Frequency and Speed Setting for Energy Conservation in Autonomous Mobile Robots

Jeff Brateman[1], Changjiu Xian[2], and Yung-Hsiang Lu[3]

[1] Purdue University, West Lafayette, Indiana brateman@purdue.edu
[2] Purdue University, West Lafayette, Indiana cjx@purdue.edu
[3] Purdue University, West Lafayette, Indiana yunglu@purdue.edu

Abstract. Autonomous mobile robots have been achieving significant improvement in recent years. Intelligent mobile robots may detect hazardous materials or survivors after a disaster. Mobile robots usually carry limited energy (mostly rechargeable batteries) so energy conservation is crucial. In a mobile robot, the processor and the motors are two major energy consumers. While a robot is moving, it has to detect an obstacle before a collision. This results in a real-time constraint: the processor has to distinguish an obstacle within the traveled time interval. This constraint requires that the processor run at a high frequency. Alternatively, the robot's motors can slow down to enlarge the time interval. This paper presents a new approach to simultaneously adjust the processor's frequency and the motors' speed to conserve energy and meet the real-time constraint. We formulate the problem as non-linear optimization and solve the problem using a genetic algorithm for both continuous and discrete cost functions. Our experiments demonstrate that more energy can be saved by adjusting both the frequency and the speed simultaneously.

1 Introduction

Autonomous mobile robots provide great potential in transportation, entertainment, environment sensing, search, rescue, reconnaissance, hazard detection, and carpet cleaning [6] [7]. Mobile robots usually carry limited energy, such as rechargeable batteries, so energy conservation is crucial. Makimoto et al. [12] predicted that robots would be a major challenge for future low-power designs. A robot requires many different sensors to detect the environment. Among all sensing technology, stereovision is widely used for determining the distances of obstacles [10] [15]. In a mobile robot, the processor and the motors are two major energy consumers [13]. In this paper, we consider a robot with only one motor, but the method can be generalized to multiple motors.

Even though dynamic voltage scaling (DVS) and energy conservation for mobile robots have been studied [1] [3] [9] [11] [13] [20] [23] [25] [26] [27], the close interaction between computation and motion remains unexplored. This paper presents a probabilistic approach for energy reduction in a mobile robot. We consider a mobile robot moving across an environment with static (i.e. not moving) obstacles, using stereovision to calculate the distance to each obstacle. We assume that each obstacle represents a pass/stop signal, and the minimum distance between signals is a known constant. The robot must recognize the actual distance to the signal before crossing the minimum distance to avoid any chance of failure. The computation cycles needed to recognize the distance to the signals follow a probability distribution. Our method controls both the robot's processor frequency (and voltage) and the motor's speed to reduce the total energy consumption. Our method can save up to 15% additional energy when it is compared with existing solutions that adjust the frequencies only and use constant motor's speeds.

A robot is a real-time system. The processor has to determine the distance of an obstacle before the robot collides with the obstacle. The robot can stop during the distance calculation. However, to conserve energy the robot should be moving while performing this calculation. Many studies have been conducted on energy conservation for real-time systems [9] [18] [25] [27]. Existing studies assume that the deadlines are *externally* determined. For example, a video player has to provide 30 frames per second to prevent jitters. This 33 ms deadline for each frame is given by human's visual perception and cannot be changed by the video player. In contrast, in a mobile robot the deadline is not pre-determined for vision. If the obstacle is static, the robot can slow down or even stop to postpone the deadline before an impending collision. Hence, the deadline is determined by the interaction between the robot's processor and its motor. This paper studies energy conservation in a real-time system in which deadlines can be *internally* adjusted. Our earlier work [3] presents the system using only discrete frequencies and discrete motor speeds, and finds the optimal schedule through an exhaustive search method. We extend this work by generating a schedule using a genetic algorithm. We show this method can obtain a near optimal schedule using both discrete and continuous frequencies and speeds.

2 Related Work

2.1 Probability-Based Voltage Scaling

Some studies have been conducted for dynamic voltage scaling (DVS) by considering the probability distributions of tasks' cycle demands [9] [11] [25] [27]. When different instances of a task's execution cycles follow a known probability distribution, the processor can start at a low frequency (and voltage). If one instance requires fewer cycles, energy is saved because of the lower voltage. If the instance requires more cycles, the processor's frequency gradually rises to ensure that the instance can finish before the deadline. This approach is called *accelerating frequencies.* Lorch et al. [11] use accelerating frequencies for a single task and treat concurrent tasks as a single joint workload. Accelerating frequencies are also used for multiple tasks based on their worst-case execution cycles

(WCEC) [9]. Yuan et al. [27] combine accelerating frequencies with soft real-time constraints for multimedia applications. Xu et al. [25] study accelerating scheduling in systems with discrete frequencies.

Suppose a task demands at most W cycles and the distribution of the cycles is expressed by the cumulative distribution function (CDF). The probability that the j^{th} cycle is needed is $P(j) = 1 - CDF(j-1)$. Note that P is non-increasing because CDF is non-decreasing. Since a task may demand millions of cycles, it is impractical to store the distribution in individual cycles. Thus, we partition $[0, W]$ into n bins and each bin contains b cycles ($b = \lceil \frac{W}{n} \rceil$). The CDF is then a function of the bins. The probability that the j^{th} bin is needed is $P(j) = 1 - CDF(j-1)$. The frequency assigned to the j^{th} bin is f_j and the execution time for this bin is $\frac{b}{f_j}$. The processor's power is proportional to $v^2 f$ and $v \propto f$ (here v is the voltage). The energy for this bin is $(v_j^2 f_j) \times \frac{b}{f_j} \propto b f_j^2$. The expected energy consumption for this bin is proportional to the product of the energy and the probability: $b f_j^2 P(j)$. Suppose the task is released at time zero and the deadline is t. The goal is to find a schedule $\{f_1, f_2, ..., f_n\}$ to minimize the total expected energy. This is formulated as follows.

$$\text{minimize} \sum_{1 \le j \le n} b f_j^2 P(j) \tag{1}$$

$$\text{subject to} \sum_{1 \le j \le n} \frac{b}{f_j} \le t \tag{2}$$

Based on earlier studies [11] [25] [27], the optimal schedules can be obtained by assigning f_j:

$$f_j = \frac{\sum_{i=1}^{n} b \sqrt[3]{P(i)}}{t \sqrt[3]{P(j)}} \tag{3}$$

2.2 Energy Conservation for Mobile Robots

Batteries are often used to provide power for mobile robots; however, batteries are heavy and have limited energy capacity. A Honda humanoid robot can walk for only 30 minutes with a battery pack [1]. Rybski et al. [20] show that power consumption is one of the major issues in robot design. Sun et al. [23] present an algorithm for finding the energy-efficient paths on terrains. Yamasaki et al. [26] present an energy-efficient walk generation algorithm for a humanoid robot. A case study [13] shows that motor power is less than 50% of the total power in a mobile robot. Hence, the power for electronic components cannot be ignored. In recent years, small robots have been studied for sensing [2] [5] [7] [21].

2.3 Image Correspondence for Stereovision

Robots can detect their surroundings, including distances to objects with two cameras and stereovision. Several advances have made stereovision both precise

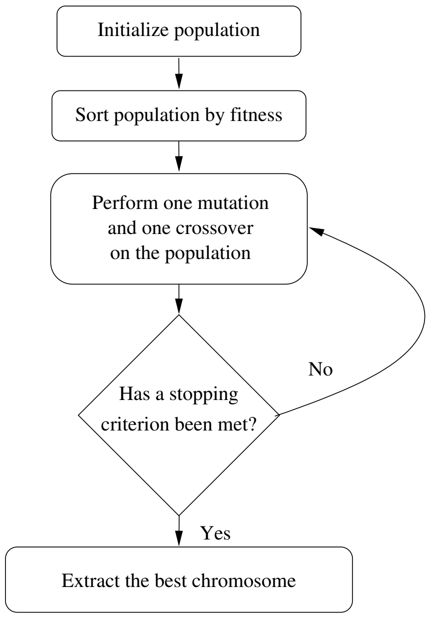

Initialize population
Sort population by fitness
Perform one mutation and one crossover on the population
Has a stopping criterion been met?
No
Yes
Extract the best chromosome

Fig. 1. A simplified view of the GENITOR algorithm.

and accurate [10]. Redert et al. [19] show the advances made for those seeking high-accuracy, high-resolution 3D scene acquisition. Stereovision has been used in mobile robots for both navigation, and terrain mapping [15] [16].

2.4 Genetic Algorithm

Genetic algorithms have been used in many practical applications [4] for problems where optimal schedules take more than polynomial time to find. GENITOR [24] is a steady-state genetic algorithm that has been shown to perform well for several problem domains [17] [22] such as resource allocation, job shop scheduling, and neural networks. A simplified view of the GENITOR algorithm is shown in Figure 1. To generate a better schedule using the GENITOR algorithm, several steps are performed. First, an initial population is generated, either through simple heuristics or random generations. The population consists of many chromosomes, or schedules in the search space. Next, the algorithm performs evolution until a stopping criterion is reached, such as reaching a maximum number of iterations or a homogeneous population.

In every iteration, one mutation and one crossover operations are performed. If the chromosome generated by a mutation or a crossover is better than the worst chromosome in the population, the new chromosome is inserted into the sorted population and the worst chromosome is removed. The fitness function is the criteria which allow a chromosome to be ranked better than another. The probability of selecting a chromosome for the mutations and crossovers is given by the linear bias function defined in [24]. To achieve the linear bias effects, the chromosomes remain sorted by their evaluation of the fitness function.

2.5 Paper Contributions

This paper makes the following contributions: (a) We consider a real-time system in which the deadline is determined by the interaction between two components: processor and motor. (b) The overall energy consumption is modeled as an optimization problem. (c) A probabilistic solution is presented to find the processor's frequency and the motor's speed. (d) We then use a genetic algorithm to find a sub-optimal schedule quickly. (e) We consider continuous processor frequencies and continuous motor speeds, and we use the genetic algorithm to obtain an energy-efficient schedule.

3 Problem Formulation

This section formulates the problem to conserve the energy of a mobile robot by adjusting the robot's processor frequency and the motor's speed. We first use a motivating example to illustrate the important concept and then formulate the problem as a probabilistic non-linear optimization problem. Next, we discuss the properties of the formulation presented in Section 3.2. We describe how to solve the optimization problem using discrete frequencies and discrete speeds with an exhaustive search. Then we use a genetic algorithm to find energy-efficient schedules for either discrete or continuous frequencies and speeds.

3.1 Motivating Example

Suppose the total power of a robot's motor is $s^2 + s + 1$ at speed s meters per second. Here, the constant 1 is used to model the DC loss of the motor. The processor's power consumption is $f^3 + 1$ at frequency f MHz and a constant leakage power of 1. Suppose the robot has to travel along a road. The road contains signs indicating whether the robot can pass or has to stop. The signs do not change (unlike traffic lights) and the minimum distance between two adjacent signs is 100 meters. Even though the distance between signs may be larger than 100 meters, the robot must recognize the sign by the time it has traveled the minimum distance to guarantee success, as shown in Figure 2. If the robot fails to recognize the sign in time, the robot may collide with the sign and fail.

We define the optimal speed as the speed to consume the minimum energy per unit distance. Suppose the minimum distance between signs is D. The time to cross this distance is $\frac{D}{s}$. The total energy consumption is $(s^2 + s + 1)\frac{D}{s}$ and the energy per unit distance is $\frac{s^2+s+1}{s} = s + 1 + \frac{1}{s}$. Thus, the optimal speed is 1

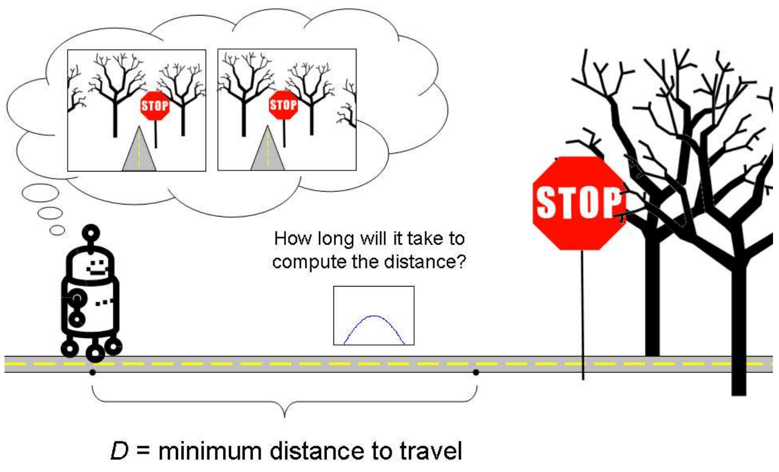

STOP
STOP
STOP
How long will it take to compute the distance?
D = minimum distance to travel

Fig. 2. Problem formulation showing that a robot must travel a minimum distance while completing a task (stereovision distance calculation) with uncertain execution time.

meter per second. If the robot moves at this speed, it takes 100 seconds to cross the minimum distance. The worst-case execution cycle is 150 million cycles and the processor has to operate at 1.5 MHz to ensure recognizing every sign before the robot reaches the sign. The total energy consumed by the motor at $s = 1$ is $3 \times 100 = 300$ J. The total energy consumed by the processor at $f = 1.5$ MHz is $(f^3 + 1)\frac{cycles}{f} = 4.38 \times 100 = 438$ J. The overall energy is 738 J to cross the minimum distance between two signs.

If we consider the power of the motor and the processor simultaneously, we can reformulate the problem as follows. The time to cross the distance is $\frac{100}{s}$ at speed s. The processor has to operate at $1.5s$ MHz to meet the deadline. The total energy is $\frac{100}{s} \times \{(s^2 + s + 1) + [(1.5s)^3 + 1]\}$. The minimum energy value occurs when $s \approx 0.62$ and the overall energy consumption is 614 J, or a 17% reduction from 738 J. This shows the importance of considering both frequency and speed simultaneously.

We consider a further extension of this example. The computation cycles vary due to the scene complexity surrounding the signs: among all signs, 30% require only 50 million cycles, 40% for 100 million cycles, and the remaining 30% for 150 million cycles. The probability can be expressed in the following way. The first 50 million cycles are always needed so the probability is 100%. The second 50 million cycles are needed with probability 70%. Finally, the last 50 million cycles are needed with a probability of only 30%. With this additional information, we can compute the *expected*, rather than the worst-case, energy consumption. We want to lower the expected energy, but still finish detecting the sign in the worst case. If the motor's speed is a constant at 1 m/s, the deadline is 100 seconds. We can adopt the strategy with accelerating frequencies

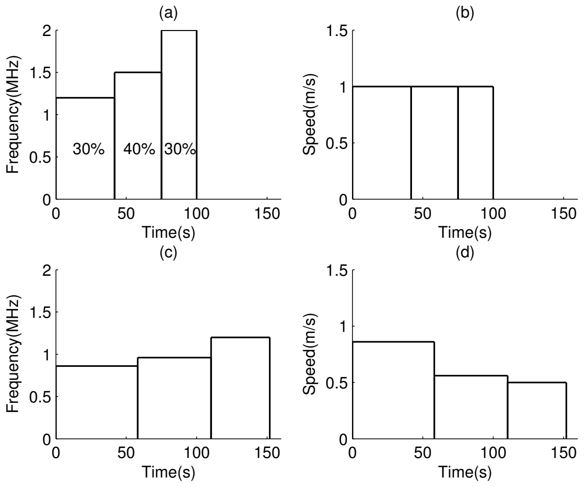


Fig. 3. Processor and motor scaling schedule assuming a constant motor speed in (a) and (b). Processor and motor scaling schedule if the motor speed is allowed to change in (c) and (d).

explained in Section 2.1 as shown in Figures 3 (a) and (b). The overall system saves energy in average cases because most tasks need only 50 or 100 million cycles. Meanwhile, the system still meets the deadline in the worst cases by using a higher frequency when needed. This, however, results in an energy consumption of 611 J, less than 1% reduction from 614 J. We can consider accelerating frequencies for the processor and simultaneously *decelerating speeds* for the motor and save more energy, as shown in Figures 3 (c) and (d). By decreasing the motor's speed, the processor's frequency does not have to rise significantly, and its expected energy is reduced substantially. This approach can further reduce the expected energy to 529 J in this example, or 14% additional savings. The following sections will explain how to determine the frequency and the speed simultaneously to achieve better energy savings.

3.2 Constrained Optimization Problem

The minimum distance between two signs is a known constant, D. The maximum number of cycles needed for recognition is W and is divided into n bins. Each bin has $b = \lceil \frac{W}{n} \rceil$ cycles. We use $P(i)$ to represent the probability that the i^{th} $(1 \leq i \leq n)$ bin of cycles is needed. As defined in Section 2.1, $P(i) = 1 - CDF(i-1)$ and $P(i) \geq P(i+1)$. The processor operates at frequency f_i for the i^{th} bin. When the processor is computing for the i^{th} bin, the robot moves at speed s_i. The execution time for the i^{th} bin is $\frac{b}{f_i}$. The distance traveled during this time is $d_i = s_i \frac{b}{f_i}$. The timing constraint is that the processor has to finish the computation of all bins before the robot crosses the distance of D. In other words, the sum of d_i cannot exceed D:

$$\sum_{i=1}^{n} d_i \leq D \Rightarrow \sum_{i=1}^{n} \frac{bs_i}{f_i} \leq D \tag{4}$$

Let $\alpha(f_i)$ be the power consumption of the processor at frequency f_i when voltage scaling is also applied. When the processor finishes the task, the processor's frequency can be reduced to zero. In this case, the processor consumes static power $\alpha(0)$. Let $\beta(s_i)$ be the power consumption of the motor at speed s_i. The *expected* energy for crossing the distance is the sum of the processor energy and the sum of the motor energy over all bins. The energy consumed can be divided into two parts: (i) when the processor is still computing, and (ii) when all computation has finished.

When the i^{th} bin is being computed, the processor consumes power $\alpha(f_i)$ and the motor consumes power $\beta(s_i)$. The duration of this bin is $\frac{b}{f_i}$, and this occurs with probability $P(i)$. Therefore, the expected energy is

$$\sum_{i=1}^{n} \frac{P(i)b}{f_i}(\alpha(f_i) + \beta(s_i)) \tag{5}$$

To compute the energy in (ii), we have to first determine the distance the robot has traveled while the processor is computing. The total expected distance traveled is $\sum\limits_{i=1}^{n} \frac{bP(i)s_i}{f_i}$ and the remaining distance is $D - \sum\limits_{i=1}^{n} \frac{bP(i)s_i}{f_i}$. When the robot is traveling through this remaining distance, the processor is turned off and consumes idle power $\alpha(0)$. Let s_o be the speed for the remaining distance. The time to cross the remaining distance is $\frac{1}{s_o}(D - \sum\limits_{i=1}^{n} \frac{bP(i)s_i}{f_i})$. Hence, the total expected energy is

$$\frac{1}{s_o}(D - \sum_{i=1}^{n} \frac{bP(i)s_i}{f_i})[\alpha(0) + \beta(s_o)] \tag{6}$$

The optimization problem is to find the values of f_i and s_i $(1 \leq i \leq n)$ and s_o for minimizing the sum of (5) and (6).

$$\min \sum_{i=1}^{n} \tfrac{P(i)b}{f_i}(\alpha(f_i) + \beta(s_i)) + \tfrac{1}{s_o}(D - \sum_{i=1}^{n} \tfrac{bP(i)s_i}{f_i})[\alpha(0) + \beta(s_o)] \tag{7}$$

with the constraint in (4). This is a problem of constrained optimization.

3.3 Frequency and Speed Scheduling

The above formulation has three sets of variables: the processor's frequency f_i, the motor's speed s_i, and time. The time intervals have been discretized; hence, the frequency and the speed can change only at the boundaries of bins. Each bin takes b clock cycles on the processor. We use $P(i)$ to express the probability that the i^{th} bin is needed. In our formulation, the time intervals are not divided into equal durations (measured by seconds). Instead, the duration of the i^{th} interval is determined by the ratio of b and f_i in order to simplify the expression in (7). It is possible to generalize the formulation and use continuous time so that (a) The duration of a constant frequency is not determined by the value of this frequency. (b) The frequency and the speed do not have to change at the same time. If we use continuous time to model the problem, the frequency and the speed are expressed as $f(t)$ and $s(t)$ respectively. The search space becomes substantially larger and it is difficult to find optimal schedules. Hence, in the rest of this paper, we use discrete time by allowing the frequency and the speed to change only at the boundaries of bins.

Our solution uses accelerating frequencies (i.e. $f_i \leq f_{i+1}$, $1 \leq i \leq n-1$) and decelerating speeds (i.e. $s_i \geq s_{i+1}$, $1 \leq i \leq n-1$). To find the initial values for f_1 and s_1, we examine the schedulability of the problem using the constraint of inequality (4). The initial value of f_1 is the lowest frequency to satisfy (4) when all s_i's are assigned the lowest speed. Similarly, the initial value of s_1 is the highest speed to satisfy (4) when all f_i's are assigned the highest frequency. If f_1 exceeds the highest available frequency or s_1 is below the minimum available speed, no schedule can be found. After finding the initial values for f_1 and s_1, we enumerate all feasible schedules and find the schedule that provides the minimum expected energy and meets the constraint in (4). For a small value of n, it takes only several minutes on a modern computer to find the optimal schedule. This schedule can be computed off-line, and loaded into the robot so that it can change to the correct speed and frequency while the task is still executing. As n increases, the time to find this schedule becomes more important, as there are many more combinations of f_i and s_i. This becomes a problem of scalability with the number of bins.

In most cases, it is impractical to wait hours to generate a schedule for different values of D. This is especially true for dynamic environments and unknown operating environment. Therefore, it is preferable to find a schedule quickly even though it may not be optimal.

3.4 Optimization using Genetic Algorithm

To determine a schedule in a reasonable amount of time, a genetic algorithm is used. Even though the genetic algorithm does not guarantee to reach the optimal schedule, we will show that the schedule produced still saves energy and approaches the optimal schedule. The technique is one adopted from [24].

A chromosome contains all the frequency and speed assignments for the mobile robot for each bin. Each chromosome contains a value for f_i and s_i ($1 \leq i \leq n$) and has $2n$ parameters, where n is the number of bins in the problem. In the discrete case, f_i and s_i are restricted to a limited set of discrete frequencies and discrete speeds, predetermined before the algorithm is run. We are minimizing the expected energy, therefore the fitness function is Equation 7.

The initial population is composed of arbitrary chromosomes. We select the first chromosome to have all frequencies set to the maximum available frequency, and all speeds set to their minimum speeds. If this schedule is not feasible, i.e., the processor still cannot finish executing the task before the robot arrives at its destination, then no schedule can be found. The remainder of the chromosomes are randomly generated.

For mutation, either one or two parameters are selected at random to change. A parameter is one of any f_i or s_i. Allowing two randomly selected parameters to change produces better schedules than changing only one parameter. In our algorithm, half of the multations change two parameters in a schedule simultaneously. Changing only one parameter in a schedule may result in being trapped in a local minimum because the mutated chromosomes are either infeasible (robot no longer meets its deadline) or increase the power consumption. To perform the crossover operation, each parameter is ordered, and a cutoff point is determined at random. This is shown in Figure 4. Any parameters before the cutoff remain the same as their parents and any chromosomes parameters after the cutoff are swapped from one parent to the other. This creates two potentially better child chromosomes.

4 Simulations

4.1 Overview

We consider both discrete and continuous frequency and speed settings for our experiments. The experiments were conducted over several workloads, using both an exhaustive search method and a genetic algorithm. Our simulations show up to 15% energy savings over those methods that scale processor frequencies only.

4.2 Hardware Models

Table 1. XScale's frequency/voltage and power.

Frequency(MHz)	150	400	600	800	1000
Voltage(V)	0.75	1.0	1.3	1.6	1.8
Power(mW)	80	170	400	900	1600

We use the voltage and frequency settings of the Intel XScale processor [25]. For the discrete experiments, we use five discrete frequency settings. Their associated power consumption is shown in Table 1. For the continuous frequency

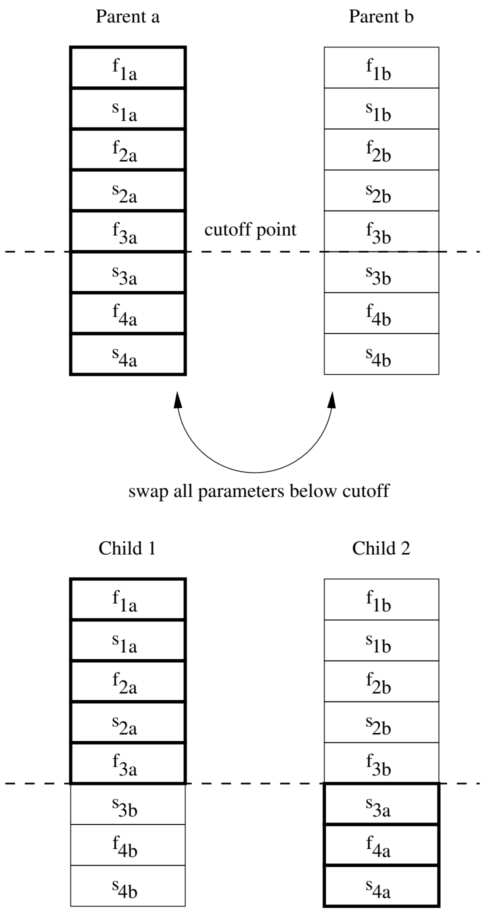


Fig. 4. Crossover operation of a sample chromosome with 4 bins and 8 parameters.

settings, we use a third-order polynomial power model based on the discrete values in Table 1 and allow the frequency to vary anywhere between 150MHz and 1GHz. The motor power is from the measurements performed by Mei et al. [14] shown in Figure 5. We limit the motor's speed between 0.5 m/s and 5 m/s with 0.5 m/s as the step size, for the discrete case. For the continuous motor speeds, we limit the motor speeds to a range of 0.5 m/s and 5 m/s. All of our calculations assume the minimum distance to travel is 500 meters for D's value.

All experiments were performed using Matlab 7.1 running on Windows XP SP2. The hardware consisted of an Intel Pentium 4 CPU running at 3.4 GHz with 1 GB of RAM. These values are important for the execution time of the genetic algorithm and the exhaustive search, as seen later in Figure 12.

We compare our approach with three other methods. The first uses a constant frequency and a constant speed. The frequency and the speed are selected

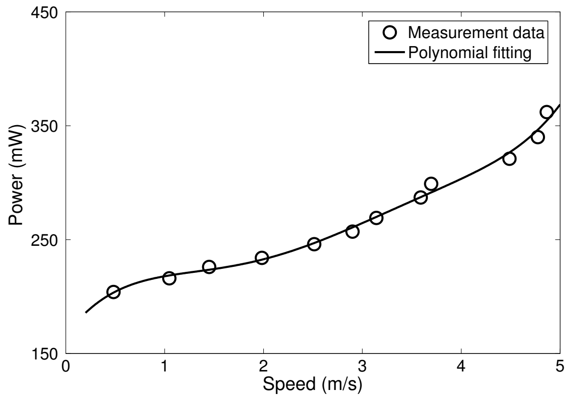


Fig. 5. Power efficiency of a robot at different speeds.

from the discrete settings such that they minimize the total energy consumption and satisfy the constraint. In the synthetic distributions, a search finds the optimal energy consumption schedule that meets the constraint to be a frequency of 400 MHz and a speed of 1.5 m/s. The second uses a constant speed and accelerating frequencies. The third uses a constant frequency and decelerating motor speeds. The processor frequency is set to the middle frequency 600 MHz. The fourth uses both accelerating frequencies and decelerating speeds; this is the method proposed in this paper.

4.3 Workloads

We use two types of workloads: synthesized workloads with different distribution functions, and a distribution function generated from captured stereo images.

The synthetic benchmarks have distributions of uniform, Gaussian, and exponential functions. These synthetic workloads have worst-case execution cycles (WCEC) of 100 billion cycles. For the uniform distribution, the actual number of needed cycles is between 0 and WCEC. For the Gaussian distribution, the mean is half WCEC and the standard deviation is a quarter WCEC. For the exponential distribution, the mean is a quarter WCEC. The distributions are normalized after removing the negative cycles and the cycles above WCEC. We varied the mean and the standard deviation (STD) of the synthetic workloads to show how different values affect the energy savings of our schedule. The energy savings calculations are done using the genetic algorithm with continuous frequencies and continuous speeds. Each run is performed over a range of means, where each mean is calculated as a percentage of the original $WCEC$. To find average energy savings, three calculations are performed for each workload. The uniform and Gaussian workloads are generated by selecting a value of the standard deviation. The exponential workload is generated only over variable means.

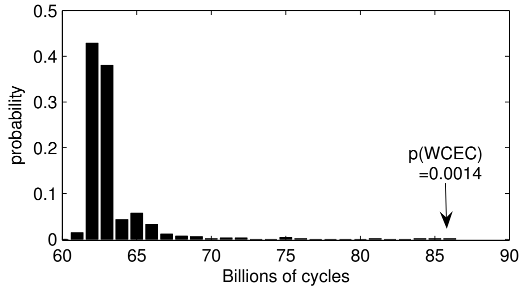


Fig. 6. Probability distribution of stereovision computations.

We generated the image workload from pairs of stereovision images taken from the image database of the city of West Lafayette and Indianapolis in the state of Indiana [8]. Pairs of stereo images are compared, and distances for several objects are returned.

4.4 Experimental Results

The experiments compare our method with several workloads. We analyze an image processing algorithm to obtain the distribution of execution cycles, and how our method performs on the workload. The genetic algorithm experiments show how a schedule can be obtained in a reasonable time, even if it is not optimal. We then show how altering synthetic workloads affects the energy savings.

Figure 6 shows the distribution of the needed cycles for running the correspondence programs on 700 pairs of images. Note that there is great potential for energy savings as the probability of the WCEC (85.7 billion cycles) is only 0.14%. We can see that the majority of tasks execute in around 62 billion cycles.

Figure 7 shows the relative energy consumption of the four methods for the four benchmarks, using the exhaustive search method. All numbers are normalized related to the first method with a constant frequency and a constant speed. As can be seen in this figure, our method can save 20% to 50% energy compared with the first method in the four benchmarks. Compared with the second and the third methods, our method can save an additional 7% to 15% energy. These results are generated using 10 bins.

An exponential distribution shows the greatest potential for savings as compared with the constant frequency and the constant motor speed schedule. In an exponential distribution, the task finishes quickly more often, and has a low probability of finishing near the WCEC. We can see the potential for reducing the expected energy as opposed to WCEC scheduling. In the stereovision distribution, the energy savings is not as high as the exponential distribution because no task finishes before $\frac{2}{3} \times WCEC$ cycles, but our method still saves 20% energy over the constant frequency and constant motor speed scheduling.

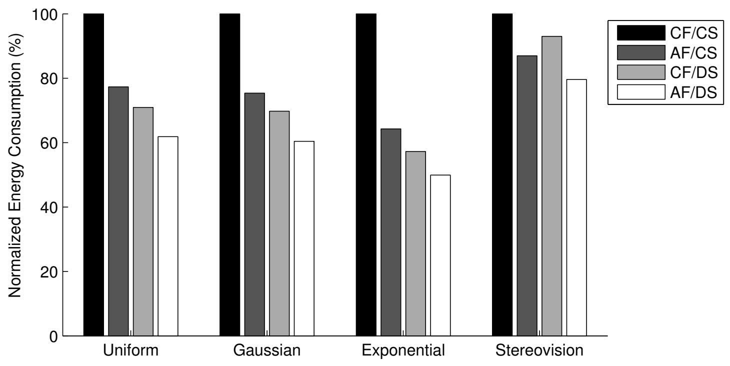


Fig. 7. Normalized energy consumption from the set of the three synthetic tasks with uniform, Gaussian, and exponential distributions, and the stereovision benchmark, respectively. The four methods use either constant frequencies (CF) or accelerating frequencies (AF), and constant speeds (CS) or decelerating speeds (DS).

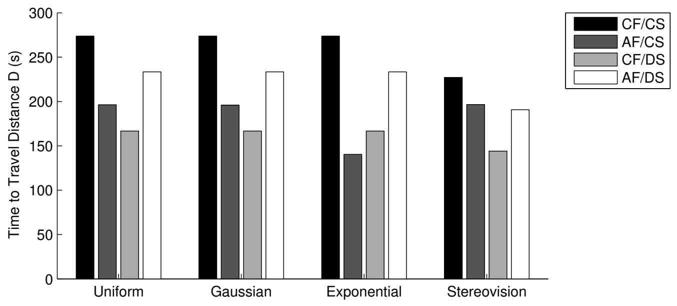


Fig. 8. Time to finish each of the three synthetic task and the stereovision benchmark for a task of WCEC cycles.

One advantage of using our method over a constant frequency and constant speed schedule is that our method will not necessarily increase the worst case travel time of the robot to the minimum distance D. For this analysis, we assume that the task takes the maximum number of cycles to execute, namely, its WCEC. Figure 8 shows the time required to travel the minimum distance. We see that in all cases, dividing the frequency and speed schedule into 10 bins allows the robot to tune its speed better, so that the robot takes less time traveling the minimum distance than the constant frequency and constant speed schedule allows.

Figure 9 shows the energy consumption for a growing number of bins, using discrete parameters. This figure indicates that energy consumption decreases as the number of bins grows because more frequencies and speeds can be used.

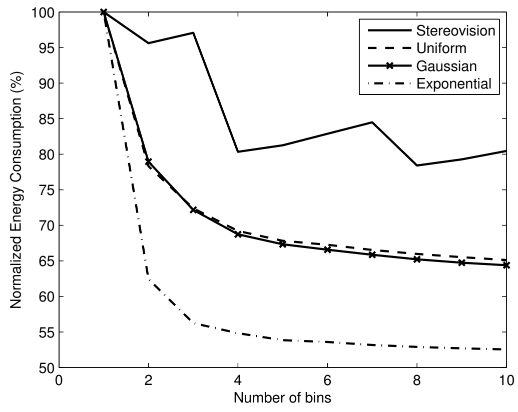


Fig. 9. Energy consumption of the optimal schedule of each benchmark over a different number of bins using discrete parameters and the exhaustive search method.

It should be noted that with the stereovision distribution, the energy actually increases in some cases. This is due to the division of the PDF into a relatively small number of bins. Some of the areas with high probability are divided in some sizes of n, resulting in an increased expected energy. However, energy is still reduced from the extreme case of one bin. Because of the small number of bins used to compute the frequency and the speed schedule, our method can be applied to practical systems, even though the method has exponential computation time.

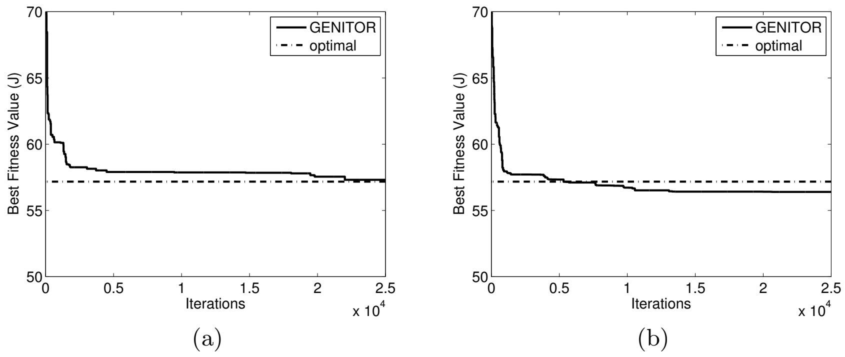


Fig. 10. GENITOR improvement versus the number of iterations compared with the optimal energy consumption for the exponential workload with *discrete* (a) and *continuous* (b) frequencies and speeds.

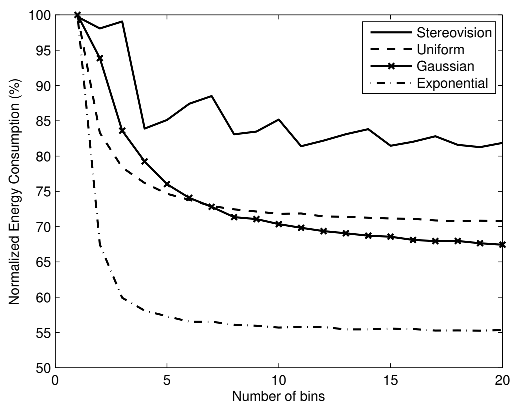


Fig. 11. Energy consumption of each benchmark over different number of bins using continuous parameters and a genetic algorithm to find a schedule.

Genetic Algorithm The genetic algorithm had a population of 50 chromosomes, starting with 50 random schedules. We can see from Figure 10 (a) that the energy consumed approaches the exhaustive search optimal schedule after only a few thousand iterations. For the exponential distribution, the energy consumed by the schedule generated by the GENITOR algorithm was within 0.24% of the energy consumed by the optimal schedule. Other distributions' simulations perform in a similar manner, and all schedules were computed in about 4 minutes.

We can see in Figure 10 (b) the results of running the genetic algorithm using continuous parameters compared with the optimal schedule using discrete parameters. The result is 14% more energy-efficient than the discrete optimal schedule. This figure shows the advantage of using continuous parameters over discrete parameters.

We show the effects of increasing the number of bins in Figure 11. Increasing the number of bins increases the number of parameters that can be adjusted. These schedules were calculated using continuous parameters, as these were shown to provide better schedules than discrete parameters. This figure can be compared to Figure 9, where each graph is calculated using discrete parameters and normalized with the original one bin discrete parameter schedule. The figure also indicates that the energy consumption begins to approach diminishing returns as the number of bins exceeds 15. In other words, a large number of bins cannot provide a significant amount of additional savings.

We use a genetic algorithm over an exhaustive search to reduce the time for finding energy-efficient schedules. In Figure 12, we show the execution time of the exhaustive search compared with the execution time of GENITOR. No times were recorded for 1 or 2 bins exhaustive search because the execution

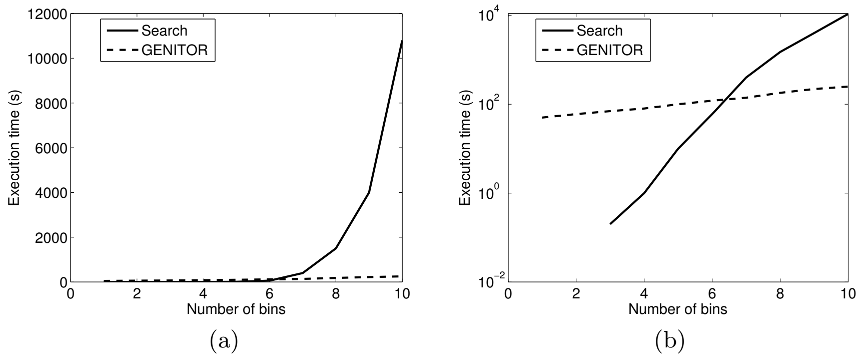


Fig. 12. Execution time of the exhaustive search method and GENITOR algorithm as the number of bins increases over a linear scale (a) and a log scale (b), using discrete processor frequencies and motor speeds.

time was negligible. We can see that using a genetic algorithm will begin to save hours as the number of bins increases.

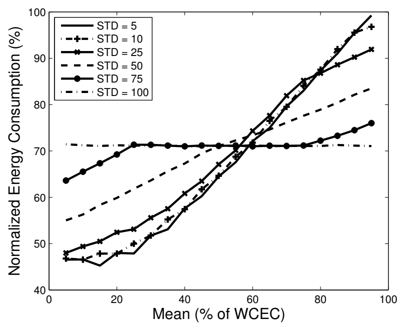
Variable Synthetic Workloads The uniform distribution results are shown in Figure 13 (a). The figure shows the increased energy consumption as the mean increases. For a small STD value, the task executing almost always executes its worst case execution cycles, while the workloads with a large STD have constant energy consumption over the selected ranges of the mean. This occurs when probabilities that are assigned to bins below 0% or above 100% get clipped, and are normalized so they sum to one. The result is the appearance of constant energy savings.

Figure 13 (b) shows the Gaussian distribution results. For each STD, we see that the energy consumption increases as the mean increases. This increase becomes more significant as the STD increases. With a large STD, the distributions approach a uniform distribution for each mean, therefore the energy savings remains constant. We also see the crossing point in the middle because a distribution with a large STD performs as well as a schedule with a small STD with a mean of 50% WCEC.

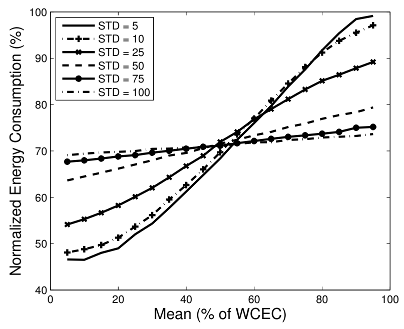
The exponential distribution results are shown in Figure 13 (c). The figure indicates the increasing energy consumption as the mean increases, but takes on a different shape than the other workloads. This is because even with the increased mean, the majority of tasks will complete early. The rate of increase with mean is small, and our method still saves energy in the worst case.

5 Conclusions

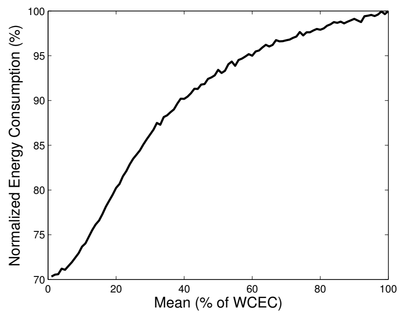
This paper presents a method to simultaneously scale processor frequencies and motor speeds for autonomous robots with hard deadlines. However, each dead-



(a)



(b)



(c)

Fig. 13. Energy consumption over a range of means of uniform (a), Gaussian (b), and exponential (c) synthetic workloads.

line is not a time deadline, rather it is a distance deadline. This problem is formulated as an optimization problem. An exhaustive search method is presented to find the optimal solution among discrete processor frequencies and motor speeds. A genetic algorithm is used to find a near-optimal solution in less time than the exhaustive search. The genetic algorithm is modified so that it can handle continuous processor frequencies and motor speeds.

A probability distribution of the number of cycles required for stereovision distance calculation is used for our simulations, along with three synthetic distributions. Our experimental results show that we achieve energy savings from 7% to 15% more than only scaling the processor frequency. These results can be achieved through the calculation of an optimal schedule off-line. We can save more energy if continuous processor frequencies and motor speeds are available using the genetic algorithm. We also show that the genetic algorithm can be used for greater energy savings with increasing numbers of bins.

6 Acknowledgments

This work is supported in part by National Science Foundation IIS-0329061, CNS-0347466, and Purdue Research Foundation. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

References

1. Aylett, R.: Robots: Bringing Intelligent Machines To Life. Barrons (2002)
2. Birch, M.C., Quinn, R.D., Hahm, G., Phillips, S.M., Drennan, B., Fife, A., Verma, H., Beer, R.D.: Design of a Cricket Microrobot. In: International Conference on Robotics and Automation, pp. 1109–1114 (2000)
3. Brateman, J., Xian, C., Lu, Y.H.: Energy-efficient scheduling for autonomous mobile robots. In: IFIP International Conference on Very Large Scale Integration (VLSI-SoC), pp. 361–366 (2006)
4. Chaiyaratana, N., Zalzala, A.: Recent developments in evolutionary and genetic algorithms: Theory and applications. In: International Conference on Genetic Algorithms In Engineering Systems:Innovations And Applications, pp. 270–277 (1997)
5. Colot, A., Caprari, G., Siegwart, R.: InsBot: Design of An Autonomous Mini Mobile Robot Able To Interact With Cockroaches. In: International Conference on Robotics and Automation, pp. 2418–2423 (2004)
6. Davids, A.: Urban Search and Rescue Robots: From Tragedy To Technology. IEEE Intelligent Systems **17**(2), 81–83 (2002)
7. Drenner, A., Burt, I., Dahlin, T., Kratochvil, B., McMillen, C., Nelson, B., Papanikolopoulos, N., Rybski, P.E., Stubbs, K., Waletzko, D., Yesin, K.B.: Mobility Enhancements to the Scout Robot Platform. In: International Conference on Robotics and Automation, pp. 1069–1074 (2002)
8. Gautam, S., Sarkis, G., Tjandranegara, E., Zelkowitz, E., Lu, Y.H., Delp, E.J.: Multimedia for Mobile Users: Image Enhanced Navigation. In: Multimedia Content Analysis, Management, and Retrieval, IST/SPIE Symposium on Electronic Imaging (2006)
9. Gruian, F.: Hard Real-Time Scheduling for Low-Energy Using Stochastic Data and DVS Processors. In: International Symposium on Low Power Electronics and Design, pp. 46–51 (2001)
10. Lenz, R.K., Tsai, R.Y.: Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology. IEEE Transactions onPattern Analysis and Machine Intelligence **10**(5), 713–720 (1988)
11. Lorch, J.R., Smith, A.J.: Improving Dynamic Voltage Scaling Algorithms with PACE. In: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 50–61 (2001)
12. Makimoto, T., Sakai, Y.: Evolution of Low Power Electronics and Its Future Applications. In: International Symposium on Low Power Electronics and Design, pp. 2–5 (2003)
13. Mei, Y., Lu, Y.H., Hu, Y.C., Lee, C.G.: A Case Study of Mobile Robot's Energy Consumption and Conservation Techniques. In: International Conference on Advanced Robotics, pp. 492–497 (2005)

14. Mei, Y., Lu, Y.H., Hu, Y.C., Lee, C.S.G.: Energy-Efficient Motion Planning for Mobile Robots. In: International Conference on Robotics and Automation, pp. 4344–4349 (2004)
15. Murray, D., Jennings, C.: Stereo Vision Based Mapping and Navigation for Mobile Robots. In: IEEE International Conference on Robotics and Automation, vol. 2, pp. 1694–1699 (1997)
16. Negishi, Y., Miura, J., Shirai, Y.: Vision-Based Mobile Robot Speed Control Using a Probabilistic Occupancy Map. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, pp. 64–69 (2003)
17. Oltikar, M., Brateman, J., White, J., Martin, J., Knapp, K., Maciejewski, A.A., Siegel, H.J.: Robust resource allocation in weather data processing systems. In: International Conference on Parallel Processing Workshops, pp. 445–454 (2006)
18. Pillai, P., Shin, K.G.: Real-time Dynamic Voltage Scaling for Low-power Embedded Operating Systems. In: ACM Symposium on Operating Systems Principles, pp. 89–102 (2001)
19. Redert, A., Hendriks, E., Biemond, J.: Correspondence Estimation in Image Pairs. IEEE Signal Processing Magazine **16**(3), 29–46 (1999)
20. Rybski, P.E., Papanikolopoulos, N.P., Stoeter, S.A., Krantz, D.G., Yesin, K.B., Gini, M., Voyles, R., Hougen, D.F., Nelson, B., Erickson, M.D.: Enlisting Rangers and Scouts for Reconnaissance and Surveillance. IEEE Robotics and Automation Magazine **7**(4), 14–24 (2000)
21. Sibley, G.T., Rahimi, M.H., Sukhatme, G.S.: Robomote: A Tiny Mobile Robot Platform for Large-scale Ad-hoc Sensor Networks. In: International Conference on Robotics and Automation, pp. 1143–1148 (2002)
22. Sugavanam, P.V., Siegel, H.J., Maciejewski, A.A., Ali, S.A., Al-Otaibi, M., Aydin, M., Guru, K., Horiuchi, A., Krishnamurthy, Y.G., Lee, P., Mehta, A.M., Oltikar, M., Pichel, R., Pippin, A.J., Raskey, M., Shestak, V., Zhang, J.: Processor allocation for tasks that is robust against errors in computation time estimates. In: International Parallel and Distributed Processing Symposium (2005)
23. Sun, Z., Reif, J.: On Energy-Minimizing Paths on Terrains for A Mobile Robot. In: International Conference on Robotics and Automation, pp. 3782–3788 (2003)
24. Whitley, L.D.: The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In: International Conference on Genetic Algorithms, pp. 116–123 (1989)
25. Xu, R., Xi, C., Mehlem, R., Moss, D.: Practical PACE for Embedded Systems. In: ACM International Conference On Embedded Software, pp. 54–63 (2004)
26. Yamasaki, F., Hosoda, K., Asada, M.: An Energy Consumption Based Control for Humanoid Walking. In: IEEE/RSJ International Conference on Intelligent Robots and System, pp. 2473–2477 (2002)
27. Yuan, W., Nahrstedt, K.: Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems. In: ACM Symposium on Operating Systems Principles, pp. 149–163 (2003)