

Evaluation of Walk Optimisation Techniques for the NAO Robot

Jason Kulk and James S. Welsh

School of Electrical Engineering and Computer Science

University of Newcastle, Australia

jason.kulk@uon.edu.au

Abstract—Locomotion performance is a critical component of any humanoid robot application. The procedure of optimising a walk engine has a high cost in both resources and time. The selection of the most appropriate optimisation algorithm, fitness function, and parameter space to maximise the benefit–cost ratio can dramatically improve the performance of the optimisation process.

In this paper, we compare different meta-optimised optimisation algorithms, different fitness functions, and two different parameter spaces, in a physics-based simulation. The purpose of the comparison is to select the most appropriate combination to be used in hardware. The combination that yields the greatest increase in walk performance given a fixed expenditure is considered as the best, and is implemented in hardware.

We found that Policy Gradient Reinforcement Learning with a fitness function based on the efficiency and a parameter space expanded to include the joint stiffnesses not only performed the best, in terms of improving the walk speed and efficiency, but also in terms of selecting gaits that were more stable. This combination was then applied to the physical NAO, improving the default walk’s speed by 57% and its efficiency by 30%.

I. INTRODUCTION

Humanoid robot locomotion is a challenging area that has been the topic of a significant body of research [1]–[4]. However, the performance of current walk engines leaves much to be improved upon. In addition to developing a new improved walk engine, an existing engine can be improved through optimisation of walk parameters [5], [6].

The process of optimising a walk engine in hardware is costly, requiring large amounts of time because fitness evaluations can take minutes to perform. Furthermore, the nature of the optimisation process itself places significant stress on a humanoid robot. New untested parameters are tried, many of these parameters will result in unstable gaits, causing the robot to fall.

In this work we seek to maximise the performance of an existing walk engine, while attempting to minimise the stress placed on the robot. To achieve this goal, we make extensive use of a simulator [7] to design the walk optimiser, with the aim of applying the optimiser only a single time in hardware.

There are three core components of a walk optimiser; the algorithm, the fitness function and the parameter space. Firstly, we need to determine which algorithm to use, as well as the parameters for that algorithm that best suit the problem of walk optimisation. We compare three different algorithms; Evolutionary Hill Climbing with Line Search (EHCLS) [8], Policy Gradient Reinforcement Learning (PGRL) [9], and Gaussian Particle Swarm Optimisation (GPSO) [10]; each of

which has been meta-optimised to select the most appropriate set of algorithm parameters.

The second component is the fitness function, that is, the mapping from a set of walk parameters to a fitness value reflecting the desirability of the resultant gait. Fitness functions based on speed, efficiency, and the walking Froude-number are compared.

The final component of a walk optimiser is the parameter space over which to optimise. We consider both the standard parameter space, as well as a parameter space expanded to include joint stiffnesses for each motor in the legs [11].

The target application for the walk optimiser is Aldebaran’s NAO. This humanoid robot is used in RoboCup’s Standard Platform League [12], which presents an interesting challenge to any walk engine.

The remainder of this paper, firstly, presents descriptions of the algorithms, fitness functions and parameter spaces compared. The simulation package is then discussed, along with the result of the meta-optimisation of each algorithm. The results of the comparison in the simulator are then presented, and the best combination is selected. Finally, results of the application to the physical NAO are presented.

II. OPTIMISATION ALGORITHMS

A. Evolutionary Hill Climbing with Line Search

The EHCLS algorithm in [8] was used, with modifications such that the limits of the search space for each parameter are used explicitly to determine the size of the mutation in each dimension. The algorithm is as follows:

```

 $\theta_{best} = \theta = \text{InitialParameters}$ 
loop
     $\text{Fitness} = \text{calculateFitness}(\theta)$ 
    if  $\text{Fitness} > \text{Best}$  then
         $\theta_{best} = \theta$ 
         $\text{Improvement} = \text{Fitness} - \text{Best}$ 
         $\alpha = 0.95 \mid \tanh \frac{\text{Improvement}}{\text{PreviousImprovement}}$ 
    end if
    if  $\text{CountSinceImprovement} > \text{ResetLimit}$  then
         $\text{Best} = \text{ResetFraction} \cdot \text{Best}$ 
         $\alpha = 0$ 
    end if
     $\theta = \theta_{best} + \alpha(\theta_{best} - \theta_{previousbest}) + (1 - \alpha)\psi \cdot \text{range}(\theta)$ 
end loop

```

where $\psi = \mathcal{N}(0, \eta \exp(\frac{\text{CountSinceImprovement}}{\text{ResetLimit}} - 1))$ specifies the size of the mutation and $\text{range}(\theta) = \{\theta_i^{\max} - \theta_i^{\min}\}_1^D$ defines the search space for each parameter.

There are three parameters of the EHCLS that can be tuned to suit a particular problem; η controls the size of the mutation at each iteration, ResetLimit specifies the reset frequency and ResetFraction specifies the reset amount.

B. Policy Gradient Reinforcement Learning

The PGRL algorithm in [9] was used with a modified calculation of the adjustment vector A . A sigmoid function was introduced to compress the estimated gradient. This was done to improve the convergence speed of the algorithm and to better handle policies that have a near-zero fitness, which occur frequently when a policy results in the robot falling. The algorithm is as follows:

```

 $\theta = \text{InitialParameters}$ 
loop
   $\{\Theta\}_1^N = \text{generatePolicies}(\theta, \epsilon)$ 
   $\text{Fitness} = \text{calculateFitness}(\{\Theta\}_1^N)$ 
  for  $i = 0$  to  $D$  do
     $\text{Avg}_{+\epsilon}^i \leftarrow$  average fitness for all  $\Theta^k$  that have
      a positive perturbation in dimension  $i$ 
     $\text{Avg}_0^i \leftarrow$  average fitness for all  $\Theta^k$  that have
      no perturbation in dimension  $i$ 
     $\text{Avg}_{-\epsilon}^i \leftarrow$  average fitness for all  $\Theta^k$  that have
      a negative perturbation in dimension  $i$ 
    if  $\text{Avg}_0^i > \text{Avg}_{+\epsilon}^i$  and  $\text{Avg}_0^i > \text{Avg}_{-\epsilon}^i$  then
       $A_i = 0$ 
    else
       $A_i = \eta \tanh(\text{Avg}_{+\epsilon}^i - \text{Avg}_{-\epsilon}^i) \cdot \text{range}(\theta)$ 
    end if
  end for
   $\theta = \theta + A$ 
end loop

```

where η controls the step size, ϵ is the size of the perturbation applied to each dimension in the policy generation stage and N is the number of policies generated near θ at each iteration. Each of these parameters can be meta-optimised.

C. Gaussian Particle Swarm Optimisation

The GPSO algorithm in [10] was used, with an initial swarm randomly distributed over the entire search space. The particle values are restricted to $\text{range}(\theta)$ and the particle velocities to $\pm 0.5\text{range}(\theta)$. The algorithm is as follows:

```

 $\{\Theta\}_1^N = \text{generateParticles}(N)$ 
loop
   $\text{Fitness} = \text{calculateFitness}(\{\Theta\}_1^N)$ 
   $\{pbest\}_1^N \leftarrow$  personal best for each particle
   $gbest \leftarrow$  global best of all particles
   $\{\dot{\Theta}\}_1^N = |\mathcal{N}(0, 1)|(\{pbest\}_1^N - \{\Theta\}_1^N) + |\mathcal{N}(0, 1)|(gbest - \{\Theta\}_1^N)$ 
   $\{\Theta\}_1^N = \{\Theta\}_1^N + \{\dot{\Theta}\}_1^N$ 
end loop

```

where N is the number of particles, and $\mathcal{N}(0, 1)$ is the normal distribution. The parameter N is the only value that can be meta-optimised with this implementation of particle swarm optimisation.

III. FITNESS FUNCTIONS FOR OPTIMISATION

To thoroughly test the performance of a set of walk parameters for an omni-directional walk engine, it is essential to measure the fitness of the parameters over a range of speeds and directions. Furthermore, incorporating changes in direction and start-stop segments, measures the agility of a walk. For these reasons, the optimisation was performed using the path shown in Fig.1, which includes forward, sideward and diagonal components, as well as a difficult omni-directional backward spin. The path is specified to the robot as a sequence of waypoints. The exact path the robot takes to each of the waypoints becomes part of the optimisation problem.

We consider three fitness functions in this paper; one based on the average speed over the path, a second based on the

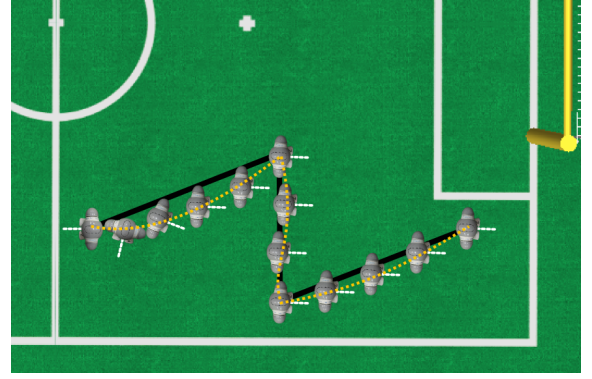


Fig. 1. The path over which the fitness of a set of walk parameters is determined. The black line indicates the path between the way points of the path, the dashed line displays a typical travelled path, and the dashed lines coming from the robot indicate its orientation.

energy used over the path, and a third based on the Froude-number.

A. Speed

The speed-based fitness function, $\mathcal{F}_S$, is given by:

$$\mathcal{F}_S = \begin{cases} \frac{\text{distance}}{\text{time}} \cdot \frac{0.5 \cdot \text{distance}}{D} & \text{if path not completed} \\ \frac{D}{\text{time}} & \text{if path completed} \end{cases}$$

where D is the distance of the path shown in Fig. 1.

For parameters that successfully complete the path, the fitness is simply the average speed. The path distance is used instead of the actual distance travelled, to avoid rewarding the robot for deviating from the path. However, if the path is not completed, the fitness is reduced by a half, to penalise the parameter set for being unstable, and is also scaled by the ratio of the distance travelled and the complete path distance, to penalise parameters that fall over quickly.

B. Efficiency

The cost of transport, c_{et} , is given by:

$$c_{et} = \begin{cases} \frac{\text{energy}}{mgD} \cdot \frac{D}{0.5 \cdot \text{distance}} & \text{if path not completed} \\ \frac{\text{energy}}{mgD} & \text{if path completed} \end{cases}$$

where energy is the total amount of energy consumed completing the path, m is the mass of the robot and D is the path distance. The energy is measured using the voltage and current readings from the battery and motors.

The c_{et} is then used to calculate the cost-based fitness, $\mathcal{F}_C$ using:

$$\mathcal{F}_C = \frac{180}{4 + c_{et}}.$$

This fitness function was chosen to give $\mathcal{F}_C$ similar properties as $\mathcal{F}_S$. In that it has a comparable range of values, and is a maximum for when the efficiency is the highest. This scaling allows the fitness functions to be interchanged without modifications to the optimisation algorithms.

C. Froude–Number

The fitness function based on the Froude–number for bipedal walking [13] is given by:

$$\mathcal{F}_F = 20 \frac{\mathcal{F}_S^2}{g\ell}$$

where ℓ is the height of the torso of the robot and is measured kinematically for each set of parameters. The scaling factor of 20 is used to give this function a similar range to that of $\mathcal{F}_S$ and $\mathcal{F}_C$.

IV. PARAMETER SPACES FOR OPTIMISATION

In addition to optimising the walk parameters, the stiffness, or controller gain, of each joint can also be optimised to improve performance [11], [14]. We vary the joint stiffness such that the values are the same for the left and right leg, however, each of the six degrees of freedom in the leg can differ. When the stiffnesses are added to the walk parameters, the optimisation problem is increased by six dimensions.

V. THE WALK ENGINE AND SIMULATOR

The walk engine selected for optimisation was an omnidirectional walk engine using ZMP preview control [15]. This engine was selected based on its similarity to the walk engine provided with the NAO. The walk parameters for optimisation are shown in Table I.

TABLE I
WALK PARAMETERS FOR NBWALK2010

Parameter	Min	Max
MaxVelocities	[0,0,0]	[70,70,2]
MaxAccelerations	[0,0,0]	[140,140,4]
StepFrequency	1 Hz	5 Hz
StepHeight	0 cm	8 cm
ZMPStatic	0	1
ZMPOffset	0 cm	7 cm
SensorGain	[0,0]	[0.2,0.2]
SensorSpring	[0,0]	[1000,1000]
DSFraction	0.1	0.8
HipHack	0 rad	0.3 rad
FootLift	-0.4 rad	0.4 rad
FootOrientation	-0.2 rad	0.4 rad
TorsoHeight	25 cm	32 cm

The Webots simulation package [7] was used to perform all of the experiments in this paper. The existing RobotStadium environment included with the package was used, however, a customised model that more accurately matches the actual NAO was used.

VI. META-OPTIMISATION OF ALGORITHMS

To adequately compare the algorithms, each needs to be meta-optimised. For the purposes of meta-optimisation the algorithms worked on $\mathcal{F}_S$ with the stiffnesses dimensions. A grid search was employed to select the best parameters for each algorithm. Four to six instances of the simulation at each grid point were run to improve the certainty of the measurement.

The fitness function for the meta-optimisation was the mean speed of the best selected gaits after a fixed expenditure. The

expense (\$) after a given number of iterations, i , and a number of falls, f , is given by

$$\text{expense} = i + 9f.$$

The expense is calculated in this way to reflect the stress placed on the robot hardware by the optimisation process. A fall costs 9 times more than a successful iteration, because it places significantly more stress on the robot. Thus, for a fixed expenditure of \$2000, it is possible to perform between 220 and 2000 iterations depending on the fall rate during the optimisation procedure.

The EHCLS performed best with $\eta = 0.1$, $\text{ResetLimit} = 5$ and $\text{ResetFraction} = 0.995$. The low ResetLimit indicates the algorithm quickly discards the gradient component, when it fails to result in further improvements.

The PGRL performed best with $\eta = 0.05$, $\epsilon = 0.05$ and $N = 12$. It was noted that η and ϵ needed to have comparable values, this prevents the algorithm from overstepping the area under which its gradient estimate is useful.

The GPSO performed best with a swarm size of $N = 30$, slightly larger than the number of dimensions.

VII. COMPARISON OF ALGORITHMS

Each of the three optimisation algorithms were tested with each fitness function, and with both parameter spaces; giving a total of 18 combinations. Each combination was tested 12 times to enable an adequate assessment of the performance, in particular, with respect to the variation in performance between trials. The results of the 216 simulation trials are summarised in Fig. 2.

The quality of the selected gaits after a fixed expenditure is used to assess the performance of each algorithm. Speed, cost of transport, and stability of a gait are the three features considered. Table II shows the mean performance of the gaits selected by each algorithm operating on each of the three fitness functions and two parameter spaces. In this way the comparison of the algorithms is more general, than if only a single fitness function and parameter space had been used. The standard deviation for each quantity is shown in brackets. Fig. 3 shows the progress of each algorithm as a function of expense.

The final column in Table II is the average fall percentage, and is used to assess the stability of a gait. For example, the EHCLS has a fall percentage of 16.3%. This means that on average, the best gaits selected by the EHCLS, have a 16.3% chance of causing the robot to fall while traversing the optimisation path.

TABLE II
PERFORMANCE OF ALGORITHMS

Algorithm	Speed (cm/s)	Efficiency (J/Nm)	Falls (%)
EHCLS	18.2 (0.8)	4.22 (0.31)	16.3
PGRL	18.5 (0.6)	4.07 (0.18)	8.7
GPSO	18.5 (0.8)	4.38 (0.28)	11.9

It is clear that after meta-optimisation each of the three algorithms produce gaits that have similar speeds and efficiencies, which are greatly improved over the initial starting point. The PGRL performs slightly better in this respect than

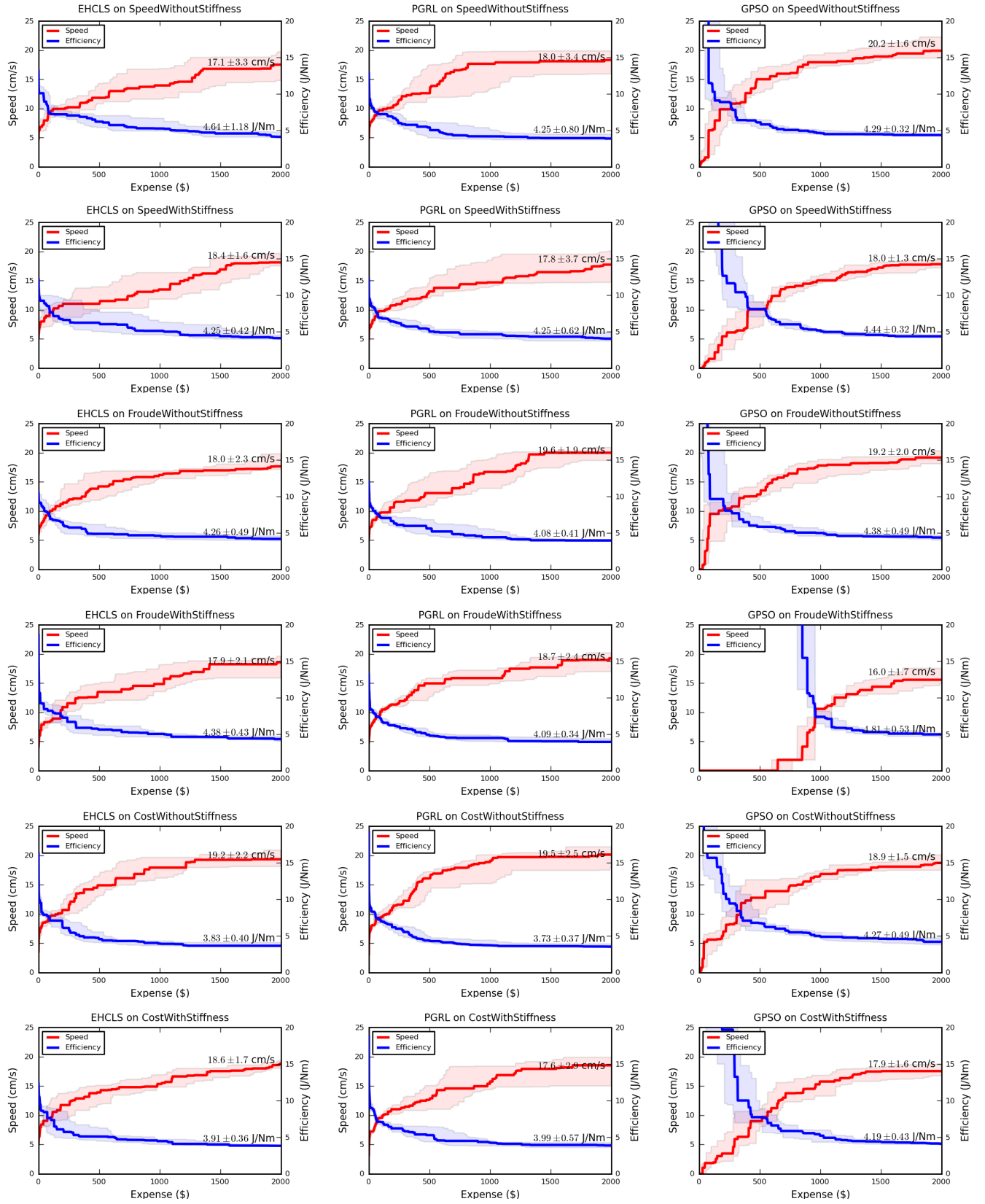


Fig. 2. The results of each optimisation algorithm operating on each of the three fitness functions, and the two parameters spaces, is shown. The thick red lines indicate the median speed, while the two thin red lines are the 25th and 75th percentiles. The thick blue line indicates the median cost of transport, and the two thin lines are the 25th and 75th percentiles.

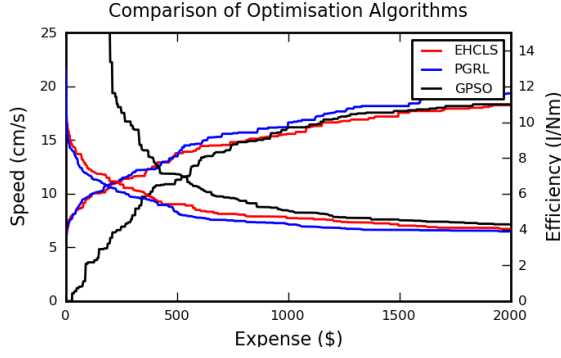


Fig. 3. The speed and efficiency of the three optimisation algorithms tested. Only the median is plotted for each algorithm, and the results for each algorithm have the same colour.

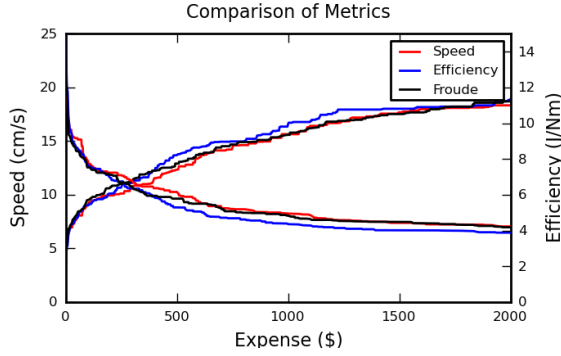


Fig. 4. The speed and efficiency of the each of the three fitness functions tested. Only the median is plotted.

the other two, and also has less variation between the selected gaits. The PGRL also produces gaits that have a significantly lower fall percentage, implying that the gaits selected by PGRL are more stable than those selected by EHCLS or GPSO. This improvement in stability is achieved despite each algorithm using the same implicit stability measure in the fitness functions.

This stability stems from the way the PGRL algorithm works. It takes the average of several nearby policies to estimate the gradient, this results in it avoiding areas where the gaits may be fast, but are either inconsistent, or such that small changes to the walk parameters result in instability. Conversely, an algorithm like the EHCLS has a tendency to explore areas where the gaits are fast, but potentially inconsistent, because it uses only a single trial to evaluate a set of walk parameters.

VIII. COMPARISON OF FITNESS FUNCTIONS

In a similar fashion to the presentation of the results for the comparison of the algorithms, the mean speed, efficiency and fall percentage are shown in Table III for each fitness function. The evolution of the performance as a function of expense is shown in Fig. 4.

TABLE III
PERFORMANCE OF FITNESS FUNCTIONS

Function	Speed (cm/s)	Efficiency (J/Nm)	Falls (%)
Speed	18.2 (0.8)	4.36 (0.34)	15.4
Efficiency	18.6 (0.6)	3.99 (0.19)	8.0
Froude	18.3 (0.8)	4.32 (0.24)	13.7

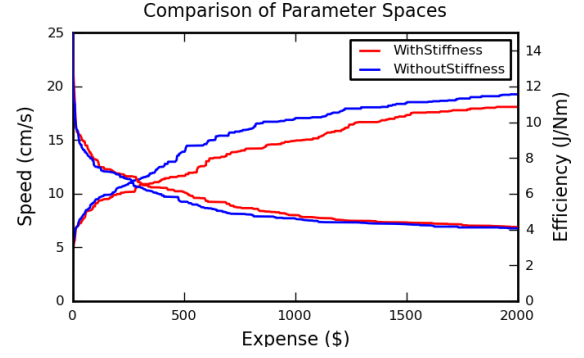


Fig. 5. The average performance of the parameter space with and without the additional stiffness dimensions. The median values of the speed and cost of transport are shown in the same colour for the two spaces.

The results in Table III demonstrate that the cost of transport based fitness function outperforms the other two in every respect. Interestingly, it even outperforms the speed based fitness function at improving the walk speed. The NAO has a high idle energy consumption, so the efficiency can be significantly improved by simply completing the path faster. This property makes optimising the efficiency particularly effective at simultaneously improving the speed. Furthermore, humans also have a high basal energy consumption [16], and there is evidence to support the proposition that human evolution has used a cost of transport based fitness function [17].

The cost of transport based fitness function also selects gaits that are significantly more stable. The gaits are forced to exploit the natural dynamics of the robot to become more efficient. Additionally, the walks become smoother as unnecessary jerk is reduced to conserve energy. These features make the walks inherently more stable.

IX. COMPARISON OF PARAMETER SPACES

The mean speeds, efficiencies, and fall percentages for the parameter spaces with and without stiffness are shown in Table IV, and the evolution of the performance is shown in Fig. 5.

TABLE IV
PERFORMANCE OF PARAMETER SPACES

Stiffness	Speed (cm/s)	Efficiency (J/Nm)	Falls (%)
With	17.9 (0.7)	4.24 (0.29)	9.7
Without	18.8 (0.8)	4.37 (0.22)	15.1

Table IV shows that the gaits produced with stiffness are slower. However, they are more efficient and more stable. Lowering the stiffness in each joint has the effect of both reducing the maximum torque exerted by the motor and reducing the controller gain. This allows the robot to ‘settle’ into a more efficient gait, rather than attempting to rigidly follow joint trajectories produced by an imperfect walk engine.

Focusing on the PGRL algorithm, when the extra stiffness dimensions are added, the mean gait speed drops from 19.2cm/s to 18.2cm/s . However, the mean fall percentage halves from 11.8% to 5.6%. Focusing further still on the PGRL with efficiency based fitness function, the speed decreases from 19.1cm/s to 18.1cm/s , however, the fall percentage reduces from 2.1% to a perfect zero. That is, when using

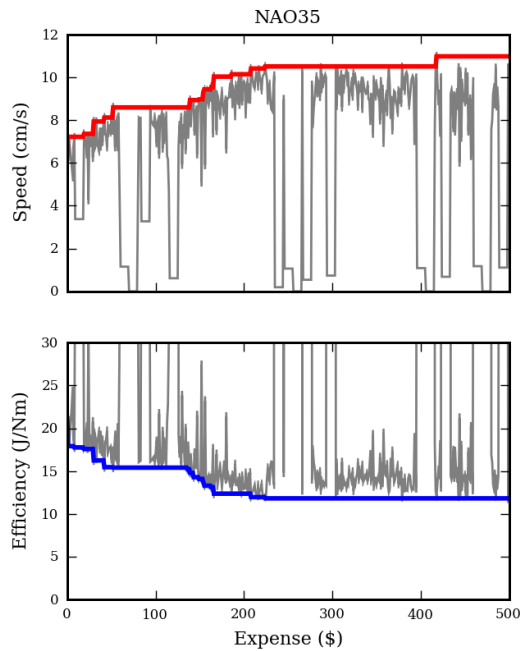


Fig. 6. The results of the PGRL with additional stiffness parameters and efficiency based fitness function on a real NAO. The black line represents the best speed and efficiency, and the grey line indicates the individual speed and efficiency for each set of walk parameters tried.

PGRL with $\mathcal{F}_C$ and the additional stiffness parameters, none of the best selected gaits ever fall.

This result is especially important for the application of this optimisation process to a physical NAO. This indicates that if this configuration is run only a single time, it is highly likely that a gait that never falls over will be selected.

X. APPLICATION TO PHYSICAL NAO

The combination of PGRL with stiffness parameters and efficiency based fitness function was applied to a physical NAO to improve the walk engine that comes with the robot. The walk optimiser developed in the simulator was applied directly, without modification, to the physical NAO. However, the walk engine on the physical NAO, is different from that on the simulated NAO. Consequently, the walk parameters for the physical NAO are slightly different. The results of the optimisation process are shown in Fig. 6.

The optimiser significantly improves the default parameters for Aldebaran's walk engine. The average speed is improved from 7.0cm/s to 11.0cm/s , and the efficiency is improved from 16.8J/Nm to 11.8J/Nm . Furthermore, the selected walk is very stable, having never fallen over after many iterations of the optimisation path.

XI. CONCLUSION

The Policy Gradient Reinforcement Learning algorithm performed slightly better in terms of improving the walk engine's speed and efficiency, and does so more consistently. Furthermore, the PGRL algorithm significantly outperforms the other two, in selecting gaits which are stable.

The cost of transport based fitness function outperformed both the speed and Froude-number based functions in every respect, especially in terms of the selected gait stability. The addition of joint stiffnesses to the parameter space also improved the efficiency and stability of the selected gaits. The results demonstrate that a significant improvement in walk stability can be achieved through careful selection of an algorithm, fitness function, and parameter space.

The best algorithm–fitness–space combination was found to be PGRL with stiffness parameters and an efficiency based fitness function. In the simulator, all of the walks selected by this combination had a zero fall percentage, while being extremely fast and efficient.

The best combination was then applied without modification to a physical NAO. The optimiser successfully improved the walk's speed by 57%, and its efficiency by 30%.

REFERENCES

- [1] M. Vukobratović, B. Borovac, D. Surla, and D. Stokic, *Biped Locomotion: dynamics, stability, control, and application*. Springer-Verlag Berlin, 1990, vol. 1.
- [2] D. Katić and M. Vukobratović, "Survey of intelligent control techniques for humanoid robots," *Journal of Intelligent and Robotics Systems*, vol. 37, no. 2, pp. 117–141, 2003.
- [3] K. Harada, *Motion Planning for Humanoid Robots*. Springer Verlag, 2010.
- [4] D. Gong, J. Yan, and G. Zuo, "A review of gait optimization based on evolutionary computation," *Applied Computational Intelligence and Soft Computing*, 2010.
- [5] S. Chalup, C. Murch, and M. Quinlan, "Machine learning with aibo robots in the four-legged league of robocup," *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, pp. 297–310, 2007.
- [6] J. Zagel and J. R. del Solar, "Combining simulation and reality in evolutionary robotics," *Journal of Intelligent and Robotic Systems*, vol. 59, no. 1, pp. 19–39, 2007.
- [7] (2010) Cyberbotics webots. [Online]. Available: <http://www.cyberbotics.com/products/webots/>
- [8] M. J. Quinlan, "Machine learning on AIBO robots," Ph.D. dissertation, School of Electrical Engineering and Computer Science, University of Newcastle, Australia, 2006.
- [9] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, apr. 2004, pp. 2619 – 2624.
- [10] R. Krohling, "Gaussian swarm: a novel particle swarm optimization algorithm," in *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, vol. 1, 2004, pp. 372 – 376.
- [11] J. Kulk and J. Welsh, "A low power walk for the nao robot," in *Australasian Conference on Robotics and Automation*, 2008.
- [12] (2009) Robocup standard platform league website. [Online]. Available: <http://www.tzi.de/spl/>
- [13] C. L. Vaughan and M. J. O'Malley, "Froude and the contribution of naval architecture to our understanding of bipedal locomotion," *Gait & posture*, vol. 21, no. 3, pp. 350–362, 2005.
- [14] J. Kulk and J. Welsh, "Autonomous optimisation of joint stiffnesses over the entire gait cycle for the nao robot," in *International Symposium on Robotics and Intelligent Sensors*, 2010.
- [15] J. Strom, G. Slavov, and E. Chown, "Omnidirectional walking using zmp and preview control for the nao humanoid robot," in *RoboCup 2009: Robot Soccer World Cup XIII*, ser. Lecture Notes in Computer Science, 2010, vol. 5949, pp. 378–389.
- [16] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.
- [17] F. C. Anderson and M. G. Pandy, "Dynamic optimization of human walking," *Journal of Biomechanical Engineering*, vol. 123, no. 5, pp. 381–390, 2001.