Simultaneous Assembly Planning and Assembly System Design Using Multi-objective Genetic Algorithms

Karim Hamza[1], Juan F. Reyes-Luna[1], and Kazuhiro Saitou[2] *

[1] Graduate Student, [2]Assistant Professor, Mechanical Engineering Department
University of Michigan, Ann Arbor, MI 48109-2102, USA
{khamza,juanfr,kazu}@.umich.edu

Abstract. This paper aims to demonstrate the application of multi-objective evolutionary optimization, namely an adaptation of NSGA-II, to simultaneously optimize the assembly sequence plan as well as selection of the type and number of assembly stations for a production shop that produces three different models of wind propelled ventilators. The decision variables, which are the assembly sequences of each product and the machine selection at each assembly station, are encoded in a manner that allows efficient implementation of a repair operator to maintain the feasibility of the offspring. Test runs are conducted for the sample assembly system using a crossover operator tailored for the proposed encoding and some conventional crossover schemes. The results show overall good performance for all schemes with the best performance achieved by the tailored crossover, which illustrates the applicability of multi-objective GA's. The presented framework proposed is generic to be applicable to other products and assembly systems.

1 Introduction

The optimization of product assembly processes is a key issue for the efficiency of manufacturing system, which involves several different types of decisions such as selecting the assembly sequences of products, assigning tasks to the assembly stations, and selecting the number and type of machines at each assembly station.

Research on assembly sequence planning was originated by two pioneer works in late eighties: De Fazio and Whitney [1] and de Mello *et al.* [2] independently presented graph-based models of assemblies and algorithms for enumerating all feasible assembly sequences. Since then, numerous work has been conducted on assembly sequence planning[1]. However, a few attentions have been paid to the integration of assembly sequence planning and assembly system design.

Assembly system design is a complex problem that may have several objectives such as minimizing overall cost, meeting production demand, increasing productivity, reliability and/or product quality. Assembly sequence planning is a precedence-

* Corresponding Author

[1] A comprehensive bibliography of the area is found at www.cs.albany.edu/~amit/bib/ASSPLAN.html

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 2096–2108, 2003.

constrained scheduling problem, which is known to be NP-complete [3]. Furthermore, allocating machines to assembly tasks is resource constrained scheduling which is also known to be NP-complete [3]. In real-life workshops, there is a need to consider assembly sequence and machine allocation simultaneously which results in a doubled difficulty that makes such problems beyond the feasibility of full enumeration. Thus, assembly systems design provides rich opportunities for heuristics approaches such multi-objective GA's. For instance, process planning using GA's was considered in the late eighties [4]. More recently, Awdah *et al.* [5] proposed a computer-aided process-planning model based on GA's. Kaeschel *et al.* [6] applied an evolutionary algorithm to shop floor scheduling and multi-objective evolutionary optimization of flexible manufacturing systems was considered by Chen and Ho [7]. Saitou *et al.*[8] applied GA for robust optimization of multi-product manufacturing systems subject to production plan variations.

This paper presents the application of multi-objective GA's to simultaneously optimize the assembly sequence, assembly stations' type and number selection, based on data extracted from a real assembly shop that assembles specially designed wind-propelled ventilators (Fig. 1). The reminder of the paper first describes the problem formulation, a special encoding and crossover schemes, the results of simulation runs with the proposed crossover scheme as well as arithmetic projection, multipoint and uniform crossovers. Finally, discussion and future extensions are provided.

2 Wind Propelled Ventilators

The family of products considered is the three models of wind propelled ventilators. Shown in Fig. 1, is a photo of model A, the basic model used in ventilating industrial or storage hangars in dry regions. The basic idea of operation is that the ventilators are placed atop ventilating ducts in the ceiling of the hangars. When the lateral wind blows across the hangar, it spins the spherically shaped blades, which in turn perform a sucking action that draws air from the top of the hangar. Model B has the same exoskeleton and blades as model A, but has different internal shaft as well as an additional rotor that improves the air suction out of the ventilation duct. Model C is the same as Model B, except that its outer blades have improved shape design. The three models share several components, and are assembled in the same assembly shop and may use the same assembly stations. Table 1 provides a listing of all components in the three models and in which models they are being used.

The problem of designing an assembly system that can assemble the three types of ventilators, models A, B and C, is formulated as a multi-objective optimization problem with two objective functions to me minimized: assembly cost f_1 and production shortage within a given production period f_2. These are the functions for the three decision variable categories: the assembly sequences of each products, the type of machines at each assembly station, and the number of machines for each type.

The model of the assembly system used in this study is simple one, which computes the production cost f_1 by summing over the startup and hourly operation rate of

the assembly stations. Production volume is estimated according to average cycle times, for the numbers and types of machines assigned to each assembly task:

Minimize
$$f_1 = \sum_{\text{Machines}} \text{Investment Cost} + \sum_{\text{Machines}} \text{Processing Time} \times \text{Hourly Rate} \quad (1)$$
$$f_2 = \sum_{\text{Products}} \max(0,\ \text{Target Production} - \text{Production Volume}) \quad (2)$$

Decision Var. Assembly Sequence, Type of assembly stations, Number of assembly stations of every used type

Fig. 1. A wind-propelled ventilator – Model A

While the simulations provided in this paper are based on the real assembly shop data in a company, the actual production volumes and assembly station rates are not revealed due to proprietary nature of the information.

Table 1. Listing of components in three models of the wind-propelled ventilator.

Component Label	Description	Used in Model		
		A	B	C
1	Duct and Main Body	x	x	x
2	Lower Bracket – Type #1	x		
3	Upper Bracket	x	x	x
4	Axle	x		
5	Lower Ball Bearing – Type #1	x		
6	Upper Ball Bearing	x	x	x
7	Lower Hub – Type #1	x		
8	Upper Hub – Type #1	x		
9	Blades – Type #1	x	x	
10	Lower Bracket – Type #2		x	x
11	Shaft		x	x
12	Lower Ball Bearing – Type #2		x	
13	Lower Hub – Type #2		x	x
14	Upper Hub – Type #2		x	x
15	Inner Rotor		x	x
16	Blades Type #2			x

3 Problem Formulation for Multi-objective GA

A software system, ASMGA (ASsembly using Multi-objective GA) was implemented, whose overall structure is illustrated in Fig. 2. It is capable of communicating with user-provided models of assembly systems involving multi-products.

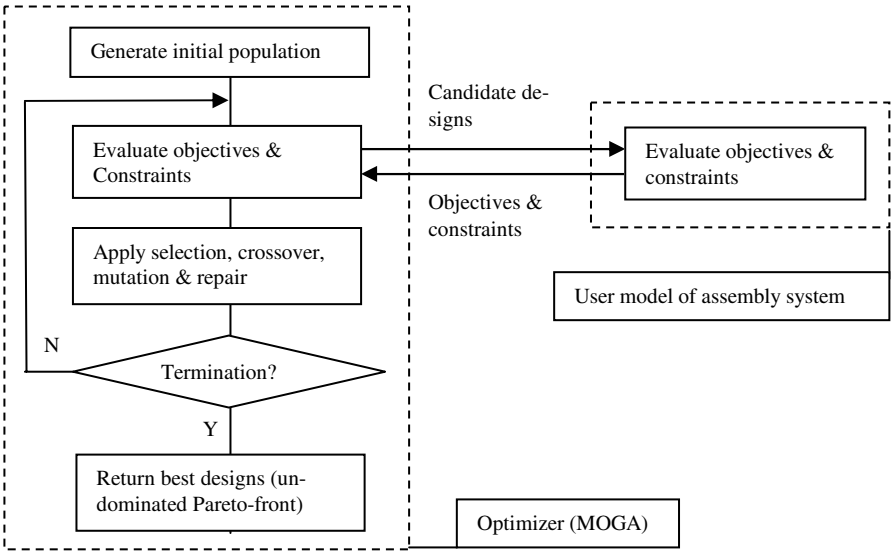

Fig. 2. Overall structure of the ASMGA (ASsembly using Multi-objective GA).

3.1 Encoding Scheme of Chromosomes

The following three decision variable *sets* are considered:

- Assembly sequence of every product;
- Choice of type of assembly station for every assembly operation;
- Number of assembly stations of every type.

For every product consisting of N components, there are $2(N-1)$ variables that control the assembly sequence, $(N-1)$ variables controlling the type of assembly stations and $(N-1)$ variables to specify the number of each type of assembly stations in the platform. In the rest of the paper, a set of values for the decision variables is sometimes referred to as a *candidate design* of the assembly system.

The above decision variables can be efficiently encoded in a fixed-length string of integers in a manner that allows efficient implementation of a repair operator to maintain the feasibility of the offspring for assembly precedence constraints. The layout of the integer string chromosome is shown in Fig. 3. The basic building block of the chromosome is a set of *four integer numbers* that define *one assembly operation*. Every product is assembled through a set of assembly operations. Performing

assembly operations to obtain an assembled product out of its individual components may be mentally visualized by thinking of a hypothetical bin, which initially contains all separate components of the product. In every assembly operation, one reaches into the bin, grabs two items assembles them together (provided they can be assembled together) and puts them back into the hypothetical bin. It can be shown that one gets a product consisting of *N* components, in this manner, in exactly *N*-1 assembly operations. These *N-1* assembly operations are laid on a linear chromosome.

The four-integer assembly operation information translates as follows: The first number refers to one of the components (or sub-assemblies) inside the bin of the product. The second number refers to the component in the bin, which will be joined to the one selected by the first number. The third number refers to the assembly station chosen to perform the operation and the fourth number refers to the number of those assembly stations that should be present in the assembly shop. For example, chromosome *c*=11421021 for the simple product shown in Fig. 4 would be translated as follows:

- In the first assembly operation (first four numbers) the component, which has the order of 1 (*i.e,*. the component labeled 2, because ordering starts at zero) is selected.
- The component labeled 2 is assembled to the second component that can be coupled to it (*i.e.*, the component labeled 3) to produce a subassembly (2,3), using the fifth type of assembly station that can perform the task and there are two such stations in the assembly plant
- In the second assembly operation, according to the new order of the components and subassemblies in the bin, the subassembly (2,3) is selected.
- The subassembly (2,3) is assembled the first (and only) component that can be coupled to it (*i.e*., the component labeled 1) to produce the full product (1,2,3), using the second type of assembly station that can perform the task and there is one such station in the assembly plan

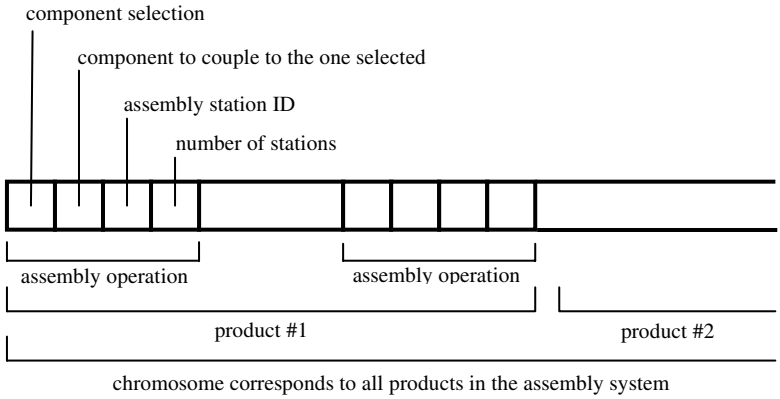

Fig. 3. Layout of chromosomes.

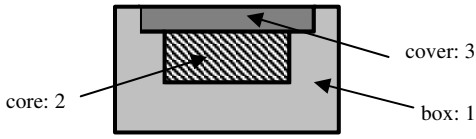

Fig. 4. A simple product.

3.2 Assembly Constraints and Repair Operator

Due to the geometrical and other quality requirements, assembly precedence relationships exist as constraints upon the assembly sequence in almost every real product. Thus it makes sense to define all the assembly constraints during a *pre-processing* stage, and then make (or attempt to make) all candidate designs of the assembly system conform to those constraints throughout the optimization. The ASMGA software includes a pre-processor that guides the user to define all feasible assembly operations and automatically generates a set of all sub-assemblies that may be encountered.

Chromosome reproduction through crossover and mutation may result in an assembly sequences that violate such precedence relationships. One way of enforcing the feasibility of the assembly sequences is through penalizing the objective functions of infeasible candidate designs. However, such penalizing means relying on the selection according to fitness, while allowing the integer numbers on the chromosome to take on any value within the maximum and minimum ranges, which would lead to an impossibly large search space that is probably beyond the capabilities of GA or any other optimization technique. Instead, after crossover and mutation, a repair operator is invoked that systematically increments the integer number on the chromosome so that the assembly sequence remains feasible. Such integer incrementing is performed for every four-integer assembly operation till it represents one of the assembly operations defined on the set of feasible assembly operations, that can be invoked to join two of the components or sub-assemblies that are in the hypothetical bin of the product.

3.3 Crossover

In the following example, four crossover schemes are tested: Arithmetic Projection, Multi-Point, Uniform, and a special crossover operator especially tailored for the proposed encoding scheme.

Arithmetic Projection Crossover

This is a blend of arithmetic crossover [9] and heuristic crossover [9]. Both arithmetic and linear projection crossover schemes are mainly applied in real coded GA's and they are tested in this paper although the chromosome is integer coded, because some portion of the chromosome has underlying continuity (the number of assembly sta-

tions). When viewing the chromosome in multi-dimensional vector space, such crossover schemes set the gene values of the offspring chromosome to some point along the line joining two parent points. In arithmetic crossover, the offspring point is in between the parents, while in projection crossover the point is outside the parents, projected in the direction of the better (more fit or higher rank) parent. Thus in the combined version of arithmetic-projection, the gene values of the offspring are given by:

$$g_C = g_{P1} + r(g_{P2} - g_{P1}) \tag{3}$$

Where g_c is the offspring gene value, g_{p1} is the gene value of the less fit (or lower rank) parent, g_{p2} is the gene value of the more fit parent, and r is a uniformly distributed random number within a certain range. If the range of r is between zero and 1.0, Eq. 1 would represent arithmetic crossover, while if the range is between 1.0 and some number bigger than 1.0, Eq. 1 would represent heuristic crossover. In this paper, r is uniformly distributed between zero and 2.5, so it represents a blend of both arithmetic and heuristic crossover.

Multipoint Crossover

Similar to single-point crossover but exchange of chromosome segments between parents to produce the offspring chromosome occurs at several points [4]. It is believed that multipoint crossover works better for long chromosomes.

Uniform Crossover

Can be viewed as an extreme case of multipoint crossover, where every building block can be exchanged between the parent chromosomes, with a specified probability.

Special Crossover Scheme

A Special crossover operator is designed based on understanding of the nature of the encoding scheme of the chromosomes. The basic idea is that the *meaning* of the numbers of a four-integer building block is dependent on the *state* of the components and subassemblies present in the hypothetical bin of the product. Thus if crossover changes some building block on a chromosome, it means crossover has subsequently changed the meaning of *all* the building blocks that follow the changed building block for that product. The special crossover operator is designed as follows:

- One single-point crossover is used for every product on the chromosome. This is slightly different from multipoint crossover as one crossover occurs in every product, while in multipoint crossover, one product can have several crossover points, while another has none.
- Crossover can occur only at intervals in between the building blocks (which are of length four, each defining an assembly operation).

- In order to grow good schemata in the population of chromosomes, the crossover location in the initial population has higher probability of occurring nearer to the start of the string, but as the search progresses, the crossover location probability *centeroid* is shifted gradually towards the end of the string.

It is worth mentioning that within one product, single-point crossover is chosen and not two-point or multi-point crossover because of the heavy backward dependency in the chromosome introduced by the encoding scheme. This backward dependency makes a second crossover point within the same product no more than a random mutation for the remaining part of the chromosome.

3.4 Mutation

When arithmetic projection, multipoint or uniform crossover are used, mutation of the building blocks occurs in a classical fashion, where according to a user specified probability, random changing of the numbers on the chromosome to any number within their allowed range with uniform probability is performed [9]. However, when the special crossover is employed, a corresponding special mutation is used.

In the special mutation, separate probabilities are assigned to each of the four numbers of the building block. Typically, the first number has the least mutation probability, and then following numbers the next have higher and higher mutation probabilities. The intuition behind this is that the *meaning* of the second number on the building block is dependent on the first number, so changing the first subsequently changes the second. Also the first two numbers on the building block play the role of defining the assembly *sequence*, so changing any of them subsequently changes the *meaning* of the whole chromosome portion that follows the location of the mutation.

It is noted that the special mutation scheme could also be applied to the classical forms of crossover considered. However, the authors preferred keeping the study of specialized operators (crossover and mutation) separate from classical ones in order to highlight the benefit (if any) that is gained by introducing prior knowledge of the problem structure into the GA operators.

3.5 Multi-objective GA

The implemented multi-objective GA is similar to NSGA-II [10, 11] with a few modifications in enforcing elitism, where the un-dominated members are copied into a separate elite population that is preserved across generations unless it grows too big. General highlights of NSGA-II that differ from single objective GA include: i) selection is based on ranking through Pareto-based dominance and ii) use of a Nitching function in selecting from with members that have the same rank. The Nitching function serves to give higher probability of selection for members that have no others

near to them and thereby enhances the spread of the population over the Pareto-front. The pseudo-code for the implemented multi-objective GA is given as:

```
1. Generate Initial Population
2. Loop Until Termination: (Max. Number of Generations or Function
   Evaluations)
3. Perform Pareto-Ranking of Current Population
4. Add Un-dominated Members of Current Population Members into Elite
   Population
5. Re-Rank Elite Population and Kill any members that become dominated
6. If Elite Population Grows beyond an allowed size, select a number of
   members equal to the allowed elite population size according to a
   Nitching function and kill the rest of the members
7. Loop until New Population is full
8. Select Parents for reproduction
       a. Perform a binary tournament to select a rank
       b. Select a parent member from within a chosen rank according
          to a Nitching function
9. Perform Crossover, mutation and Repair then place new offspring in
   New Population
10. When New Population is full, replace Current Population with the new
    one and repeat at Step #2.
```

4 Results and Discussion

The feasible assembly operations satisfying the assembly precedence relations for each of the ventilator models are shown in a tree representation in Fig. 5. A list of assembly stations is provided in Table 2. Ten multi-objective GA runs are performed for each of the crossover schemes, using a population size of 150, for 100 generations, with a limit on the elite population size of 30 individuals. Crossover and mutation follow the general recommendation [4] of high crossover probability (about 0.9), low mutation rate (about 2%).

Figure 6 shows typical populations at the beginning and end of the search and their corresponding un-dominated elite front. Since the objective is to minimize both assembly cost f_1 and production shortage f_2, the closer a candidate design gets to the lower left corner of the plot, the better the design is. The difference between the initial and final populations is quite apparent. The initial population is scattered over a wide range on the $f_1 - f_2$ space, while final population has its un-dominated front closer to the lower left corner of the plot and the whole population is gathered pretty close to the un-dominated front. As such, the plot implies that convergence is achieved and that the choice of number of generations is sufficient.

Of particular interest, is the extreme point along the Pareto-front, which has zero production volume shortage (*i.e.* f_2=0), meaning that it has minimal production cost while meeting the required production volume during a given production period. This point on the Pareto-front is referred to as the *best point*. In the best-known solution of the problem, the best point has the assembly sequences shown using thicker lines in

Fig. 5. The best-known solution also utilizes types and number of assembly stations as shown in the last column of Table 2. It is observed, that whenever possible, the best solution avoids employing assembly stations that incur extra cost to do an operation.

The history of the average value of f_1 for the best point during the search is displayed in Fig. 7. The percentage success in attaining the best-known solution is given in Table 3. It is seen in Fig. 7 that given enough generations, all the considered crossover schemes succeeded in bringing down the objective function to the near-optimal region. Arithmetic-Projection crossover seems to have the fastest early descent because of its embedded gradient following heuristic, but has the least probability of success in attaining the best-known solution, which is probably due to premature convergence. Multi-point crossover has the slowest convergence rate but has a good chance at reaching the best-known solution. The proposed crossover scheme, which is tailored according to the structure of the encoded chromosome, has a moderate convergence rate, but superior chance at reaching the best-known solution. A possible reason is that the employed encoding scheme introduces heavy backward dependency to the chromosome genes, which presents difficulty to traditional crossover schemes, but the special crossover scheme is better geared to operate with such encoding.

Overall, all the considered crossover schemes performed well at getting close to the best-known solution, which implies the effectiveness of using multi-objective GA's, and in particular the adapted version of NSGA-II, for similar problems.

Table 2. List of assembly stations.

ID	Operation	Inv. Cost	Hr. Rate	# in Opt.	ID	Operation	Inv. Cost	Hr. Rate	# in Opt.
1	Frame to L. Bracket	1000	5.0	1	11	Blades	4000	10.0	2
2	Frame to U. Bracket	3000	10.0	1	12	L. Brckt 2 to L. Brg. 2	2000	8.0	1
3	L. Brckt to Axle 1	2000	8.0	1	13	U. Brckt to U. Brg.	2000	8.0	1
4	L. Brckt to Axle 2	3000	8.0	NA	14	L. Brg. 2 to Shaft	4000	10.0	1
5	U. Brckt to Axle	4000	8.0	1	15	U. Brg. to Shaft	2000	8.0	1
6	Axle to L. Brg.	1000	8.0	1	16	Inner Rotor 1	3000	8.0	2
7	Axle to U. Brg. 1	1000	8.0	1	17	Inner Rotor 2	5000	10.0	NA
8	Axle to U. Brg. 2	4000	8.0	NA	18	Shaft to L. Hub	3000	8.0	1
9	L. Brg. to L. Hub	1000	8.0	1	19	Shaft to U. Hub	3000	8.0	1
10	U. Brg. to U. Hub	1000	8.0	1					

5 Conclusion

This paper presented the application of an adapted version of NSGA-II for multi-objective optimization of a real multi-product assembly shop. Although using real data, the model of an assembly system is rather simple. Further extension of this study, would test the same approach on more sophisticated assembly models through

linking to random discrete time event simulation software such as ARENA [12]. Also presented in this paper, was a special encoding scheme for the decision parameters and associated special crossover, mutation and repair operators. The obtained results for the simplified assembly model are encouraging and motivate further exploration.

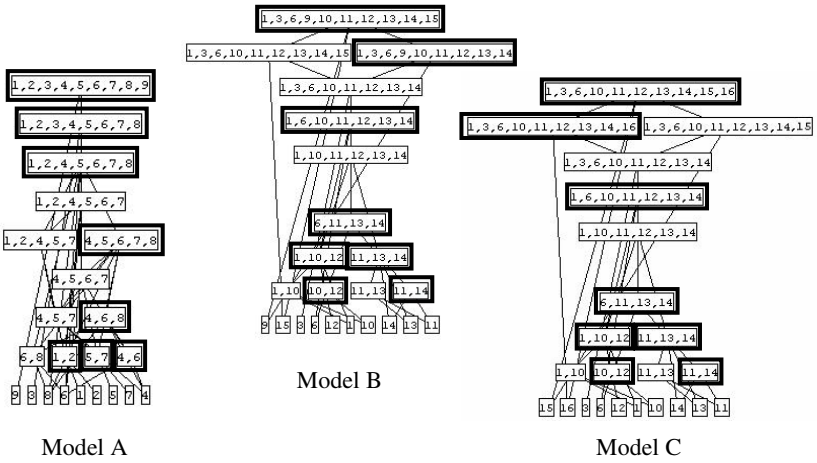

Fig. 5. Feasible assembly operations and sequences for best-known solution (thick lines).

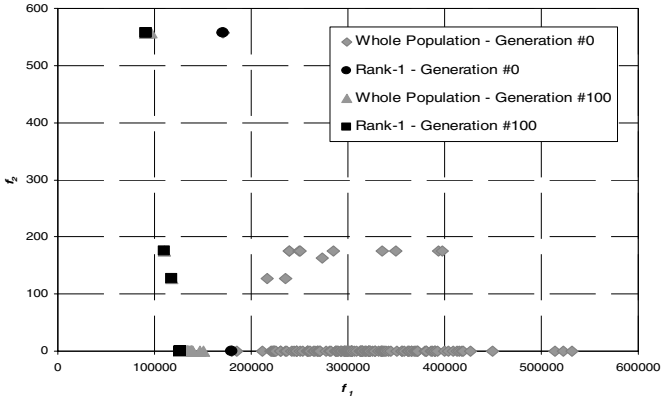

Fig. 6. Typical display of initial and final populations and their un-dominated Pareto-fronts.

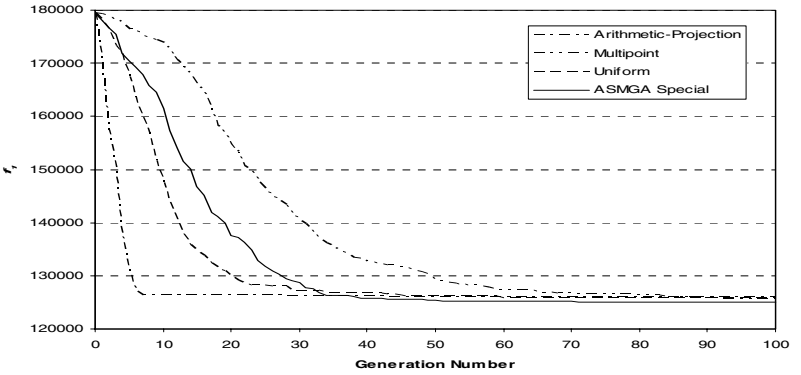

Fig. 7. History of search average progress for the best point.

Table 3. Percentage success in attaining the best-known solution.

Crossover Scheme	Percentage Success
Arithmetic Projection Crossover	30%
Multipoint Crossover	50%
Uniform Crossover	40%
ASMGA Special Crossover	90%

Acknowledgements. This work is an extension of a course project for ME588: Assembly Modeling conducted during the Fall 2002 semester at the University of Michigan, Ann Arbor. MECO: Modern Egyptian Contracting provided the data for the wind-propelled ventilators.

References

1. De Fazio, T., Whitney, D.: Simplified Generation of All Mechanical Assembly Sequences. IEEE J. of Robotics and Automation, Vol. 3, No. 6, (1987) 640–658.
2. De Mello, H., Luiz S., Sanderson, A.: A correct and complete algorithm for the generation of mechanical assembly sequences. IEEE Transactions on Robotics and Autonomous Systems, Vol. 7, No. 2 (1991) 228–240.
3. Garey, M., Johnson, D.,: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, NY, (1979).
4. Goldberg, D.: Genetic Algorithms in Search Optimization and Machine Learning. Addison – Wesley Publishing Company (1989).
5. Awdah, N., Sepehri, N., Hwalwshka, O.: A Computer-Aided Process Planning Model Based o Genetic Algorithms. Computers and Operations Research, Vol. 22, No. 8, (1995) 841–856.

6. Kaeschel, J., Meier, B., Fisher, M., Teich, T.: Evolutionary Real-World Shop Floor Scheduling using Parallelization and Parameter Coevolution. Proceedings of the Genetic and Evolutionary Computation Conference, Las Vegas, NV (2000) 697–701.
7. Chen, J., Ho, S.: Multi-Objective Evolutionary Optimization of Flexible Manufacturing Systems. Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA (2001) 1260–1267.
8. Saitou, K., Malpathak, S., Qvam, H.: Robust Design of Flexible Manufacturing Systems using Colored Petri Net and Genetic Algorithm. Journal of Intelligent Manufacturing, Vol. 13, No. 5, (2002) 339–351.
9. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996).
10. Deb, K., Argawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. Proceedings of the Parallel Problem Solving from Nature VI Conference, Paris, France (2000) 849–858.
11. Coello, C., Van Veldhuizen, D., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers (2002).
12. Kelton, W., Sadowski, R., Sadowski, D.: Simulation with ARENA. 2nd edn, McGraw Hill (2002).