

The initial temperature, the Markov chain length and the cooling temperature rate are important variables in the design of simulated annealing algorithm. The neighbour of the new solution is generated usually by adding a small real random number $[-b, b]$ to each particle in the new solution. The stopping criteria of the algorithm is usually when ΔE is less than a threshold ω or when a minimum temperature is reached or when the algorithm reaches the maximum number of iterations. The Markov chain length varies according to the difficulty of the problem.

B. Genetic Algorithms Overview

Over the past years, the original genetic algorithm (GA) introduced by Holland [10] has evolved significantly in order to suit real-world optimization challenges faced by engineers and scientists. A major alteration is in the development of real coded genetic algorithms (RCGA), using real numbers in their chromosomes rather than traditional bit strings. The major advantages of RCGA over standard binary-coded GA include the maintainability of precision of real numbers usually lost in binary GAs and the reduction in the size of the chromosome, reducing the computation cost. In addition, real-valued encoding also provides a natural way of approximating real world applications. The computational and optimisation power of RCGA has also been demonstrated [11], [12].

The basic idea used in RCGA optimization is as follows. Initially, a number of possible random solutions (chromosomes) make up a population. Over time, the GA evaluates each chromosome according to its fitness and employs genetic operators like *selection*, *crossover* and *mutation* for producing new solutions. These new solutions are called *offsprings*, which are added into the new population. The process is repeated until the algorithm obtains the best solution. The selection of parent chromosomes from the population is done using the selection operator. In the standard procedure, usually two or more parents are selected using the selection operator from a population to make single child or multiple children. The choice of the appropriate genetic operator is important as it directly influences the convergence of the GA.

C. Hybrid Meta-heuristic Techniques

Hybrid meta-heuristic [13], Hyper-heuristic [14] and Memetic algorithms [15] refer to the group of hybrid algorithms, where two or more meta-heuristic search techniques are combined to solve difficult problems. They provide intensification and diversification during the search process for obtaining a stronger solution. An example of the hybrid meta-heuristic approach is the *CoSearch* method [16], which uses the combination of genetic algorithms, Tabu search and local search with an adaptive memory that contains history of the search already done.

Hybrid meta-heuristic algorithms are grouped into two major categories. In the first category, two or three heuristic search algorithms are crossed over into a hybrid parallel search paradigm to provide intensification and diversification in search. Examples include the combination of genetic

algorithm with Tabu search [17] and simulated annealing [18]. In the second category, a meta-heuristic solution is used to guide or provide initial search for another algorithm [19].

The Hybrid GA-SA approach combines both algorithms in parallel where the GA uses the SA for a predefined number of iterations during the GA search process. In the past, the development of hybrid genetic algorithm and simulated annealing has been implemented using parallel processing machine where simulated annealing is employed with lower temperatures after the use of genetic recombination operators such as crossover and mutation over the whole population[20].

In this paper, we use the hybrid approach where the GA uses the SA after the crossover operation (hybrid GA-SA). In this approach, the SA replaces the mutation operator. The hybrid paradigm is shown in Algorithm 1. Note that the number of iterations (N) for simulated annealing depends on the nature of the problem.

Alg. 1 Hybrid GA-SA.

```

Initialize Population (P) of P solutions
while !termination do
  while  $i < P.size()$  do
    1) Evaluate fitness
    2) Selection of Parent 1 and Parent 2
    3) Employ Crossover and produce a single Child
    4) Present the Child to SA for  $N$  Iter.
    5) Copy the updated Child given by SA into population
    6) Lower the initial temperature T
    7) increment  $i$ 
  end while
  Update population
end while
Get the best solution

```

The inner loop of Algorithm 1 beings by calculating the fitness of each solution in the population. Furthermore, the selection operator chooses two parents depending on the selection criteria. In this implementation, the roulette wheel selection is used. The crossover operator combines both parents and produces a *single* child. The child is presented to the simulated annealing algorithm with a specified initial temperature for N iterations. Afterwards, the refined child from the simulated annealing algorithm is copied back to the new population.

The simulated annealing process is dependent of a predefined probability. Therefore, there will be cases where the procedure will not be applied to some members of the population. This implies that weaker solutions are sometimes retained in the future populations as they may contain useful properties for future convergence.

Wright's heuristic crossover operator has shown superior performance in comparison with other crossover operators for a set of optimization problems [21], so it is used here. The Wright's heuristic operator produces a single offspring given two parents. For a pair of parents $x^1 = (x_1^1, x_2^1, x_3^1, \dots, x_n^1)$

and $x^2 = (x_1^2, x_2^2, x_3^2, \dots, x_n^2)$, an offspring is produced as shown in Equation(1):

$$y_i = r(x_i^1 - x_i^2) + x_i^1 \quad (1)$$

where r is a real random number belonging to $[0,1]$ and x^1 is the parent with the best fitness. Usually, a new offspring is produced with a new r until a chromosome with a better fitness is created.

The *Pivot Mutation* operator introduced in [22] selects a pivot point in the chromosome and all the genes after the selected pivot are mutated by adding different small real-random numbers, respectively. For instance, given a chromosome $x = (x_1, x_2, x_3, \dots, x_n)$, the resulting pivoted chromosome becomes $y = (x_1, x_2, y_3, \dots, y_n)$, where $y_i = x_i + r$ given that r is a small real random number in the interval $[a, b]$, where a and b are small negative and positive real numbers chosen by the user, respectively.

In [23], this hybrid method was used for the forward kinematics of the 6-6 general parallel manipulator. In this paper, this method will be used for the forward kinematics of 3RPR planar parallel manipulator.

III. FORWARD KINEMATICS OF PARALLEL MANIPULATORS

The idea to design planar parallel mechanisms can be traced back as early as in the beginning of the 1940s with Pollard and his *five-bar mechanism*¹. More generally, parallel mechanisms have then attracted much attention since they have been successfully applied as flight simulators, [24]. Even though they were extensively studied, the impact of planar parallel manipulators in the industry remains limited.

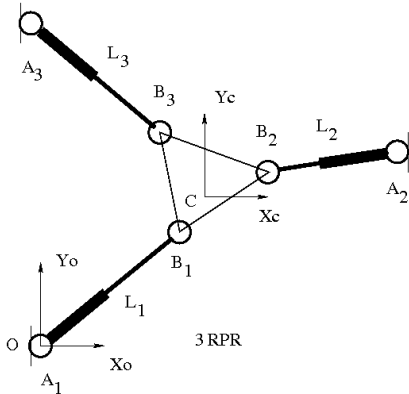


Fig. 1. The general planar manipulator and the typical **3-RPR** tripod [6].

Typically, planar parallel manipulators are characterized by one base, one mobile platform and three kinematics chains which lie in one plane, namely the **3-RPR** [25]. Moreover, the manipulator end-effector displacements are restricted to that same plane as shown in Figure(1). The design hypotheses states that the bodies are infinitely rigid and the joints do not yield friction nor play. The redundant

¹W-L-G. Pollard, Spray painting machine, August 26, 1940, USA Patent no. 2,213,108, Evanston, ILL, USA

manipulators shall not be studied in this article. Moreover, the number of actuated and measured joint variables equals the number of end-effector *degrees-of-freedom* (DOF).

A. The kinematics of parallel manipulators

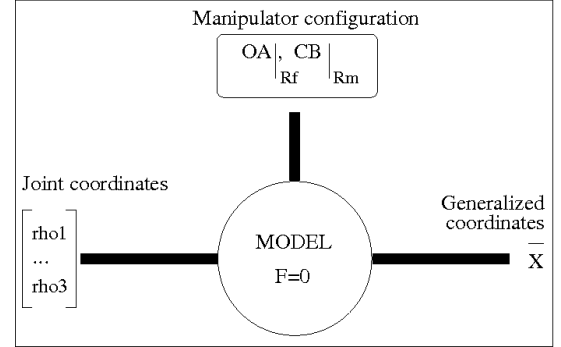


Fig. 2. Kinematics model [6].

Any manipulator is characterized by its mechanical configuration parameters and the posture variables. The configuration parameters are thus $\mathbf{OA}_{|R_f}$, the base attachment point coordinates in R_f (the base reference frame), and $\mathbf{CB}_{|R_m}$, the mobile platform attachment point coordinates in R_m (the mobile platform reference frame). The kinematics model variables are the joint coordinates and end-effector generalized coordinates. The joint variables are described as l_i , the prismatic joint or linear actuator positions. The generalized coordinates are expressed as \vec{X} , the end-effector position and orientation.

The kinematics model is an implicit relation between the configuration parameters and the posture variables, $F(\vec{X}, \vec{\rho}, \mathbf{OA}_{|R_f}, \mathbf{CB}_{|R_m}) = 0$ where $\vec{L} = \{l_1, l_2, l_3\}$.

This article shall only concentrate on the forward kinematics problem (**FKP**) as shown in Figure(2). Usually the *inverse kinematics problem* is required to model the **FKP** and is defined as: *given the generalized coordinates of the manipulator end-effector, find the joint positions*.

Accordingly, the *forward kinematics problem* is defined as: *given the joint positions, find the generalized coordinates of the manipulator end-effector*.

In the majority of parallel manipulator cases, the **FKP** is a difficult problem, [26].

B. Vectorial formulation of the basic kinematics model

Containing as many equations as variables, vectorial formulation constructs an equation system [27], Figure(3), as a closed vector cycle between the following points: A_i and B_i , kinematics chain attachment points, O the fixed base reference frame and C the mobile platform reference frame. For each kinematics chain, an implicit function $\vec{A_i B_i} = U_1(X)$ can be written between joint positions A_i and B_i . Each vector $\vec{A_i B_i}$ is expressed knowing the joint coordinates $\vec{L_i}$ and X giving function $U_2(X, \vec{L_i})$. The following equality has to be solved: $U_1(X) = U_2(X, \vec{L_i})$. The distance

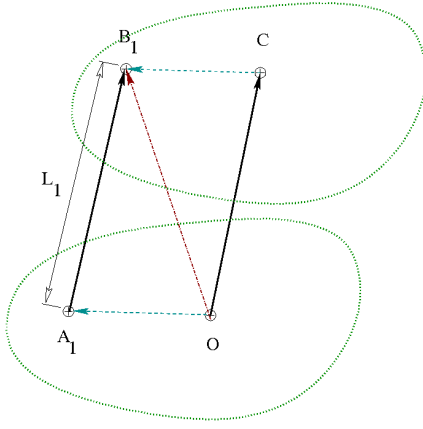


Fig. 3. The vectorial formulation for one kinematics chain [6].

between A_i and B_i is set to L_i . Therefore, the end-effector position X or C can be derived by one platform displacement \overrightarrow{OC} and then one platform general rotation expressed by the rotation matrix \mathcal{R} . Vectorial formulation in Equation (4) evolves as a displacement based equation system using the relation given in Equation (2).

$$\overrightarrow{A_i B_i} = \overrightarrow{OC} + \mathcal{R} \overrightarrow{C B_i} - \overrightarrow{O A_i} \quad (2)$$

As given in Equation(2), for this planar manipulator, the rotation matrix \mathcal{R} is expressed in Equation (3).

$$\mathcal{R} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \text{labelrotategg} \quad (3)$$

$$L_i^2 = \|\overrightarrow{A_i B_i}\|^2 \quad (4)$$

For each distinct platform point $\overrightarrow{O B_i}|_{R_f}$ with $i = 1, \dots, 3$, each kinematics chain can be expressed using the distance norm constraint [28].

C. The inverse kinematics problem

A review of planar parallel manipulators shows that the majority of the proposals fall into the four following classes: **3-RPR**, **3-RRP**, **3-PRR** and **3-RRR**, [6]. We shall only study the **3-RPR**. For each kinematics chain, the **RPR** manipulator is constituted by a prismatic actuator located between two ball joints fixed on the base and the platform. Lets have $L_i = l_i$. Taking Equation (2) and substituting it in Equation (4), one obtains the following equation:

$$l_i^2 = \|\overrightarrow{OC} + \mathcal{R} \overrightarrow{C B_i} - \overrightarrow{O A_i}\|^2, \quad i = 1 \dots 3 \quad (5)$$

The rotation matrix \mathcal{R} is expressed in terms of one rotation parameter.

D. The Forward Kinematics and Conversion to Optimization Problem

The **IKP** expression gives an algebraic system comprising the first three equations in terms of the classical variables : x_c, y_c, θ .

Being only applicable to optimization problems, the simulated annealing and genetic algorithm handles a cost function to be maximized or minimized, respectively. Therefore, we need to effectively convert the problem which solves a system of equations into an optimization problem. Hence, only the inverse kinematic model is required from which we can easily derive a cost function, also called the fitness function. The fitness function represents the total error on each leg lengths. Let lg_i be the leg length of kinematics chain i which is given as input of the problem. Therefore, the fitness function is set to:

$$\sum_{i=1}^3 (l_i - lg_i)^2 \quad (6)$$

IV. RESULTS AND ANALYSIS

In this section, one example of the FKP resolution on a typical 3-RPR manipulator configuration is examined. The manipulator base coordinates of the joint center position $O A_i|_{R_f}$ in the base reference frame R_f and the mobile platform coordinates of the joint center position $C B_i|_{R_m}$ in the platform reference frame R_m , and the minimum bar lengths are shown in Table(I).

TABLE I

PLANAR PARALLEL MANIPULATOR CONFIGURATION TABLE

Config	$A_1(x)$	$A_1(y)$	$A_2(x)$	$A_2(y)$	$A_3(x)$	$A_3(y)$
3RPR8	0	0	200	0	0	200
Id	$B_1(x)$	$B_1(y)$	$B_2(x)$	$B_2(y)$	$B_3(x)$	$B_3(y)$
3RPR8	0	0	50	0	40	40

The joint variables are the kinematics chain lengths. The following leg lengths were used, $L := [100, 120, 150]$.

A. Experimental Setup

In this section, the performance of simulated annealing, genetic algorithms and hybrid GA-SA is evaluated. The fitness function derived from the inverse kinematics of tripod 3-RPR parallel manipulator as shown in Equation (6) is used. In all experiments, for population based methods, the roulette wheel selection is used in conjunction with the elitist strategy method. Note that if the population size is P , then $2P$ selections are done in order to make P offspring for the new population. The following meta-heuristic strategies were used in order to find the best heuristic method for the problem. Note that the respective mutation and crossover rates were determined in trial experimental runs.

- 1) **SA**: The simulated annealing algorithm with initial temperature of 100. The Markov chain lengths of 20, 50 and 100, and cooling rate of 0.01 is used.
- 2) **GA(P)**: The real-coded genetic algorithm with Wright's heuristic crossover operator and pivot mutation. The crossover operator is used at a rate of 0.9 while the mutation rate is 0.1.
- 3) **GA(N-U)**: The real-coded genetic algorithm used in [1] with Wright's heuristic crossover operator and non-uniform mutation. The crossover operator is used at a rate of 0.9 while the mutation rate is 0.1.

TABLE II
HYBRID META-HEURISTIC PARADIGMS FOR THE FKP OF 3RPR MANIPULATOR.

Method	Markov Chain	No. Itera	Fitness	CPU Time(S)	Success/50
SA	20	1130± 31	6.55E-5 ±7.34E-6	0.14 ± 0.09	50
SA	50	1066± 25	6.13E-5 ±7.47E-6	0.30 ± 0.12	50
SA	100	1073± 28	5.07E-5 ±7.36E-6	0.54 ± 0.13	50
GA (P)	-	302±72	0.0001±0.00007	16.86±3.99	50
GA (N-U)	-	863±70	0.010±0.011	48.24±4.27	45
Hybrid GA-SA	20	18±3	6.13E-5±7.13E-6	13.78±3.01	50

4) **Hybrid GA-SA:** The genetic algorithm uses the Wright's heuristic crossover operator at a rate of 0.9 to build a single child from two parents. The simulated annealing algorithm uses an initial temperature of 10 and N of 500 iterations. The Markov chain length of 20 and the cooling rate of 0.01 was used. The SA is used as a genetic operator and applied at the rate of 0.9 as shown in Algorithm(1).

All experiments initialize the population with real numbers in the range of $[-10, 10]$. The population size of 40 and chromosome size of 3, represents the position's x , y and θ . For each method, a total of 50 experimental runs were done. The search process halted when the fitness value gets lower than 0.0001. The experiments were performed on an IBM compatible personal computer with 1.74 GHz dual core processors with Linux operating system. The computation times are given in seconds. They were calculated using the standard C++ built in functions.

B. Results

The value for N used in the SA process of Hybrid GA-SA was determined in trial experimental runs with 100, 200, 300, 400 and 500. The best performance was shown when N was 500, therefore, this value was used in all experiments. The SA used the Monte Carlo length of 20.

Table 2 shows the mean and 95 percent confidence interval for 50 experimental runs for each meta-heuristic paradigm. The optimization results for coordinates x , y , and orientation θ with its respective fitness is given in Table III while the solutions obtained in earlier work using the algebraic method is given in Table IV. The given forward kinematics problem has 2 unique solutions. The optimization time is given by the number of iterations (No. Itera) and CPU Time in seconds taken by the respective paradigm in converging to the required solution accuracy. The success rate shows how well the given paradigm can guarantee a solution when given any initial random solution within the search space.

C. Solution Given by Algebraic Method Used for Comparison

In order to verify the solutions obtained by the respective methods, we revert to an algebraic method which can compute the certified exact results, [6]. For the aforementioned planar parallel manipulator, the exact results are shown in Table IV.

TABLE III

OPTIMAL x , y AND θ VALUES FROM THE BEST EXPERIMENTAL RUN FOR EACH METHOD

Method	Solution	x	y	θ (deg)	Fitness
SA	1	52.8582	84.8841	-33.4618	1.39E-05
	2	97.9989	19.9168	85.0198	2.17E-05
GA	1	52.8614	84.8901	-33.4616	1.63E-05
	2	97.9919	19.9123	85.0191	5.16E-05
Hybrid	1	52.8587	84.8884	-33.4629	1.26E-05
	2	97.9974	19.9146	85.0217	3.09E-05

TABLE IV
EXACT SOLUTIONS

	x	y	θ
Solution 1	52.8654	84.9536	-33.4487
Solution 2	97.9911	19.9193	85.0154

D. Discussion

In the case of optimization using genetic algorithms, the results in Table II show that Wright's heuristic crossover operator combined with the introduced Pivot mutation provides the best result in terms of least training time and fitness, respectively. However, the simulated annealing algorithm has shown the best performance in terms of CPU computation time when compared to genetic algorithms(GA) and hybrid GA-SA. Note that in SA, the choice of the Monte Carlo length for simulated annealing also affects the CPU computation time.

Table III reveals the solutions obtained by the respective meta-heuristic approaches are relatively similar to the exact solution given by the algebraic method as shown in Table IV. The Hybrid GA-SA delivered the solutions to similar accuracy of SA, however, the hybrid method used more CPU computation time when compared to SA. This is because the hybrid method required extra time in evaluating the population of solutions. Note that the Hybrid method delivered better solution in terms of accuracy and CPU time when compared to standard genetic algorithm in Table II. The solutions given by the exact Algebraic method in Table IV are similar to that of the meta-heuristic methods given in Table III. Therefore, we note that our results are feasible.

We also note that all the meta-heuristic approaches were successful in finding the two distinct solutions in multiple runs. This was done by running multiple experiments with different initial positions in the search space. In this way, the heuristic algorithm converges towards the solution nearest to

the initial search position. For this reason, they have their importance in solving the forward kinematics of parallel manipulators in terms of assembly mode analysis for example. Standard techniques such as gradient descent and Newton's Method fail to provide multiple solutions and are also prone to convergence in a local minimum when given with a initial search position that is distant from the solution. The two solutions reported in this work were not found in the same population. They were obtained from different populations given by multiple experimental runs.

Note that the same paradigm was used in [23] for solving the forward kinematics of the 6-6 general parallel manipulator. The results showed that although the simulated annealing was faster than the Hybrid GA-SA, the Hybrid GA-SA provided better success rate than simulated annealing and genetic algorithm. This is different from the results obtained here as the 6-6 general parallel manipulator problem is a more difficult optimisation problem than the 3RPR.

V. CONCLUSIONS

The results demonstrate that the simulated annealing algorithm has provided a better solution in terms of fitness accuracy and CPU computation time when compared to genetic algorithm. Although the simulated annealing algorithm works with a single solution, it has performed the same task in a fraction of time taken by the genetic algorithm. Furthermore, the hybrid GA-SA has also shown an improved performance when compared to the standard genetic algorithm. This work has also shown that meta-heuristic algorithms, either single solution or population based, are able to provide distinct solutions given multiple trial runs with different initial search positions. This is advantageous to optimization problems where more than one distinct solution is present.

Meta-heuristic paradigms are easier to implement and is independent of the problem domain. The use of hybrid meta-heuristic paradigm is promising for optimization problems in areas of robotics in general. In future work, it may be useful to use hybrid meta-heuristic paradigms in solving other problems such calibration and manipulator design.

REFERENCES

- [1] R. Boudreau and N. Turkkkan, "Solving the forward kinematics of parallel manipulators with a genetic algorithm," *Journal of Robotics Systems*, vol. 13, no. 2, pp. 111–125, 1995.
- [2] T. J. Ypma, "Historical development of the newton-raphson method," *SIAM Rev.*, vol. 37, no. 4, pp. 531–551, 1995.
- [3] D. S. Boudreau, R. and N. Turkkkan, "Etude comparative de trois nouvelles approches pour la solution du problème géométrique direct des manipulateurs parallèles," *Mechanism and Machine Theory*, vol. 33, no. 5, pp. 463–477, 1998.
- [4] A. Omran, G. El-Bayiumi, M. Bayoumi, and A. Kassem, "Genetic algorithm based optimal control for a 6-dof non redundant stewart manipulator," *International Journal of Mechanical Systems Science and Engineering*, vol. 2, no. 2, pp. 73–79, 2008.
- [5] H. M. Wang X. and Y. Cheng, "On the use of differential evolution for forward kinematics of parallel manipulators," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 760–769, 2008.
- [6] L. Rolland, "Synthesis on the forward kinematics problem algebraic modeling for the planar parallel manipulator," *Advanced Robotics*, vol. 20, no. 9, pp. 1035–1065, 2006.

- [7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.
- [8] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, January 1985.
- [9] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.
- [10] J. H. Holland, *Adaptation in natural and artificial systems*. Cambridge, MA, USA: MIT Press, 1992.
- [11] D. E. Goldberg, "Real-coded genetic algorithms, virtual alphabets, and blocking," *Complex Systems*, vol. 5, pp. 139–167, 1991.
- [12] N. J. Radcliffe, "Equivalence class analysis of genetic algorithms," *Complex Systems*, vol. 5, pp. 183–205, 1991.
- [13] M. Lozano and C. Garca-Martnez, "Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report," *Computers and Operations Research*, vol. In Press.
- [14] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.
- [15] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Transactions on Systems Man and Cybernetics – Part B*, vol. 36, no. 1, pp. 141–152, 2006.
- [16] E.-G. Talbi and V. Bachelet, "Cosearch: A parallel cooperative meta-heuristic," *Journal of Mathematical Modelling and Algorithms*, vol. 5, no. 1, pp. 5–22, 2006.
- [17] F. Glover, J. P. Kelly, and M. Laguna, "Genetic algorithms and tabu search: hybrids for optimization," *Comput. Oper. Res.*, vol. 22, no. 1, pp. 111–134, 1995.
- [18] H. Kim, K. Nara, and M. Gen, "A method for maintenance scheduling using ga combined with sa," *Comput. Ind. Eng.*, vol. 27, no. 1-4, pp. 477–480, 1994.
- [19] R. Chandra and C. W. Omlin, "The comparison and combination of genetic and gradient descent learning in recurrent neural networks: An application to speech phoneme classification," in *Proc. of International Conference on Artificial Intelligence and Pattern Recognition*, 2007, pp. 286–293.
- [20] S. W. Mahfoud and D. E. Goldberg, "Parallel recombinative simulated annealing: a genetic algorithm," *Parallel Comput.*, vol. 21, no. 1, pp. 1–28, 1995.
- [21] A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991, pp. 205–218.
- [22] L. Rolland and R. Chandra, "Forward kinematics of the 6-6 general parallel manipulator using real coded genetic algorithms," in *Proc. of IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2009)*, 2009, p. In Press.
- [23] R. Chandra, M. Frean, and L. Rolland, "Hybrid meta-heuristic paradigm for the forward kinematics of the 6-6 general parallel manipulator," in *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2009, p. In Press.
- [24] D. A. Stewart, "Platform with 6 degrees of freedom," *Proceedings of IMechE*, vol. 180, no. 9, pp. 371–386, 1965.
- [25] C. Gosselin and J. P. Merlet, "The direct kinematics of planar parallel manipulators : special architectures and umber of solutions," *Mechanism and Machine Theory*, vol. 29, pp. 1083–1097, 1994.
- [26] M. Raghavan and B. Roth, "Solving polynomial systems for the kinematic analysis and synthesis of mechanisms and robot manipulators," *Trans. ASME*, vol. 117, pp. 71–79, 1995.
- [27] D. E. Parrish, R. and B. R., "An actuator extension transformation for a motion simulator and an inverse transformation applying newton-raphson's method," D-7067 NASA, Tech. Rep., 1972.
- [28] J. P. Merlet, *Les Robots Parall'les*. Hermès, 1997.