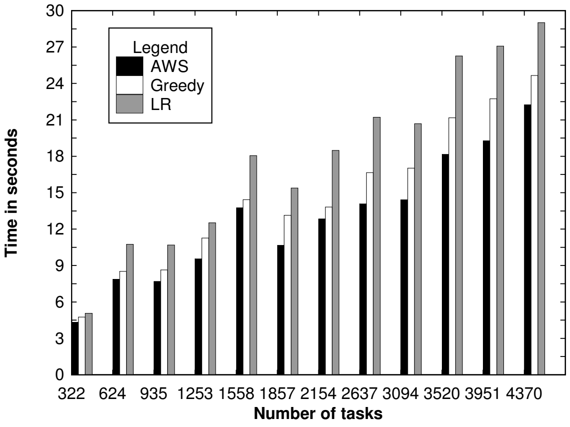

Figure 6: Avg. execution time (large size problems).

5 Related Work

Most DPM techniques utilize instantaneous power management features supported by hardware. For example, Ref. [1] extends the operating system's power manager by an adaptive power manager (APM) that uses the processor's DVS capabilities to decrease or increase the CPU frequency, thereby minimizing the overall energy consumption [2]. The DVS technique at the processor-level together with a turn on/off technique at the cluster-level to achieve high-power savings while maintaining the response time is proposed in [9]. In [8], the authors introduce a scheme to concentrate the workload on a limited number of servers in a cluster such that the rest of the servers can remain switched-off for a longer period of time.

While the closest techniques to combining device power models to build a whole system has been presented in [3], our approach aims at building a general framework for autonomic power and performance management, where we bring together and exploit existing device power management techniques from a whole system's perspective. Furthermore, while most power management techniques are either heuristic-based approaches [4] or stochastic optimization techniques [7], we use adaptive techniques to seek radically fast and efficient solutions compared to the traditional multi-objective optimization techniques.

6 Conclusions

This paper presented a power-aware resource allocation strategy in data centers. The solution quality of the proposed technique was compared against the optimal for small-scale problems, and greedy and linear relaxation heuristics for large-scale problems. The simulation results confirm superior performance of the proposed scheme in terms of reduction in energy consumption and makespan compared to the heuristics and the optimal solution obtained using LINDO. This work leaves much to be desired of in terms of exploring problems of dynamic nature, precedence preserving workload, and preemptive models. We encourage researchers to extend the proposed model described in this paper to capture all of the above desirables.

References

[1] T. F. Abdelzaher and C. Lu, "Schedulability Analysis and Utilization Bounds for Highly Scalable Real-time Services, *7th Real-Time Technology and Applications Symposium*, pp. 15–25, 2001.

[2] R. Bianchini and R. Rajamony, "Power and Energy Management for Server Systems," *IEEE Computer*, vol. 37, no. 11, pp. 68–74, 2004.

[3] J. Chen, M. Dubois, and P. Stenström, "Simwattch: Integrating Complete-system and User-level Performance and Power Simulators," *IEEE Micro*, vol. 27, no. 4, pp. 34–48, 2007.

[4] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini, "Energy Conservation in Heterogeneous Server Clusters," *10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 186–195, 2005.

[5] I. Y. Kim and O. L. de Weck, "Adaptive Weighted-sum Method for Bi-objective Optimization: Pareto Front Generation," *Structural and Multidisciplinary Optimization*, vol. 29, pp. 149–158, 2005.

[6] J. Lin, "Multiple Objective Problems: Pareto-optimal Solutions by Methods of Proper Equality Constraints, *IEEE Transactions on Automatic Control*, vol. 21, pp. 641–650, 1976.

[7] R. Nathuji, C. Isci, and E. Gorbatov, "Exploiting Platform Heterogeneity for Power Efficient Data Centers," *4th International Conference on Autonomic Computing*, 2007.

[8] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load Balancing and Unbalancing for Power and Performance in Cluster-based Systems," *Workshop on Compilers and Operating Systems for Low Power*, 2001.

[9] C. Rusu, A. Ferreira, C. Scordino, and A. Watson, "Energy-efficient Real-time Heterogeneous Server Clusters," *12th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 418–428, 2006.

[10] L. Schrage, *Linear, Integer, and Quadratic Programming with LINDO*, Scientific Press, NY, USA, 1986.

[11] United States Air Force Satellite Control Network Data, Available on-line at: `http://www.cs.colostate.edu/sched/index.html`.

[12] Y. Yu and V. K. Prasanna, "Power-aware Resource Allocation for Independent Tasks in Heterogeneous Real-time Systems," *9th International Conference on Parallel and Distributed Systems*, pp. 341–348, 2002.