7

Multi-Objective Meta-heuristic for the Inventory Routing Problem

7.1 • Introduction

The Inventory Routing Problem (IRP) is a very important problem in logistics, especially for the vendor managed inventory (VMI) replenishment [Jia et al., 2014, Kleywegt et al., 2002]. Many companies are looking for a win-win strategy for the supplier and customers by integrating inventory management, vehicle routing and delivery strategies [Campbell et al., 1998]. This chapter considers a finite horizon tri-objective stochastic IRP with a single supplier and multiple geographically distributed customers. In this problem, each customer has an uncertain demand each day for a single product. The customers are replenished from the central supplier by a fleet of homogeneous vehicles with limited capacity. The goal is to find a delivery strategy and routing schedule that can simultaneously minimize three objectives: routing cost, inventory cost and stockout cost.

Many variants of IRP have been developed since the first pioneer paper of the inventory routing thirty years ago [Coelho et al., 2013]. The IRP was regarded as an extension of the vehicle routing problem in the early papers, and only the routing cost was considered, while the inventory levels were regarded as fixed constraints that had to be high enough to satisfy the stochastic periodical demand of customers [Bell et al., 1983]. Burns et al. integrated inventory cost as another objective and analyzed the trade-offs between inventory cost and routing cost with finite time horizon [Burns et al., 1985, Dror and Ball, 1984]. But the time horizon can also be infinite which can be seen in [Anily and Federgruen, 1990, Gallego and Simchi-Levi, 1990]. In some papers, direct

routing is adopted to simplify the problem, which means one vehicle replenishes only one customer in each period [Barnes-Schuster and Bassok, 1997, Gallego and Simchi-Levi, 1990, Kleywegt et al., 2002]. However, most real world problems are not using direct routing. Recent studies focused on the problem in which one vehicle is able to serve several customers each time [Bertazzi et al., 2002, Coelho and Laporte, 2013b]. During the execution of inventory routing a stockout may occur. This can be avoided by adding a back-logging strategy in case customers are willing to wait till the next delivery day [Carter et al., 1996, Chien et al., 1989]. Stockout is not permitted in most IRP with deterministic demands [Aghezzaf et al., 2006, Campbell et al., 1998, Geiger and Sevaux, 2011]. However, when the demands of customers are stochastic, it is impossible to avoid the stockout [Huang and Lin, 2010]. Hence, expected stockout should be regarded as the third objective to be minimized in the inventory routing problem.

The inventory routing problem with two objectives has been approached in different methods ranging from exact algorithms to metaheuristics algorithms. Archetti et al. implemented a branch-and-cut algorithm for a small scale single vehicle IRP with a short time horizon [Archetti et al., 2007]. Solyali et al. improved the branch-and-cut algorithm to an extended small scale single vehicle IRP using a strong formulation [Solyali and Süral, 2011]. In the extension, each customer has an external dynamic demand and is controlled by a deterministic order-up-to-level policy. Recently, this algorithm has already been implemented to solve the larger scale IRP with multiple vehicles [Adulyasak et al., 2014, Coelho and Laporte, 2013a]. However, exact algorithms are only able to solve the small scale IRP within limited time. In order to obtain high quality solutions for the large scale IRP, metaheuristics algorithms are implemented in the current research. Zhao et al. proposed a variable neighborhood search for the IRP in a three-echelon logistics system [Zhao et al., 2008]. A tabu search algorithm combined with ad hoc designed mixed integer programming models was applied by Archetti et al. for an IRP in discrete time [Archetti et al., 2012]. Ribeiro et al. introduced an iterative local search algorithm to solve the IRP with stochastic and deterministic demand [Ribeiro and Ramalhinho-Lourenço, 2003]. They decomposed the IRP into an each day VRP, and the iterative local search algorithm was used to find a good feasible of the VRP. Salvesbergh et al. combined an improved branch-and-cut algorithm and the greedy heuristic which is used to solve the IRP with continuous moves [Savelsbergh and Song, 2008]. A hybrid genetic algorithm

was developed for a finite horizon, multi-periods, multi-products and many-to-one distribution IRP by Moin et al. [Moin et al., 2011]. The authors used an allocation first and route second strategy to construct a solution. New crossover and mutation operators and new presentations are introduced in order to adapt the algorithm to the IRP. However, these papers all focus on bi-objective IRP. The tri-objective IRP has not been studied well yet. A modified ant colony algorithm was developed for multi-item inventory routing problem with demand uncertainty [Huang and Lin, 2010]. They proposed a new pheromone updating rule which could integrate the stockout cost. But they integrate a three-objective problem into a single objective problem by giving a tradeoff weight to each objective. Geiger and Sevaux developed a local search strategy by modifying delivery frequencies which generated a Pareto front approximation for the bi-objective inventory routing problem [Geiger and Sevaux, 2011]. They also proposed 14 benchmark instances with certain demands.

Multi-objective optimization problems became a hot topic in recent years [Campomanes-Álvareza et al., 2013, Jia et al., 2014, Li et al., 2015]. The goal in this chapter is to reduce three different cost factors simultaneously: routing cost, inventory cost and stockout cost. Whereas the first is related to the economical and ecological aspect of the problem (fuel consumption), the latter is related to the quality of service that needs to be optimized, the last one indicates the expected value of the lost sale that should be minimized. The goal of this study is to compare two different, though closely related, strategies for computing the Pareto optimal front of this problem.

Many algorithms have been implemented to solve multi-objective optimization problems [Cheng et al., 2015, Rokni and Fayek, 2010, Siddique and Adeli, 2013, Zhu et al., 2015]. In this chapter, four multi-objective optimization algorithms are compared to solve the tri-objective inventory routing problem with uncertain demand in logistics. Two of them are state-of-the-art evolutionary multi-objective optimization methods, namely Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [Deb et al., 2000] and S-Metric Selection Evolutionary Algorithm (SMS-EMOA) [Beume et al., 2007]. Moreover, an extension of a state of the art bi-objective inventory routing method by Huber, Geiger and Sevaux [Huber et al., 2013] for two objectives (inventory and routing cost) is developed to solve the tri-objective problem. Since the hypervolume indicator is a good measure for the quality of the Pareto front and particle swarm algorithms works well in optimization problems, in this chapter we also propose a new hypervolume

indicator based particle swarm optimizer. It will be called multi-objective optimization cooperative particle swarm (MOCOPS).

This chapter is structured as follows: In Section 7.2, the inventory routing problem is defined and preliminaries from the literature are described. Section 7.3 demonstrates the structure of the solver and provides a detailed description of the four algorithms. In section 7.4, the results on the benchmark problems are discussed and the performances of the proposed algorithms are compared. Finally, Section 7.5 concludes the work with a summarizing discussion.

7.2 · Problem Definition and Preliminaries

7.2.1 · Problem Definition

In general, a multi-objective optimization problem is a problem with two or more objective functions to be optimized simultaneously. Such problems are defined as:

$$f_1(\mathbf{x}) \rightarrow \min, \ldots, f_m(\mathbf{x}) \rightarrow \min \tag{7.1}$$

subject to

$$g_1(\mathbf{x}) \leq 0, \ldots, g_k(\mathbf{x}) \leq 0 \tag{7.2}$$

$$\mathbf{x} \in \mathcal{X} \tag{7.3}$$

Here $\mathbf{x}$ is an element in decision space $\mathcal{X}$, that is the space of all possible alternative solutions. $f_i(\mathbf{x})$ is the ith function of the m objective functions and $g_i(\mathbf{x})$ is the ith function of the k constraint functions. In general, the search spaces could be continuous variables, discrete variables or both. In this chapter, the search space is a discrete one and the focus is on problems with three objective functions.

The multi-objective IRP with uncertain demand can be described as follows: In this problem, products are repeatedly delivered from a single supplier to a set of n geographically dispersed customers over a given planning horizon T (in days). On different days, each customer consumes a stochastic amount of products. Moreover, customers maintain a local inventory with a maximum inventory level. The supplier has to service all customers with a fleet of homogeneous vehicles with an equal maximal capacity. The objective in [Geiger and Sevaux, 2011] was to minimize the total inventory cost and the total routing cost during the planning period. In this thesis, it is extended to a tri-objective problem that also states expected stockout cost as an objective function.

Formally, the problem is set up as described in [Geiger and Sevaux, 2011]: There are n customers and at most one vehicle with capacity C per customer. Deliveries cannot be split in the model, namely a customer can be visited at most once by one vehicle per day. Each customer $v_i, i \in \{1, \ldots, n\}$ has a maximum inventory level denoted with Q_i. For each customer $v_i, i \in \{1, \ldots, n\}$ and time $t \in \{1, \ldots, T\}$ (denoting an index for the days), $L_{i,t}$ denotes the inventory level, $q_{i,t}$ is the shipping quantity, $d_{i,t}$ is the demand to be satisfied. At day 1 the values of $L_{i,t}$ are set to some predefined initial inventory level. The inventory levels are then updated according to the equation given by Geiger and Sevaux [Geiger and Sevaux, 2011].

The stockout cost of this problem can be computed as $S_{i,t} = \max\{0, d_{i,t} - L_{i,t-1} - q_{i,t}\}$. A positive $S_{i,t}$ means that there are not enough products available at time t for customer v_i. In [Geiger and Sevaux, 2011], positive $S_{i,t}$ were avoided altogether by considering them as strict constraints. However, if demands are stochastic meaning they are not known beforehand, stockout cost cannot be avoided entirely.

A solution candidate is represented by a tuple of delivery frequencies $(\pi_1, ..., \pi_n)$ with $\pi_i \in \{1, \ldots, T\}$ for $i = 1, \ldots, n$. For each customer it determines how often it is visited by a vehicle. For instance, $\pi_i = 1$ means a day-to-day delivery for customer i, $\pi_i = 2$ indicates that on every second day a delivery takes place, and so forth. The required shipping quantities are then determined by

$$q_{i,t} = \min \left\{ \left(\sum_{\ell=t}^{t-1+\pi_i} d_{i,\ell} \right) - L_{i,t-1},\ Q_i - L_{i,t-1},\ C \right\}$$

The two objectives which were stated in [Geiger and Sevaux, 2011] are defined as:

$$f_1 = \sum_{t=1}^{T} \sum_{i=1}^{n} L_{i,t} \rightarrow \min \tag{7.4}$$

$$f_2 = \sum_{t=1}^{T} VRP_t(q_{1,t}, \ldots, q_{n,t}) \rightarrow \min \tag{7.5}$$

where Eq. 7.4 describes the total inventory cost and Eq. 7.5 represents the total cost for the routing. The latter one is determined by solving for each day a vehicle routing problem $VRP_t(q_{1,t}, \ldots, q_{n,t})$ with the given shipping quantities $(q_{1,t}, \ldots, q_{n,t})$ for day t (they are determined by the frequencies).

The vehicle routing problem is defined in the standard way: Given a set of n customers and a depot, it is required to visit each customer exactly once and deliver the

quantity q_i to customer v_i. Multiple vehicles can be used and each vehicle has the same capacity. One vehicle must start from a depot and return to it. The number of vehicles is flexible, and therefore the constraints can always be satisfied. For a formal problem description, see Chapter 2. The routing cost is proportional to the total distance of all tours.

Finally, in the scenario of uncertain demands, it is proposed here to introduce stockout cost by computing the expected value of the total stockout in the scenario as:

$$f_3 = \sum_{t=1}^{T} \sum_{i=1}^{n} S_{i,t} \rightarrow \min \tag{7.6}$$

Here $S_{i,t}$ is the stockout cost of customer v_i in day t. This way, a multi-objective optimization problem with three objectives has been defined.

7.2.2 · Multi-objective Optimization

Miettinen [Miettinen, 1999] distinguishes the *a-priori* and *a-posteriori* methods for multi-objective optimization. In the *a-priori* approach, first an aggregating utility function is defined and then the optimization is carried out. In the *a-posteriori* approach a set of non-dominated solutions is computed and presented to the decision maker for selection and trade-off assessment. In this work, the *a-posteriori* approach is applied and algorithms are proposed to compute approximations to the non-dominated set (or Pareto front) of the multi-objective optimization problem.

In order to introduce a preorder on the search space, the usual Pareto dominance relation is used. A point $\mathbf{x}^{(1)} \in \mathcal{X}$ is said to *(Pareto) dominate* a point $\mathbf{x}^{(2)} \in \mathcal{X}$, if and only if for all objective function values of $\mathbf{x}^{(1)}$ are not worse than values of $\mathbf{x}^{(2)}$, and $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in at least one objective function value. The set of non-dominated solutions in $\mathcal{X}$ will be called the *efficient set* $\mathcal{X}^*$ and its image set $PF = \{\mathbf{f}(\mathbf{x}) | \, \mathbf{x} \in \mathcal{X}^*\}$ is commonly termed the *Pareto front*.

A performance indicator for how well a Pareto front is covered is the *hypervolume indicator* of a population. The hypervolume indicator defined in this way is a standard indicator in Pareto optimization and it has been shown by Zitzler et al. [Zitzler et al., 2003] that it has favorable properties compared to all other indicators when the goal is to obtain a well distributed and well converged approximation of the Pareto front and the quality of a Pareto front approximation needs to be measured without knowledge of the ‘true‘ Pareto front. It is defined by the Lebesgue measure of the dominated subspace

in the objective space:

$$HV(\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}) = \lambda\left(\bigcup_{i=1}^{n}\left[\mathbf{f}(\mathbf{x}^{(i)}), \mathbf{r}\right]\right) \tag{7.7}$$

Here λ denotes the Lebesgue measure in dimension m. $\mathbf{r}$ is the reference point of this problem. It is assumed that the objective function vectors $\mathbf{f}(\mathbf{x}^{(i)})$ dominate $\mathbf{r}$, so that the orthogonal ranges $\left[\mathbf{f}(\mathbf{x}^{(i)}), \mathbf{r}\right] \subset \mathbb{R}^m$ are well defined for $i = 1, \ldots, n$.

Informally, the hypervolume indicator can be defined as a measure for the size of the set of dominated objective vectors, for which there exists a solution $\mathbf{x} \in \mathcal{X}$ that is either better or equal to the objective vector. To make this measure finite, the measured set is cut from above by a reference point. Increasing the hypervolume indicator means therefore to increase the amount of available options.

Figure 7.1 displays a projection of some population of 5 points to the objective space for some hypothetical functions. The points $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, and $\mathbf{x}^{(4)}$ are non-dominated and the point $\mathbf{x}^{(5)}$ is dominated. For each point the dominated part of the objective space cut by the reference point is indicated by a gray shaded box and surrounded by a dark gray line. The hypervolume indicator of this population and reference point r equals the size of the entire gray shaded area.

In this chapter algorithms are compared which seek to maximize the hypervolume indicator over the set of all subsets of $\mathcal{X}$ of size n. The algorithms produce a sequence of so called approximation sets that gradually converge to a diverse approximation of the Pareto front with maximal hypervolume indicator value. Indicator-based multi-objective optimization (IMO) seeks to improve the performance indicator of a solution set and by doing so they achieve a good approximation to the Pareto front.

7.3 · Multi-objective Optimization Algorithms

Next we will introduce four algorithms that will be used for solving the tri-objective inventory routing problem. All algorithms are population-based metaheuristics, that is algorithms that maintain a population of search points. They aim to move the points closer to and across the Pareto front.

We will start with the introduction of the two IMO methods, namely the SMS-EMOA and MOCOPS. The SMS-EMOA is an evolutionary algorithm and the MOCOPS is a swarm based algorithm. In mathematical programming, in evolutionary algorithms and in particle swarm algorithms, different terminologies are used. An

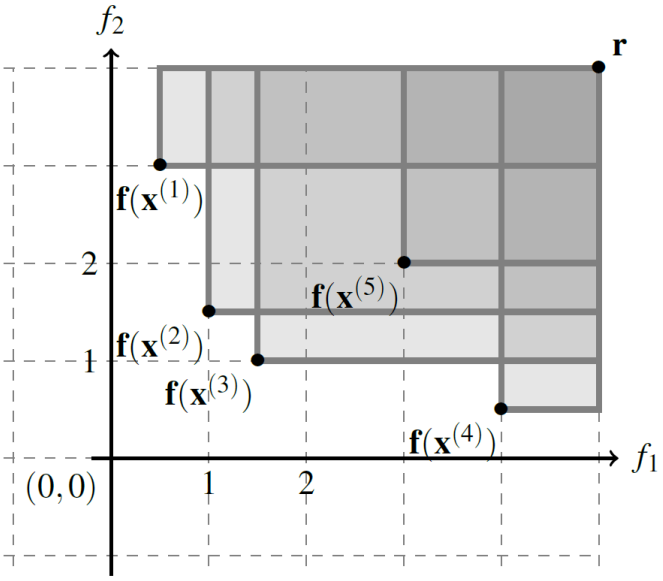

Figure 7.1 Projection of a population to the objective space.

Table 7.1 Terminology in set-oriented optimization.

Evolutionary Optimization	Swarm Optimization	Mathematical Programming
Population	Swarm	Approximation set
Individual	Particle	Decision vector solution
Generation	Position update	Iteration

approximation set (to the efficient set) in mathematical programming is called a population of individuals in evolutionary algorithms and a swarm of particles in swarm algorithms. Iterations of an iterative search algorithm correspond to generations in evolutionary algorithms and position updates in swarm algorithms. The terminology is summarized in Table 7.1.

In this chapter, both SMS-EMOA and MOCOPS aim to maximize the hypervolume indicator of a population solution and thereby create a diversified set on the Pareto front. SMS-EMOA is an evolutionary algorithm with a single point replacement selection scheme (steady state selection), and MOCOPS is a swarm based algorithm where each point in the population is viewed as a search agent that seeks to improve its individual

contribution to the hypervolume indicator. The structure of the solvers for inventory routing problem can be seen in Figure 7.2. The procedure starts with the initialization of the population. Based on the delivery frequencies given by the initialization, customers are assigned to be served at a certain day. For each day, how much quantity of products should be delivered would be decided. The optimization of the delivery process in itself constitutes a classical vehicle routing problem. After the evaluation of one solution is done, the hypervolume contribution of each customer is calculated. Then, the population is updated according to the hypervolume contribution of each customer. This loop is reiterated until a user defined termination criterion is reached.

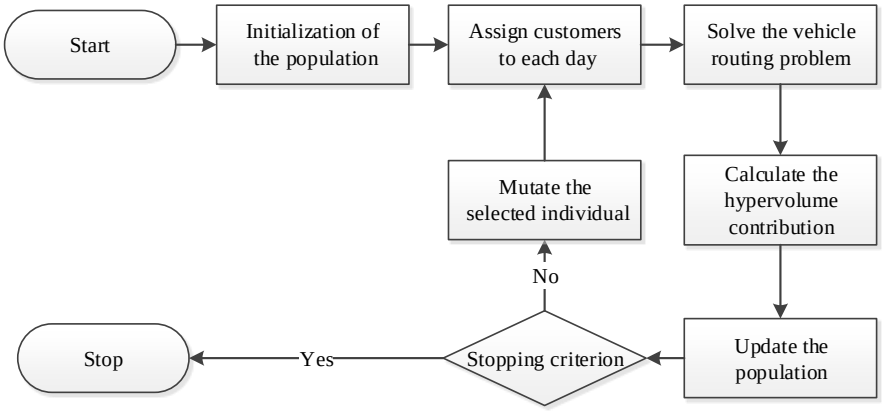

Figure 7.2 The structure of multi-objective optimization algorithm for inventory routing problem

7.3.1 · Initialization Procedure

In order to make the population evenly spread in the search space, a special initialization procedure is implemented which is discussed in [Geiger and Sevaux, 2011]. First, solutions are initialized by providing an identical delivery period starting with 1 and increasing in steps of 1 until no non-dominated alternatives can be added to the population anymore. For example, if there are 3 customers to be served with a max period of 3, then the initial delivery periods are π= { [1,1,1,1], [2,2,2,2], [3,3,3,3]}. The delivery periods are randomly set to j or $j + 1$, which can fill the gaps between the

purely identical delivery period solutions. The initial delivery periods with size of 8 solutions would be, for instance, π= { [1,1,1,1], [2,2,2,2], [3,3,3,3], [2,1,1,1], [2,2,2,1], [2,2,3,1], [3,3,2,2], [3,2,2,3]}. It will be decided based on the delivery frequencies, when and how many of the products should be delivered. For each period, a classical vehicle routing problem is defined.

Since the vehicle routing problem is an NP-hard problem [Kimura and Ikeguchi, 2007, Yang et al., 2015a], it is really time consuming to solve this problem repeatedly. Therefore, the classical savings heuristic [Clarke and Wright, 1964] is applied to construct the solution of the vehicle routing problem. This algorithm is very fast and it is well-known that it finds good solutions on a wide range of problems. A number of heuristic approaches such as the genetic algorithm and the ant colony algorithm are able to improve the performance of the VRP [Forcael et al., 2014, Gambardella et al., 1999a, Siddique and Adeli, 2013, Solomon, 1987]. However, these algorithms are not chosen in this chapter, because they are really time consuming.

7.3.2 · Evolutionary Algorithm: SMS-EMOA

The basic iteration of SMS-EMOA creates a new individual by mutation and crossover operators and then adds it to the population. From this large population one solution is removed based on dominance rank and hypervolume contribution. Given a population P, the hypervolume contribution $\Delta H(\mathbf{x})$ is defined as: $\Delta HV(\mathbf{x}) = HV(P) - HV(P \backslash \{\mathbf{x}\})$. In the SMS-EMOA, points can be removed and new points might appear in the course of the evolution [Beume et al., 2007, Emmerich et al., 2005]. Viewed as stochastic systems, it is a branching process, with the "birth event" creating a new branch, and the "death event" terminating a branch in the population. The SMS-EMOA is otherwise very similar to the swarm-based algorithm, as it bases the decisions on hypervolume contributions of points.

For the visualization of a population with 3-D objective vectors, see Figure 7.3. An asymptotically optimal algorithm for computing all hypervolume contributions in a population of 3-D vectors has been described by Emmerich and Fonseca [Emmerich and Fonseca, 2011]. The running time complexity is $O(|P| \log |P|)$. Here $|P|$ is the population size, and this step is therefore up to a constant factor as fast as computing the hypervolume of a population.

The simple version of the SMS-EMOA that was used in our experiments is outlined in Algorithm 9. After initializing a population, in each iteration, a new solution is

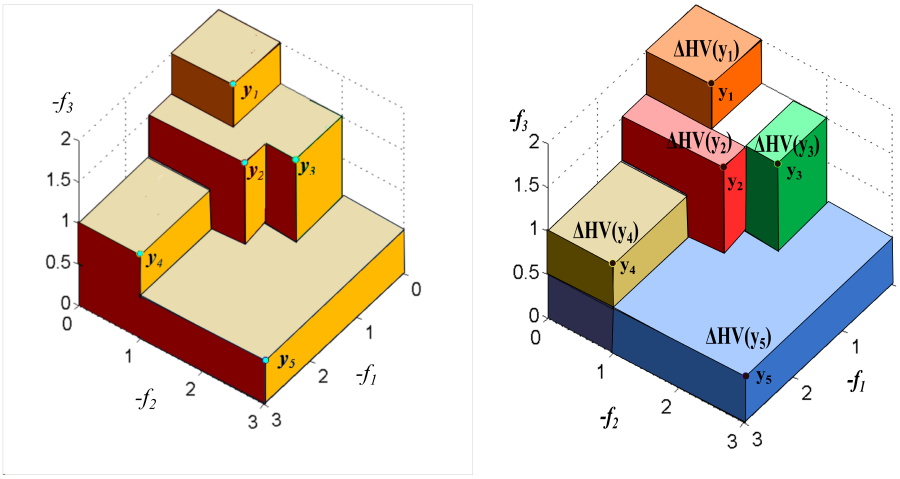

Figure 7.3 Left: Hypervolume dominated by a set of 3-D objective vectors. Right: *Hypervolume contributions* of these vectors.

created by mutating an existing solution *(Mutate(.))*. Then it is added to the population. Subsequently, the hypervolume contributions of all population members are computed and those members with the least contribution are determined. From these chosen members a random one is chosen and it is then discarded from the population. Due to the last step, the population size is kept constant and there is a selection pressure towards sets which cover more hypervolume of the search space. Note that dominated solutions have a hypervolume contribution of zero and solutions that are non-dominated with respect to P have a positive contribution. Therefore, dominated solutions are always discarded first.

7.3.3 · Multi-objective Cooperative Particle Swarm

As opposed to the evolutionary algorithm, multi-objective optimization cooperative particle swarm is a randomized search heuristics where a swarm of particles moves gradually towards a hypervolume-maximizing solution set driven by randomized modification operators and interaction between the particles. It could be described as a multi-trajectory stochastic process. It is related to the particle swarm optimization algorithms (PSO) that have been suggested for other problems in the literature [Boulkaibet et al., 2015, Rodríguez and Reggia, 2009].

In conventional PSO algorithms, the swarm is driven by a leader, who is the

Algorithm 9 S-Metric Selection Evolutionary Algorithm (SMS-EMOA)*

Input initial population P_0
while termination criterion is not reached **do**
  $t \leftarrow t + 1$
  $x^{(s)}$: Random select on individual from the population
  $\mathbf{x}_{old} = \mathbf{x}^{(s)}$
  $\mathbf{x}_{new} = Mutate(\mathbf{x}_{old})$
  $P \leftarrow P_t \cup \{\mathbf{x}_{new}\}$
  ▷ Determine the (set of) least hypervolume contributors
  $\mathcal{L} \leftarrow \arg\min_{\mathbf{x} \in P} \Delta HV(\mathbf{x})$
  Chose randomly a solution $x^{(s)}$ in $\mathcal{L}$
  $P \leftarrow P_t \setminus \{\mathbf{x}^{(s)}\}$
end while
(*Simple version with random selection among dominated solutions.)

currently best individual in a population, and by local memories of particles of their so-far best positions. In single-objective optimization such processes will typically converge to a local optimum, or sometimes even to a global optimum. In multi-objective optimization such an approach could be easily used to find a single point on the Pareto front, but it is not well suited to distribute points across the Pareto front, because all particles strive to resemble the leader which is counter productive when searching for a diverse set of solutions. To a certain extent this can be compensated by assigning local leaders, but it makes the algorithm quite complicated and adds more parameters to the algorithm (i.e., number of leaders).

In multi-objective optimization there is no definition of a best solution and thus there is no obvious choice for a leader individual. However, one could for instance form subpopulations or use local metrics. The use of the traditional PSO for multi-objective optimization problems has been addressed already in the literature, both in the context of general multi-objective optimization [Coello and Lechuga, 2002] and for finding Pareto fronts that maximize the hypervolume indicator [Mostaghim et al., 2007, Mostaghim and Teich, 2003]. Both approaches lead to algorithms that produce good approximations to Pareto fronts. The version of multi-objective particle swarm optimization used in this chapter resembles closely [Mostaghim et al., 2007], but it is using a simplified leader free selection and a different variation scheme.

For the cooperative particle swarm algorithm, the following properties distinguish it from previous swarm-based approaches:

- Leader-free: The particles in the population cooperate in covering the Pareto front, instead of competing with each other. There is no leader in the swarm; each particle strives to contribute to the global performance of the swarm by improving its position.

- Indicator-based: The algorithm seeks to maximize a unary performance indicator. Here the hypervolume indicator is used.

The new approach is deliberately kept very simple. This is for two reasons: Firstly we want to demonstrate that only a few essential components are needed to steer a swarm towards a Pareto front. Secondly, the simplicity will make the algorithm easier accessible to a rigorous theoretical analysis.

Here we term this approach multi-objective optimization cooperative particle swarm. The particles in the swarm strive to contribute as much as possible to the team performance of the population. By doing so, they seek to contribute in different ways to the goal of covering the Pareto front (here stated as hypervolume maximization).

In traditional PSO algorithms, a swarm will thus lead to unity, whereas in multi-objective optimization you need the diversification to adequately approach the Pareto set. The pseudo code for the proposed MOCOPS algorithm is given in Algorithm 10. It starts with a randomly initialization of a set of particles. Then, in each iteration of the algorithm, a particle is randomly selected and a small random variation of this particle is generated by adding a random perturbation (Mutate(.)).

Whether the particle would move to the new position or not is based on the hypervolume contribution of the mutated position and the original position. Firstly, it will be tested which one of the two positions leads to a better hypervolume indicator of the population. Secondly, if both positions are equally good (which will typically occur for dominated solutions), the point that has a better value in the linearly aggregated objective function with equal weights is considered. Note that if one solution is dominated by the other solution it will also be considered better in the latter comparison (because of the positive weighting). Therefore, eventually all solutions will strive towards the non-dominated front and then their hypervolume contributions will be considered. The cycle continues with picking a random particle again. Care must be taken to ensure $\vec{\mathbf{x}}_{new} \in \mathbb{S}$ (e.g. by rejecting infeasible vectors).

One iteration of the bi-objective MOCOPS algorithm can be performed with a time complexity in $O(|P|log|P|)$. Here $|P|$ is the size of the Pareto front set. This

Algorithm 10 Multi-objective Optimization Cooperative Particle Swarm (MOCOPS)

Input initial population P_0
while termination criterion is not reached **do**
  $t \leftarrow t + 1$
  $x^{(s)}$: Random select on individual from the population
  $\mathbf{x}_{old} = \mathbf{x}^{(s)}$
  $P \leftarrow P_t \setminus \{\mathbf{x}^{(s)}\}$
  ▷ Try to improve position of particle $\mathbf{x}^{(s)}$
  $\mathbf{x}_{new} = Mutate(\mathbf{x}_{old})$
  if $HV(P \cup \{\mathbf{x}_{new}\}) > HV(P \cup \{\mathbf{x}_{old}\})$ **then**
    $P_t = P \cup \{\mathbf{x}_{new}\}$
  else if $HV(P \cup \{\mathbf{x}_{new}\}) < HV(P \cup \{\mathbf{x}_{old}\})$ **then**
    $P_t = P \cup \{\mathbf{x}_{old}\}$
  else if $f_1(\mathbf{x}_{new}) + f_2(\mathbf{x}_{new}) < f_1(\mathbf{x}_{old}) + f_2(\mathbf{x}_{old})$ **then**
    $P_t = P \cup \{\mathbf{x}_{new}\}$
  else
    $P_t = P \cup \{\mathbf{x}_{old}\}$
  end if
end while
Return P_t

can be achieved by using a dimension sweep algorithm and an AVL tree [Emmerich and Fonseca, 2011]. However, by implementing the algorithm as an online algorithm, that is using incremental update steps, we can compute a single iteration with time complexity in $O(log|P|)$ (amortized over the number of iterations) [Hupkens and Emmerich, 2013]. This algorithm dynamically updates the AVL tree keeping non-dominated points sorted in the first coordinate. Fast hypervolume update schemes with linear time complexity are also known for three objective functions [Guerreiro et al., 2012]. The computational complexity is expected to grow exponentially in the number of objective functions [Bringmann and Friedrich, 2009]. For this reason the scheme probably does not lend itself very well for many-objective optimization.

On first glance MOCOPS and SMS-EMOA look similar. However, there is an important difference. The difference is given by the stochastic dynamics of the two algorithms. The SMS-EMOA is a branching process - a single (parent) point can generate multiple offspring over time or it might also disappear without ever producing an offspring. In the MOCOPS a point is either preserved or replaced by a better neighboring point (local move). In this sense the dynamics of the MOCOPS is simpler,

but it is less likely in the MOCOPS to abandon subspaces entirely, as it might occur in the SMS-EMOA.

7.3.4 · NSGA-II

NSGA-II is a classical multi-objective algorithm proposed by Deb et al. [Deb et al., 2002]. It is the most commonly applied evolutionary algorithm for multi-objective optimization and serves here as a reference algorithm. The algorithm works as follows: In each iteration, the tournament selection, recombination and mutation operator are used to generate the offspring. Then a fast non-dominated sorting method and crowding distance as a secondary ranking order are applied to select the population for the next generation which can maintain the diversity of the population. Since in the IRP the solution is an integer vector, the operators are adapted to a perturbation.

7.3.5 · Algorithm by Huber, Geiger and Sevaux

In addition, we extended the state-of-the-art decomposition based bi-objective optimization algorithm for bi-objective inventory routing by Huber, Geiger, and Sevaux (HGS14) to the 3-D case [Huber et al., 2013]. They use the same initialization procedure as described in this section. All non-dominated initial solutions are added to an archive. Then, in the improvement procedure, a set of reference points are selected. To select the representatives of solutions, the objectives of solutions are normalized and the solutions that minimize the distance to these reference points are selected from the archive in each iteration. For each reference point the nearest neighbor in the Chebychev distance is selected from the archive and then improved by the local search method, also minimizing this Chebychev distance. New solutions will be added to the archive and the dominated solutions in the archive are deleted. Then new reference points are selected and determined based on the new non-dominated set, and the procedure is repeated. The process is terminated after a prescribed number of steps. The local search is exhaustive and uses the local search operator increment or decrement of a single vector position by 1.

As a major extension, we suggest the reference points selection methods from 2-D to 3-D. A regular pattern is used to distribute the reference points on the boundaries of a simplex which dominates the current Pareto front. Suppose the space in each dimension is divided into n intervals, the value of reference points in each dimension should belong to the set $s = \{s_i | s_i = i/n, i = 0, \ldots, n\}$. Then the reference points set can be defined

as $R = \{\overrightarrow{r} | \forall i \in \{1,2,3\} : r_i \in s \wedge \exists j \in \{1,2,3\} : r_j = 0\}$. For instance, for $n = 3$, we construct the reference points by the base pattern {(0,0,0); (0,0,1/3); (0,0,2/3); (0,0,1); (0,1/3,0); (0,1/3,1/3); (0,1/3,2/3); (0,2/3,0); (0,2/3,1/3); (0,1,0); (1/3,0,0); (1/3,0,1/3); (1/3,0,2/3); (1/3,1/3,0); (1/3,2/3,0); (2/3,0,0); (2/3,0,1/3); (2/3,1/3,0); (1,0,0)}. This example of reference points can be seen in Figure 7.4 and is used in the studies for this chapter. The base pattern is rescaled by multiplying each component of a base vector by the maximal value for that dimension in the current Pareto front approximation. Note that, NSGA-III also use the concept of reference points. However, they use a reference plane to select them [Seada and Deb, 2015].

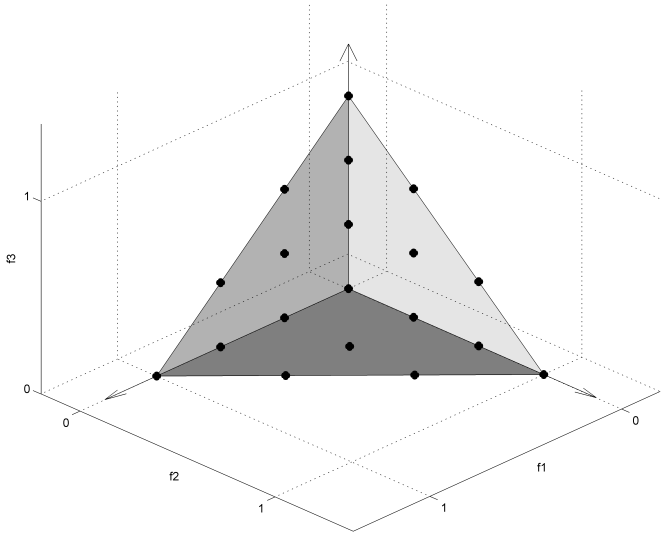

Figure 7.4 An example of refenence points selected on 3-D method

7.4 · Computing Results

In this section, the test instances are describe and the parameter settings are set. More important, the results of the bi-objective IRP results and the tri-objective IRP with uncertain demand results are shown and discussed.

7.4.1 · Test Instances

The test benchmark instances are proposed by Sevaux et al. [Geiger and Sevaux, 2011], which are available from `http://logistik.hsu-hh.de/IRP`. The concept of periodical demand is added to the classical VRP benchmark data. They generated a total of $T = 30$ periodical demand of all customers. In each period, the given demand range from -25% to 25% around the average demand. The number of customers for problem GS-01, GS-02, GS-03, GS-04 and GS-05 are 55, 75, 100, 150 and 200, respectively. In this chapter, an extension is made to generate the uncertain demand. The periodical demand of each customer is given as a random number with a Poisson distribution whose expected value is μ, the average periodical demand which is given in Sevaux's benchmark. For a typical value of μ, that is $\mu = 20$, the probability density function of this Poisson distribution can be seen in Figure 7.5. The Poisson distribution is used, because it models the sum of Bernoulli variables and the probability of the event that a customer buys a product at a shop where an inventory is kept can be modeled as a Bernoulli random variable.

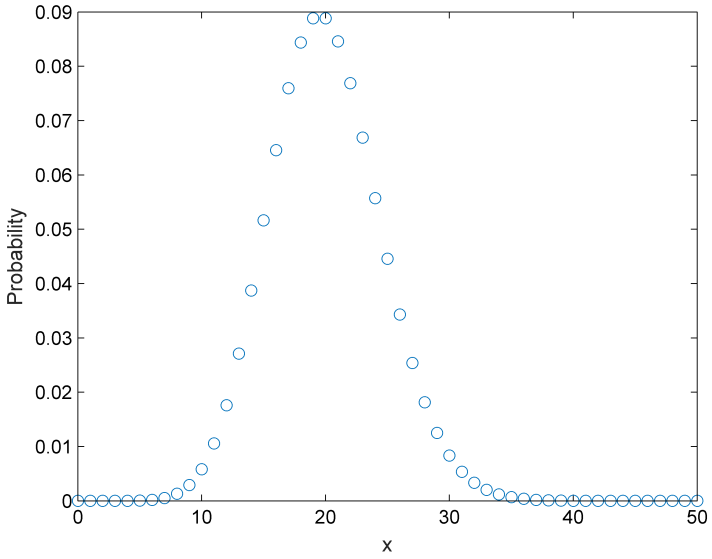

Figure 7.5 The specific probability density function of the demand

Table 7.2 The reference points for the hypervolume indicator used in different instances

Instances	reference points
GS-01	(233100,72070.43,20895)
GS-02	(409200,108925.70,36949)
GS-03	(437400,149682.68,39494)
GS-04	(670500,220815.05,60500)
GS-05	(955800,288252.24,86207)

7.4.2 • Parameter Settings

All four algorithms are tested on the bi-objective and tri-objective inventory routing problem. The matlab code is made available via http://natcomp.liacs.nl/index.php?page=code. The algorithm setup was as follows: For the mutation integer mutation with geometrical distribution is used [Li et al., 2006] and if interval boundaries are exceeded, they are set to the boundary. The vehicle routing was done with the parameter-free savings heuristics. All other problem data was chosen according to [Geiger and Sevaux, 2011]. The runs were conducted for 5000 evaluations of the objective functions.

For the hypervolume indicator, the reference point $R = (r_1, r_2, r_3)$ was used. In order to determine a reference point, an upper bound for all objective function values was required: For the first objective (inventory cost) r_1, the assumption is that all inventories are always at their maximum allowed level. For the second objective (routing cost) r_2, the assumption is that every day each customer is served with one vehicle. Finally, for the third objective (stockout cost) r_3, the assumption is no deliveries would take place. The reference points for the hypervolume indicator used in different test instances can be seen in Table 7.2.

For the population based algorithm, population size is a sensitive parameter that should be carefully treated. In this chapter, parameter tuning is done by changing the population size from 10 to 100 with an interval of 10. In Figure 7.6, it can be seen that the hypervolume indicator values increase with the population before 80 and then drop gradually. The results show that the population size should be set to 80 to reach optimal hypervolume indicator values. Due to the super-linearly increasing computational effort in the population size, larger population sizes should be avoided.

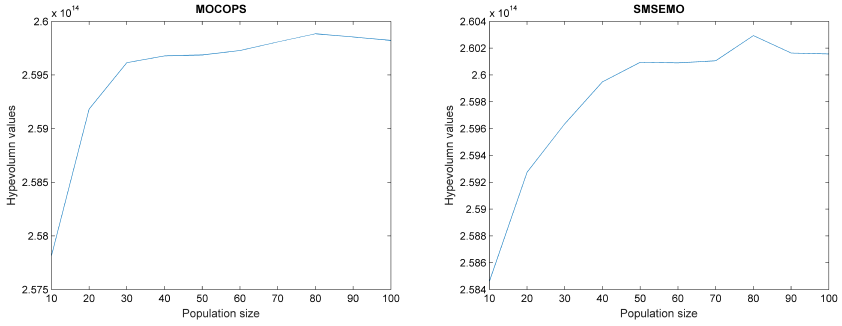

Figure 7.6 Left: The hypervolume indicator of MOCOPS changes with population sizes. Right: The hypervolume indicator of SMS-EMOA changes with population sizes.

7.4.3 · Biobjective IRP Experiment

The first comparison is on the bi-objective problem. Figure 7.7 shows results of the comparison. Clearly the results of the algorithms seem to be very similar. The leftmost point appears as an isolated solution around (0.5, 1.6). The delivery frequencies for all customers are 1, which means every day all customers are served. Moving further right by holding more inventories would make a big reduction of routing cost. However, after the inventory cost reaches 5, the inventory cost would not increase so much any more, because maximum inventory levels are reached. The results also show that NSGA-II, SMS-EMOA and MOCOPS can get similar results in the bi-objective inventory routing problem. The algorithm of Huber, Geiger and Sevaux performs slightly better for this problem.

7.4.4 · IRP with Uncertain Demand

In this experiment, all four algorithms are executed 10 times on 5 test instances with the number of customers varying from 50 to 200. Since Sevaux et al. only provided results of bi-objective IRP, there is no existing results that can be used as a comparison on this problem. Therefore, these algorithms are compared to each other, which provide some preliminary insights into the Pareto optimal front of the IRP with uncertain demand.

In order to visualize Pareto front approximations in the 3-D case, attainment surface plots are used. The attainment surface separates the dominated subspace (grey volume) from the non-dominated subspace. Using the same notation than in Section 7.3.2 it can be defined as the set of points that are only weakly dominated by some points

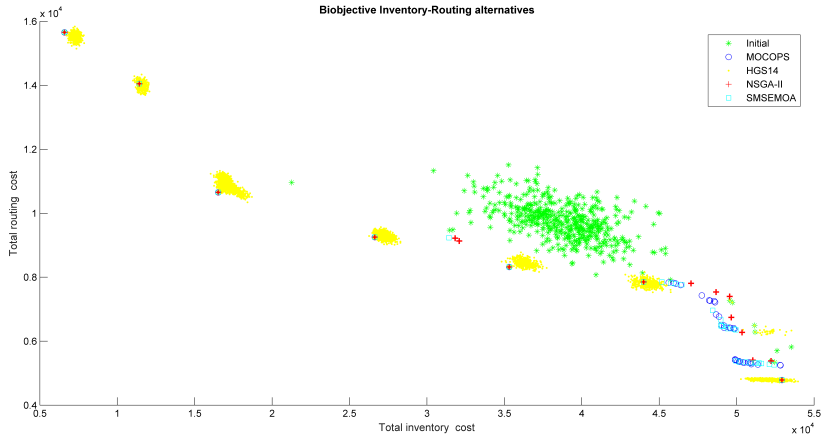

Figure 7.7 Pareto front approximations of the bicriteria problem obtained with different algorithms.

in the Pareto front set P, i.e. $\{y|\exists y' \in P : y' \leq y \wedge \neg\exists y'' \in P : y'' < y\}$, where '$\leq$' denotes the weak componentwise order and '$<$' denotes the strict componentwise order. In order to more accurately assess the quality of single points we also provided the three projections as a scatter plot. An example of 3D visualization of a Pareto front approximation generated by these two algorithms on problem GS-01 can be seen in Figure 7.8 and Figure 7.9. Results of other test problems are shown in Figure 7.10 and Figure 7.11.

As we can see, SMS-EMOA and MOCOPS produced very similar results, which could be interpreted as an indication that a good approximation to the hypervolume maximal front was obtained. The interpretation of 3-D results is more involved, as three trade-offs need to be taken into account: Firstly, from the projection to routing cost and inventory cost we obtain a similar set of non-dominated solutions, in the 2-D projection, to the 2-D study. This means there is a clear conflict between the inventory cost and routing cost. Then, from the projection of routing cost and stockout cost, it can be seen that there is also a correlation between them. In order to decrease the stockout cost, the decision makers could try to make the delivery frequency smaller, which would increase the routing cost. The stockout cost would reach the optimum at the point where the routing cost is around 8000. The maximal inventory levels are often reached in this case. The relation between the stockout cost and the inventory cost is special. It is either possible to improve the stockout cost by increasing the routing cost,

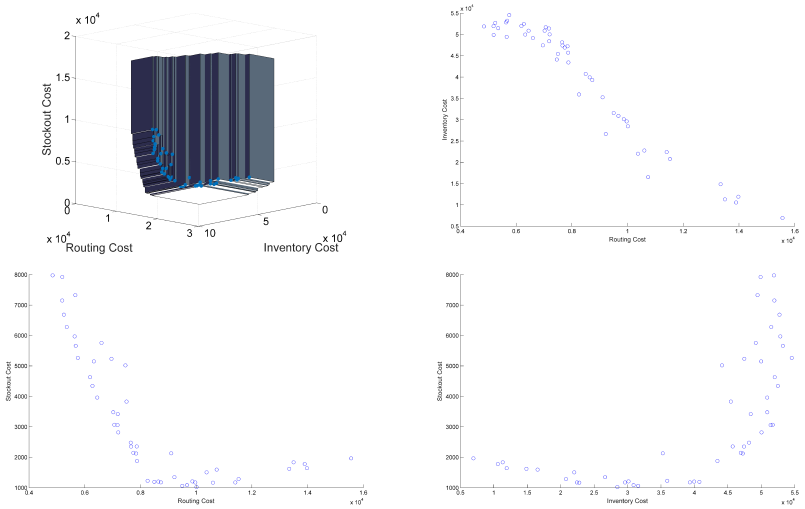

Figure 7.8 Plots of tricriteria MOCOPS for IRP with uncertain demands : Points and dominated subspace (upper left), routing cost vs. inventory cost (upper right), routing cost vs. stockout cost (lower left), inventory cost vs. stockout cost (lower right).

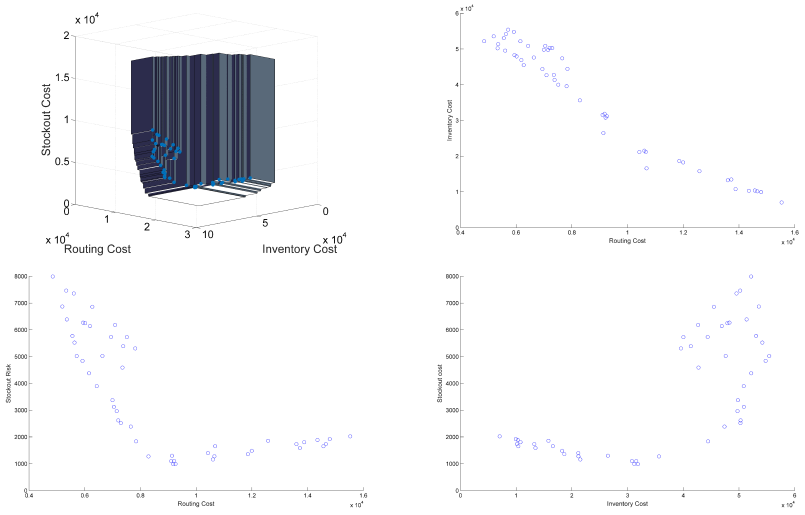

Figure 7.9 Plots of tricriteria SMS-EMOA for IRP with uncertain demands: Points and dominated subspace (upper left), routing cost vs. inventory cost (upper right), routing cost vs. stockout cost (lower left), inventory cost vs. stockout cost (lower right).

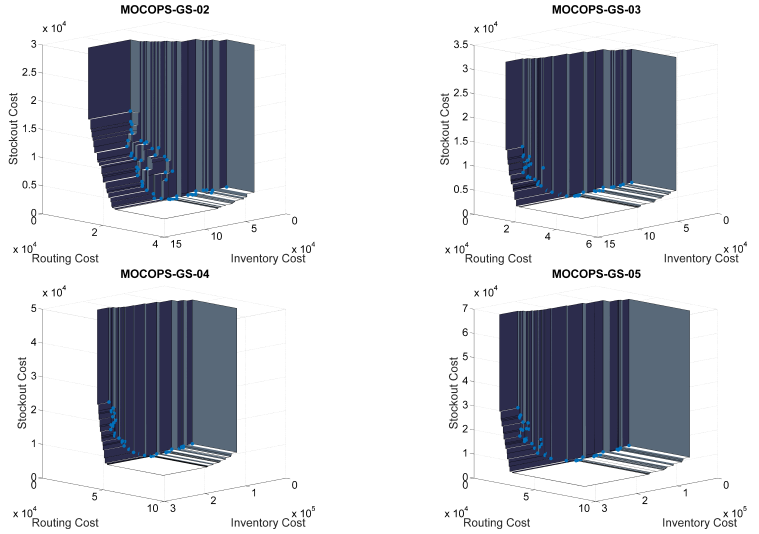

Figure 7.10 3D Plots of tricriteria MOCOPS for IRP with uncertain demands

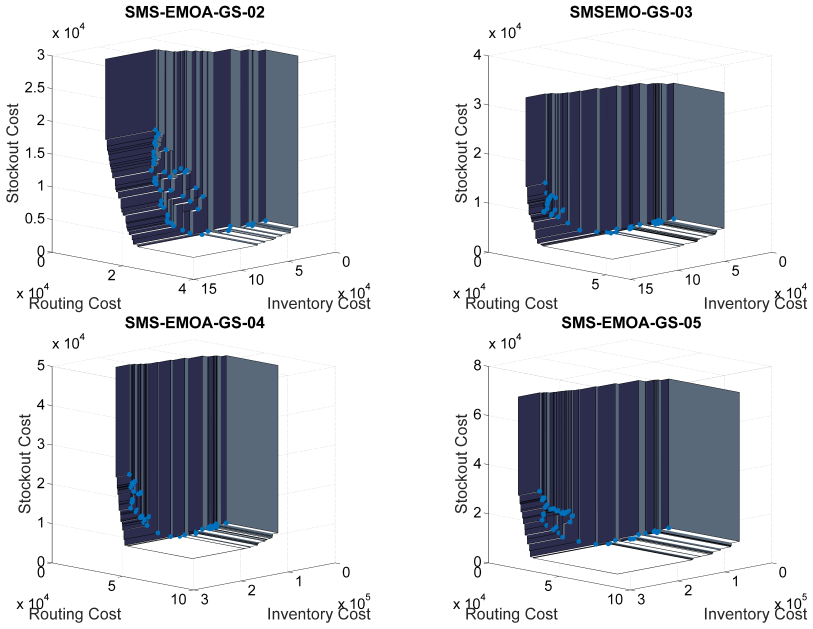

Figure 7.11 3D Plots of tricriteria SMS-EMOA for IRP with uncertain demands

or by increasing the inventory cost. For very high values of the π vector, the routing cost will be low but the demands of customers cannot be satisfied. This will eventually increase the stockout cost. There is also a weak decrease of the stockout cost with growing inventory costs before the inventory reaches a value of ca. 30000. Then the stockout cost grows rapidly and progressively with increasing inventory cost. These points have a high inventory cost and a high stockout cost. The reason they are still non-dominated is that they have a small routing cost. The infrequent delivery causes the raise of stockout costs.

In order to compare the quality of the Pareto front approximations of these algorithms, the hypervolume of all test instances are given in Table 7.3. The results show that these four algorithms could get a similar hypervolume for all test instances. HGS14 works better in small scale instances such as GS-01 and GS-02. SMS-EMOA has a good performance on GS-03. However, the MOCOPS could achieve a better performance in large scale instance such as GS-04 and GS-05. The NSGA-II seems to run faster than other algorithms in most cases. The MOCOPS runs faster on GS-02 and GS-03. This is due to the fact that it does not require hypervolume computations. However, the future work might improve the complexity of the SMS-EMOA and the MOCOPS by using incremental updates of hypervolume calculations.

7.5 · Summary

In this chapter the bi-objective inventory routing problem is extended to a tri-objective inventory routing problem by introducing the stockout cost as a third objective function. This is important in the context of uncertain demand distributions. A Poisson random variable is used to model the demand and resulting stockout costs.

Four population based metaheuristics are studied to compute the 3-D Pareto front of the tri-objective inventory routing problem, namely NSGA-II, HGS14, SMS-EMOA and MOCOPS. The NSGA-II and the SMS-EMOA are state-of-the-art evolutionary optimization methods, whereas the MOCOPS is a new, customized version of a swarm based optimizer. It was shown that all algorithms achieved similar results in various runs and this consensus makes us believe that they were all able to find near optimal approximation sets to the Pareto front. On small instances, the HGS14 performed best and for big instances the MOCOPS is the best.

The resulting Pareto front revealed interesting insights into the trade-off between different objectives. The results confirm that the stockout cost is in conflict with the

Table 7.3 Hypervolume indicator values and computation time of all problems

		NSGA-II			HGS14			SMS-EMOA			MOCOPOS		
Test Case	Pop Size	HV (10^{11})	Std (10^{11})	CT (h)	HV (10^{11})	Std (10^{11})	CT (h)	HV (10^{11})	Std (10^{11})	CT (h)	HV (10^{11})	Std (10^{11})	CT (h)
GS-01	50	2932.14	5.67	**0.7**	**2953.12**	5.21	1.2	2952.57	**3.60**	0.8	2948.96	4.22	0.8
GS-02	75	13640.85	27.25	1.4	**13747.60**	16.78	2.1	13694.38	21.58	1.5	13728.82	**11.55**	**1.3**
GS-03	100	22014.79	**22.42**	2.2	22088.44	25.91	3.5	**22093.473**	23.199	2.6	22079.416	25.695	**2.1**
GS-04	150	76362.16	152.40	**4.2**	76691.19	**47.17**	6.8	76587.74	112.64	5.0	**76709.02**	77.63	4.5
GS-05	200	202419.1	182.13	**6.8**	203044.0	**31.97**	9.9	202971.27	209.33	8.1	**203093.22**	122.26	7.3

distance cost and inventory cost. To reduce stockout cost one has to either accept higher inventory costs or higher distance costs. This is observed across several benchmark instances and the 3-D Pareto fronts have a similar shape for the different benchmark problems. Based on these empirical findings and their explanation, we can conclude that there is a something like a typical (parabolic) Pareto front shape for a tri-objective vehicle routing problem, and it is possible to compute these Pareto fronts in practically feasible time. Computation times range from 20 minutes (50 customers) to 8 hours (200 customers) on a typical desktop PC.

In the future, in order to allow for real-time optimization, it will be interesting to further reduce the computation time for large instances. For instance, this could be achieved by using more efficient and precise search procedures. Moreover, more advanced algorithms should be implemented to solve the subordinated vehicle routing problem, such as ant colony algorithms or genetic algorithms. Finally, an interesting investigation will be the study of real world applications with empirically fitted demand distributions including pilot experiments.