Review of techniques for optimization of funding allocation

Jacques Heunis
HNSJAC003
University of Cape Town

ABSTRACT

In large corporations, loans between sub-companies are necessary in order to make good use of the total available funds. This is an optimization problem that is current done manually, leading to inefficient allocations which increase reliance on more expensive third-party sources of funding. We examine the role of optimization in the finance industry, considering what it is applied to and what methods are used. We also review common solutions to two related optimization problems, namely resource allocation and job scheduling, in order to examine their applicability to the presented fund matching problem. We find genetic algorithm and hybrid genetic algorithm approaches as used in the solution of resource allocation problems to be the most promising method.

CCS Concepts

•**Mathematics of computing** → **Optimization with randomized search heuristics; Evolutionary algorithms;** *Bio-inspired optimization;* •**Theory of computation** → *Scheduling algorithms;* •**Computing methodologies** → *Genetic algorithms;*

Keywords

Resource Allocation Problem, Job Shop Scheduling Problem, Genetic Algorithm

1. INTRODUCTION

Many large companies are composed of a number of smaller sub-companies, each with their own levels of income and expense. These sub-companies can be categorized by their profitability, either generating excess income (a funding source) or being in need of outside income (a funding sink). Ordinarily, companies that require income from third-parties would need to request funding from an external source such as a bank. Being part of a larger company, however, allows funds to be easily loaned between sources and sinks. Currently, sources are matched to sinks manually by a company employee. This results in suboptimal arrangements, increasing the need for third-party funding and costing the company money.

We are therefore presented with the following problem: Given a set of funding sources and sinks, each with a starting date, duration, funding amount and tax class, match the sources to sinks, such that the total amount of unsatisfied funding required by all sinks, is minimized. The set of solutions is also constrained by the fact that a sink and source can only be matched together if they both do, or do not have a tax class. Clearly this is an optimization problem, however optimization problems can vary widely and solutions need to be selected and tailored according to the specific problem at hand. We survey common solutions to both the resource allocation problem, and the job-shop scheduling problem, since each bears some resemblance to the problem we are trying to solve.

The Resource Allocation Problem (RAP) can be defined as follows: Given a set of resources and a set of tasks where each task has a resource cost and an amount of value gained on completion, find the optimal allocation of resources to tasks. How optimality is defined for an allocation depends on the context of the problem, the optimal solution is usually one that minimizes the total resources required, or maximises the total value gained from all activities. Our problem is similar to RAP if we consider funding sources to represent the resources and funding sinks to represent the activities, we seek the allocation which results in the smallest number of sinks not fully funded.

The Job Shop Scheduling Problem (JSSP) can be defined as follows: Given a set of jobs each consisting of an ordered sequence of operations and a set of machines each of which can complete a specific operation, find the optimal schedule that assigns jobs to machines such that all operations of all jobs are completed in sequence. In JSSP, an optimal solution is one which minimizes the so-called "makespan", which is the total time taken to complete all jobs. Our problem is somewhat similar to JSSP if we consider machines to represent funding sources and jobs to represent funding sinks. There is a variation of JSSP called the Flexible Job Shop Problem (FJSP) in which an operation can be completed by any one out of a set of machines. This is more similar to our problem because any source may fund any sink as long as they both do or do not have a tax class.

Section 2 briefly explains basic algorithms and terminology found in optimization literature. Section 3 covers optimization problems in the financial industry. Section 4 reviews common strategies for solving the resource allocation and job-shop scheduling problems. Section 5 gives a discussion of those strategies and Section 6 concludes by evaluating their suitability, both in the context of our specific problem.

2. BACKGROUND

Dynamic Programming (DP) is a technique commonly used in algorithm design but also applicable as an optimization method. DP can be employed when considering a problem whose solution can be stated as a combination

of solutions to smaller instances of the same problem. DP can provide very efficient algorithms for some problems but storing the solutions to the smaller instances of the problem, allowing larger instances to simply use the result rather than having to recompute it multiple times.

Simulated Annealing (SA) was first presented by Gelatti et al. in 1983[10]. It is an iterative algorithm that searches for a global solution to an optimization problem by starting with any solution, repeatedly selecting a neighbouring solution (one which is only a slight modification of the current one) and checking if it is better. If it is better, the new solution is adopted. If it is worse, it is adopted with some probability. That probability depends on a parameter t which is decreased over time in order to prevent the algorithm from jumping around too much and instead slowly converge to a solution.

A Genetic Algorithm (GA) was first proposed by Holland in 1975[13]. A GA consists of a population (a set of solutions), mutation and crossover operators (unary and binary operations respectively which modify members of the population to produce new ones, a so-called "child" population), and a selection process (by which members of the current and child populations are selected for the next iteration's population). The selection process is based on a fitness function which evaluates candidate solutions (members of the population) to determine how good of a solution they are.

Particle Swarm Optimization (PSO) was first presented by Eberhart and Kennedy in 1995[8]. As with GA above it works by iteratively updating a set of solutions that converge over time. Unlike GA though, PSO is not generational, the population does not get replaced, only updated. Each iteration every member of the population considers the best solution encountered so far by itself, its neighbours (a set of randomly selected members of the population) and by the entire population. It then moves a small distance towards a weighted average of those three points in solution space. This type of movement requires that moves of any magnitude and direction are possible in solution space, meaning that PSO is only applicable to problems with a continuous domain.

Ant Colony Optimization (ACO) is an algorithm aimed at solving discrete optimization problems that was first proposed by Dorigo in his PhD thesis in 1992[7]. ACO only works on discrete problems because it requires candidate solutions to be constructed out of some kind of component, for example an allocation of an operation to a machine at a given time in JSSP. Each component has a fitness, called a "pheremone" associated with it. Every iteration, a number of candidate solutions are constructed by randomly selecting components. The probability that a component is selected depends on its pheromone. When an entire population of solutions has been constructed, the fitness of each solution is calculated and the pheromones of the components of every solution is increased according to that solution's fitness. All pheromones are decreased by a constant factor every iteration to prevent pheromone values from growing unbounded. ACO keeps track of the best solution found so far, updating it any time an even better solution is found. This best solution is then given as the final solution after the desired number of iterations.

Evolutionary Algorithm is the term given to generation-based stochastic search techniques such as GA. Multi-objective Evolutionary Algorithms (MOEAs) are evolutionary algorithms that aim to simultaneously maximize or minimize more than one fitness function (or "objective"). These objectives may be at odds with each other so finding a single best solution is usually not possible to do algorithmically. In order to be able to reason about the trade-offs between objectives and what it means for a candidate solution to be "optimal" the notion of Pareto dominance is used. A solution X is said to dominate solution Y if X is at least as good as Y with respect to every objective, but strictly better with respect to at least one objective. A solution is said to be non-dominated if there are no solutions which dominate it. The Pareto front is the set of all non-dominated solutions to a multi-objective problem, and this is what MOEAs aim to compute. Since the algorithm cannot say that one non-dominated solution is "better" than another, it gives all of them and leaves it up to the user to decide which one to use. Another concept which is important in MOEAs is that of Pareto rank. Pareto rank is defined recursively with solutions on the Pareto front being given rank 1. If the solutions on the Pareto front were removed from the population and the front were recalculated, those solutions that would be in the new front would have rank 2. This process is continued until the entire population has been assigned a rank. Unlike the terms explained above, MOEA is a class of algorithms rather than a specific algorithm. An example of an MOEA is the Non-Dominated Sorting Genetic Algorithm 2 (NSGA-II) proposed by Deb et al[6]. Being a GA, NSGA-II is generational, but in addition to the usual population for every generation, an archive of the best solutions found so far in the entire run (IE across generations) is kept, and new generations are created from this archive population, rather than from the generational population. Every iteration the Pareto rank of every individual in the entire (generation and archival) population is computed and a new archive is created, adding individuals of highest rank until it is full. The output is the Pareto front of the last generation.

3. OPTIMIZATION IN THE FINANCIAL INDUSTRY

The problem of optimally matching funding sources to sinks is currently solved manually. As a result, we were unable to find any literature regarding the solution to this specific problem in a financial context. However, optimization is commonly applied to other problems in the financial industry such as portfolio optimization or ensuring fair allocations of funding to projects. One of the main topics of interest is the minimization of risk. While there is a large amount of literature on the minimization of risk in finance, we will not explore it in detail because risk is not a significant component of the problem we are interested in.

3.1 Risk management

Many problems in finance inherently have multiple (often competing) objectives. A clear example of this is the competition between the minimization of risk and maximization of reward, a problem that appears in investment portfolio management and stock trading[19]. As a result of this innate multi-objectivity, much of the recent literature focusses on applying MOEAs to financial decision problems. Both Metaxiotis et al. and Ponsich et al. published reviews on literature regarding the application of MOEAs to portfolio

management[17][19], in both cases most of the reviewed work made use of MOEAs, but other techniques including SA and PSO were also shown to be successful. While they do not involve the specific problem we are interested in, these reviews indicate that there is much work being done on improving optimization techniques in the context of the financial industry, and that there is significant interest in improving the ability for algorithms to aid in financial decision making.

Ponsich's review is more useful for us, because it includes a summary of other applications of MOEAs in the financial industry. Other problems mentioned include prediction of financial time-series and recommendation of stock purchases based on the stock's historical performance. Of particular interest is that no mention is made of our problem. This could indicate that there is little-to-no existing literature on optimization of corporate fund matching.

3.2 Maintaining fairness

Gurrola et al. present an algorithm for funding allocation that attempts to minimize envy[11], which they define as how much a party feels that he/she has been allocated an unfair amount of some resource (in this case funding). Their description of their algorithm mentions that it runs in iterations, but does not include what happens on each iteration or what changes across iterations, making it very hard to reproduce. In addition no comparison is given to other solutions of the problem, so while their given results appear to indicate that their method is a viable solution, there is no context within which to evaluate it as *good* solution.

Chang et al. also attempted to minimize envy in the allocation of funding to transportation projects[4]. Instead of implementing an optimization algorithm themselves, the SolveXL software package was used. It is not clear exactly what method or algorithm SolveXL uses, it is only stated that it "uses evolutionary algorithms"[21]. Although it is possible to acquire SolveXL in order to replicate results, this lack of information makes detailed comparisons of algorithms very difficult. The performance of SolveXL's evolutionary algorithm was compared to that of a greedy algorithm across 5 instances of their funding allocation problem, showing that the evolutionary algorithm produced a significantly lower average envy than the greedy algorithm.

4. OPTIMIZATION STRATEGIES

There is a wide variety of optimization problems and each one responds differently to types of solutions. In order to get an accurate depiction of what methods would be most effective, we need to consider literature on problems that are the most similar to the one we are trying to solve. To that end we will consider recent solutions to the resource allocation problem (Section 4.1) and the job-shop scheduling problem (Section 4.2).

4.1 Resource Allocation

As mentioned in section 1, our problem is similar to the general resource allocation problem (RAP) if the funding sources represent the resources and the sinks represent the activities. The solution is then the allocation that best matches sinks to sources such that we minimize the total amount of funding needed to satisfy all of the sources.

Given the complexity of solving RAP, most methods rely on metaheuristics (stochastic optimization algorithms that try to find good solutions through a search that is directed by an estimate of how "good" a given solution is) to search for good approximations to the optimal solution, not worrying if the output is not actually the optimal solution as long it is reasonably close. We will therefore focus on reviewing these methods.

4.1.1 Ant Colony Optimization

Lee et al. presented a new hybrid algorithm that combines the global search capabilities of a GA (which considers a wide variety of solutions, rather than focusing on a small area of solution space) with the exploitative benefit of ACO[15]. The algorithm works like a normal GA, except that it keeps track of the best solution seen so far. Every time a new best solution is found in a population, ACO is run on the current to push the members of the population towards a local optimum. Their algorithm performed significantly better than SA, GA and ACO solutions on a set of randomly generated resource allocation problems. The reproducibility of their results is hindered because the procedure by which the random problems are generation is not described, however the way in which GA and ACO were combined is easily transferable to other methods.

Chaharsooghi et al. presented a version of ACO that had been modified to apply to multi-objective optimization problems[3]. In contrast to Lee's choice of random problem generation above, Chaharsooghi tested the new algorithm on a problem previously presented by Lin et al.[16], allowing for a more reproducible comparison. They also provide analysis of how their algorithm arrived at its result, showing that the final solution was found by convergence of the entire population rather than by random exploration of a single member of the population. This is important as it shows a measure of stability in the algorithm, allowing it to find good solutions consistently.

Vitanov et al. applied an ACO the problem of allocating buffer space between sections of an assembly line[22]. Since there is no obvious way of evaluating the efficiency of a given allocation they relied on simulation instead of a direct objective function. This use of simulation removes the need to construct an accurate objective function, but warrants a modification of the usual ACO algorithm, a stochastic variant attributable to Gutjahr was used[12]. In order to evaluate the performance of ACO on this problem, the algorithm was compared to an SA approach implemented by the existing WITNESS optimization software package. Their results show that both the SA and ACO approaches returned very similar solutions. While the fact that ACO was able to do as well as an existing solution shows that it is capable of effectively solving the problem, it does not show ACO to actually be an improvement on existing solutions. It is important to note however, that the ACO solution performed equally well while running far fewer simulations than the SA solution. This was done due to the performance constraints of Vitanov's implementation and a different implementation with fewer performance constraints might be able to give better results in a comparable amount of compute time.

4.1.2 Genetic Algorithms

To see why metaheuristics are commonly selected for solving resource allocation problems, we can refer to the work of Osman et al. who presented a comparison between genetic algorithms and classical dynamic programming approaches

to solving multi-objective problems[18]. They found that while dynamic programming can effectively provide optimal solutions, it can only optimize a single objective function at a time, meaning that multiple objective functions must be combined in a weighted average whose weights represent how "important" each objective is, and are pre-determined. Requiring a pre-determined weighting of objectives essentially shifts some of the work of finding good solutions from the algorithm onto the user, likely resulting in convergence to sub-optimal solutions. In addition, the dynamic programming solution does not scale very well when compared to newer multi-objective genetic algorithms which scale far better to large problems and do not require that the different objectives be given pre-decided "importance" weights.

Lin et al. also presented a solution to the multi-objective resource allocation problem[16]. Unlikely Chaharsooghi's solution however, Lin's algorithm does not apply ant-colony optimization, preferring instead to take a hybrid approach based on genetic algorithms and local search. The algorithm runs a genetic algorithm to explore the problem space, but uses local search to improve each child before it is passed to the selection process for the next generation. Of interest is the way in which they chose to represent their population. Since they are trying to solve a resource allocation problem with a fixed number of activities (in this case: four), they elected to represent each chromosome as a "path" through the list of activities, with each gene corresponding to a particular activity and following along the chromosome from left to right maps out a path through all activities to which the resource should be allocated. The algorithm was only tested on a single problem, and no comparison other approaches was given, although the author claims on the test problem, the algorithm managed to efficiently find a good Pareto front.

Xu et al. considered the problem of assigning virtual machines to physical servers in cloud infrastructure as a multi-objective optimization problem trying to minimize resource wastage, power consumption and maximize heat dissipation[23]. This can be framed as a bin-packing problem, which traditional GAs are not well-equipped to solve[9] and so they make use of a modification of the traditional GA called the "grouping genetic algorithm" (GGA) which takes the structure of the problem ("grouping" VMs together such that they can be allocated to a physical server) into account to improve performance. They compare their proposed Multi-objective GGA with four traditional bin-packing heuristics (first-fit decreasing and best-fit decreasing, ordering by CPU and memory requirements) and two single-objective GGA algorithms (optimizing with respect to power consumption and temperature). Their proposed multi-objective solution performs better than the compared solutions in most cases, the only case where it was beaten was when comparing it to the single-objective GGA that optimized for heat dissipation, and evaluating the solutions by temperature alone. Overall this indicates that the proposed solution is an effective method for solving this problem.

4.2 Job Scheduling

We can roughly relate our problem to the job shop scheduling problem (JSSP) if we think of the funding sources as machines and the sinks as jobs. While the similarity to our problem is not as close as with resource allocation, we feel that it is still sufficiently close in principal for a review of the current literature to be useful.

We will focus on metaheuristic solutions since JSSP is NP-Hard and so the vast majority of recent literature on the topic focusses on these types of solutions in order to keep to a reasonable computation requirement.

4.2.1 Genetic Algorithms

One variation of the JSSP is the flexible job shop scheduling problem (FJSP) whereby instead of each stage of each job needing to be processed by a specific machine, it can be processed by any of a set of machines. This is actually closer to our problem than JSSP because we do not distinguish between "stages" of funding. Cwiek et al. presented a genetic algorithm for solving the FJSP that randomly selects between two combination operators at each iteration[5]. The crossovers used are active schedule constructive crossover (ASCX), developed by Park[14] and generalized order crossover (GOX) developed by Bierwirth[2]. The algorithm was run with a variety of configurations on benchmark problems and the results show that a fairly elitist parent selection process, coupled with a high probability of selecting ASCX over GOX, resulted in more optimal output on average although the 3D nature of the presented graphics somewhat obscure the details of this trade-off.

Naturally real-world problems usually involve optimization according to more than one criteria. To address this, Ali et al. presented a GA that optimizes according to two objectives by first running a normal GA on each one individually and them getting a list of solutions that fall within the area of problem space bounded by the best solution for each (since it is assumed that an optimal solution for one criteria does not have an optimal value for the other)[1]. This list is then processed to find the actual Pareto front for the two objectives and finally, a weighting is introduced to calculate a single "best" solution and avoid manual analysis of the Pareto front. It is unclear why this final step is necessary since other authors are content to leave the problem at a set of Pareto-optimal solutions and leaving it up to the reader to decide how they should select between them. The solution is run on five benchmark problems but no other solutions are given to compare to, so there while it is stated that the algorithm "has shown its effectiveness and feasibility for solving JSPs"[1], there is no basis on which to judge the algorithm's performance from the results provided in the paper alone.

4.2.2 Particle Swarm Optimization

The JSSP is an inherently discrete problem, and while Particle Swarm Optimization (PSO) is an algorithm targeted specifically at continuous domains, Zhang et al. have proposed a hybridization of PSO with tabu search and simulated annealing in order to solve the JSSP[24]. The article does not appear to detail how PSO is modified in the proposed algorithm in order to work in a discrete problem space. The proposed algorithm was tested on a variety of benchmark problems and compared to known optimal solutions to those problems as well as other algorithms that are described using abbreviations which do not appear to be explained. The presented algorithm appears to do well but we are somewhat suspicious of the given findings being misleading. For example one table of results gives the fitnesses of solutions found a number of algorithms. In some cases the fitnesses of the proposed algorithm are given in

bold (which in other literature, indicates the best solution although in this case it is not mentioned what bold indicates) even though other algorithms performed better.

Shao et al. presented a hybrid SA-PSO approach to solving the multi-objective FJSP without the need for pre-defined weighting values to compile the different objectives[20]. In order to apply PSO to the discrete scheduling problem, the particles are defined to have two vectors associated with them (similarly to position and velocity in normal PSO), one for the permutation of operations within each job, and one for the ordering in which each machine is visited. These are then updated based on their similarity to the local and global best-seen solution where Hamming distance is used as a measure of similarity. This key modification allows for greater efficiency in the algorithm as the state of every particle is always a valid scheduling solution, avoiding the problem of having to discard invalid solutions and try again whenever they appear, which is essentially a waste of computation time. The hybrid nature of the algorithm comes in when, after being shifted around each iteration, the particles are improved using simulated annealing for a few iterations before continuing. This contrasts with the explorative nature of PSO to exploit the existing solution and tend towards local optima. The proposed algorithm is finally compared to 6 other existing algorithms on an assortment of benchmark problems, performing at least as well across all objectives in most scenarios.

5. DISCUSSION

MOEAs are commonly used in the financial industry for solving optimization problems including portfolio management and minimization of fairness, as demonstrated by Ponsich and Metaxiotis[19, 17]. However, these problems do not bear very much similarity to our problem in terms of their goals and what factors they take into account, meaning that the work is largely not applicable.

Lee, Chaharsooghi and Vitanov all presented ACO solutions to the resource allocation problem[15, 3, 22], however Chaharsooghi was the only one to provide an analysis of how the final result was discovered by the algorithm, showing that it was not simply by luck which is good for the reliability of the algorithm, especially in a corporate context. This is the exception though, and other authors still run their algorithms with sufficiently many trials to provide a convincing case for the reliability of their solutions. Of the 3 solutions mentioned, only Lee's was a hybrid approach, incorporating the benefits of both ACO and GA in an attempt to improve performance. The incorporation of ACO is sufficiently simple that it can easily be incorporated into other solutions.

Lin, Osman and Xu all presented GA-based solutions to the multi-objective resource allocation problem[16, 18, 23]. Lin and Xu both attempted novel solutions, with Lin incorporating local search into their GA for improvement of the child population, while Xu et al. related their problem to bin packing and used a variant of the standard GA which is better suited to so-called "grouping" problems. Osman on the other hand, compared his solution to a more traditional dynamic programming algorithm aimed at multi-objective optimization problems, finding that the GA performed far better on large-scale problems such as those that would be found in most real-world applications. In all three cases, the GA solution proved effective at finding good solutions to the multi-objective problem.

Looking to solutions for job scheduling, Cwiek presented a novel GA solution to the single-objective FJSP which randomly selects between crossover operators in order to improve performance[5]. The solution was tested with a variety of settings for the probability with which each operator was selected, which turned out to be useful because the optimal setting was to strongly favour one of the operators.

Ali proposed a solution to JSSP with two objectives which involved running a normal GA on each objective before combining them at the end[1]. This contrasts with the other multi-objective GA solutions reviewed above in that usually multi-objective problems are addressed with algorithms specifically designed for handling multiple objectives. This is exactly so that they can avoid the problem of having to give weights to the different objectives, instead outputting a list of solutions which are all optimal in some sense. Even though Ali's approach does compute such a list of solutions, it does not stop there and elects to still combine them in some pre-defined way to arrive at one final solution, leaving no room for human intervention to decide which of the "optimal" solutions to select.

Taking advantage of the work done to solve RAP using ACO will likely be difficult because our problem is more naturally presented in a continuous formulation and so the standard ACO algorithm cannot be used and a modification would be needed or the problem would need to be formulated in a discrete manner. Similarly, the work done on JSSP and FJSP is largely not directly applicable because of the discrete nature of the problem. The ideas presented by the reviewed work could be useful though, for example Cwiek's random selection of crossover operators could be applied to any GA solution.

6. CONCLUSIONS

To conclude, given Metaxiotis and Ponsich's extensive reviews on portfolio and risk management[17, 19], it is clear that the financial industry is making extensive use of the recent advances in the field of optimization to gain business value, however our problem was nowhere to be found in literature, which leaves a convenient gap that we can fill regarding methods of solution to this problem.

Real-world optimization problems are frequently multi-objective, causing most recent research to focus on methods for solving problems with more that one optimization criteria, however doing so increases the complexity (both conceptual and computation) of the problem. Even so, a number of effective solutions have been presented some of which require pre-defined weighting of the objectives and some of which do not, leaving room for comparison between approaches to a specific problem.

While job scheduling appears similar to the problem we are trying to solve, its discrete nature makes it difficult to borrow solutions directly, leaving us to instead make use of the ideas presented by solutions to JSSP or FJSP and rely more heavily on solutions to RAP.

The most promising area of those reviewed is therefore the application of standard and hybrid GA solutions as used to solve RAP.

7. REFERENCES

[1] A. Ali, P. Hackney, D. Bell, and M. Birkett. Genetic algorithms for solving bicriteria dynamic job shop

scheduling problems with alternative routes. In *Proceedings of the The International Conference on Engineering & MIS 2015*, ICEMIS '15, pages 31:1–31:8, 2015.

[2] C. Bierwirth. A generalized permutation approach to job shop scheduling with genetic algorithms. *Operations-Research-Spektrum*, 17(2):87–92, 1995.

[3] S. Chaharsooghi and A. H. M. Kermani. An effective ant colony optimization algorithm (aco) for multi-objective resource allocation problem (morap). *Applied Mathematics and Computation*, 200(1):167 – 177, 2008.

[4] C. M. Chang and E. Montes. An optimization approach applied to fair division transportation funding allocation models. *Journal of the Transportation Research Forum*, 53(1):101 – 118, 2014.

[5] M. Cwiek and J. Nalepa. A fast genetic algorithm for the flexible job shop scheduling problem. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO Comp '14, pages 1449–1450, New York, NY, USA, 2014.

[6] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings*, chapter A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II, pages 849–858. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[7] M. Dorigo. *Ant Colony Optimization for vehicle routing problem*. PhD thesis, PhD thesis, Politecnico di Milano, Milan, Italy, 1992.

[8] R. C. Eberhart, J. Kennedy, et al. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[9] E. Falkenauer and A. Delchambre. A genetic algorithm for bin packing and line balancing. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 1186–1192 vol.2, 1992.

[10] C. Gelatt, M. Vecchi, et al. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[11] E. Gurrola and H. A. Taboada. A sequential fund allocation approach to minimize envy. In *Proceedings of the 41st International Conference on Computers and Industrial Engineering*, pages 325–330, Los Angeles, CA, USA, 2011.

[12] W. J. Gutjahr. S-aco: An ant-based approach to combinatorial optimization under uncertainty. In *Ant colony optimization and swarm intelligence*, pages 238–249. Springer, 2004.

[13] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press, 1975.

[14] L. JeongPark and C. HoonPark. Genetic algorithm for job shop scheduling problems based on two representational schemes. *Electronics Letters*, 31:2051–2053(2), 1995.

[15] Z.-J. Lee and C.-Y. Lee. A hybrid search algorithm with heuristics for resource allocation problem. *Information Sciences*, 173(1):155 – 167, 2005.

[16] C.-M. Lin and M. Gen. Multiobjective resource allocation problem by multistage decision-based hybrid genetic algorithm. *Applied Mathematics and Computation*, 187(2):574 – 583, 2007.

[17] K. Metaxiotis and K. Liagkouras. Multiobjective evolutionary algorithms for portfolio management: A comprehensive literature review. *Expert Systems with Applications*, 39(14):11685 – 11698, 2012.

[18] M. Osman, M. Abo-Sinna, and A. Mousa. An effective genetic algorithm approach to multiobjective resource allocation problems (moraps). *Applied Mathematics and Computation*, 163(2):755 – 768, 2005.

[19] A. Ponsich, A. L. Jaimes, and C. A. C. Coello. A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications. *Transactions on Evolutionary Computation*, 17(3):321–344, 2013.

[20] X. Shao, W. Liu, Q. Liu, and C. Zhang. Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 67(9):2885–2901, 2013.

[21] SolveXL. Solvexl - genetic algorithm optimization add-in for microsoft excel, 2016.

[22] I. V. Vitanov, V. I. Vitanov, and D. K. Harrison. Buffer capacity allocation using ant colony optimisation algorithm. In *Winter Simulation Conference*, WSC '09, pages 3158–3168. Winter Simulation Conference, 2009.

[23] J. Xu and J. A. B. Fortes. Multi-objective virtual machine placement in virtualized data center environments. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 179–188, 2010.

[24] X.-F. Zhang, M. Koshimura, H. Fujita, and R. Hasegawa. Combining pso and local search to solve scheduling problems. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '11, pages 347–354, New York, NY, USA, 2011.